CS106A
Midterm Review

Miles Seiver
3p - 5p
27 October 2013

problems by Keith Schwarz

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

Mehran Sahami
CS106A

Handout #28
October 23, 2013

Practice Midterm Examination

Midterm Time: Tuesday, October 29th, 7:00P.M.–9:00P.M.

Midterm Location (by last name):

Last name starts with A-L: go to Dinkelspiel Auditorium
Last name starts with M-T: go to Hewlett 200
Last name starts with U-Z: go to 320-105

Portions of this handout by Eric Roberts and Patrick Young

This handout is intended to give you practice solving problems that are comparable in format and difficulty to those which will appear on the midterm examination.

Exam is open book, open notes, closed computer

The examination is open-book (specifically the course textbook *The Art and Science of Java* and the Karel the Robot coursereader) and you may make use of any handouts, course notes/slides, printouts of your programs or other notes you've taken in the class. You may not, however, use a computer of any kind (i.e., you cannot use laptops on the exam).

Coverage

The midterm exam covers the material presented in class through today (October 23rd), which means that you are responsible for the Karel material plus Chapters 1 through 9 of the class textbook *The Art and Science of Java*, plus the use of mouse listeners from Chapter 10 (sections 10.1-10.4). Data files (which will be covered in class on Friday) are not covered on the exam.

General instructions

Answer each of the questions included in the exam. Write all of your answers directly on the examination paper, including any work that you wish to be considered for partial credit.

Each question is marked with the number of points assigned to that problem. The total number of points is 120. We intend for the number of points to be roughly comparable to the number of minutes you should spend on that problem.

In all questions, you may include methods or definitions that have been developed in the course, either by writing the `import` line for the appropriate package or by giving the name of the

Midterm Time: Tuesday, October 29th, 7:00P.M.–9:00P.M.

Midterm Location (by last name):

- **Last name starts with A-L: go to Dinkelspiel Auditorium**
- **Last name starts with M-T: go to Hewlett 200**
- **Last name starts with U-Z: go to 320-105**

Exam is open book, open notes, closed computer

The examination is open-book (specifically the course textbook *The Art and Science of Java* and the Karel the Robot coursereader) and you may make use of any handouts, course notes/slides, printouts of your programs or other notes you've taken in the class. You may not, however, use a computer of any kind (i.e., you cannot use laptops on the exam).

Coverage

The midterm exam covers the material presented in class through today (October 23rd), which means that you are responsible for the Karel material plus Chapters 1 through 9 of the class textbook *The Art and Science of Java*, plus the use of mouse listeners from Chapter 10 (sections 10.1-10.4). Data files (which will be covered in class on Friday) are not covered on the exam.

Each question is marked with the number of points assigned to that problem. The total number of points is 120. We intend for the number of points to be roughly comparable to the number of minutes you should spend on that problem.

Other tips

Other tips

- Style doesn’t matter. You don’t need to comment.

Other tips

- Style doesn’t matter. You don’t need to comment.
- Don’t worry about imports.

Other tips

- Style doesn’t matter. You don’t need to comment.
- Don’t worry about imports.
- Pseudocode credit is capped at 50%.

Other tips

- Style doesn’t matter. You don’t need to comment.
- Don’t worry about imports.
- Pseudocode credit is capped at 50%.
- Edge-cases are really important.

Specific topics?

charAt

Specific topics?

charAt

Specific topics?

Karel

charAt

comparing chars

Specific topics?

Karel

charAt

comparing chars

comparing Strings

Specific topics?

Karel

charAt

comparing chars

comparing Strings

Specific topics?

instance variables

Karel

charAt

comparing chars

comparing Strings

Specific topics?

instance variables

Karel

mouseListeners

charAt

comparing chars

integer division

comparing Strings

Specific topics?

instance variables

Karel

mouseListeners

modulus

charAt

comparing chars

integer division

comparing Strings

Specific topics?

Karel

instance variables

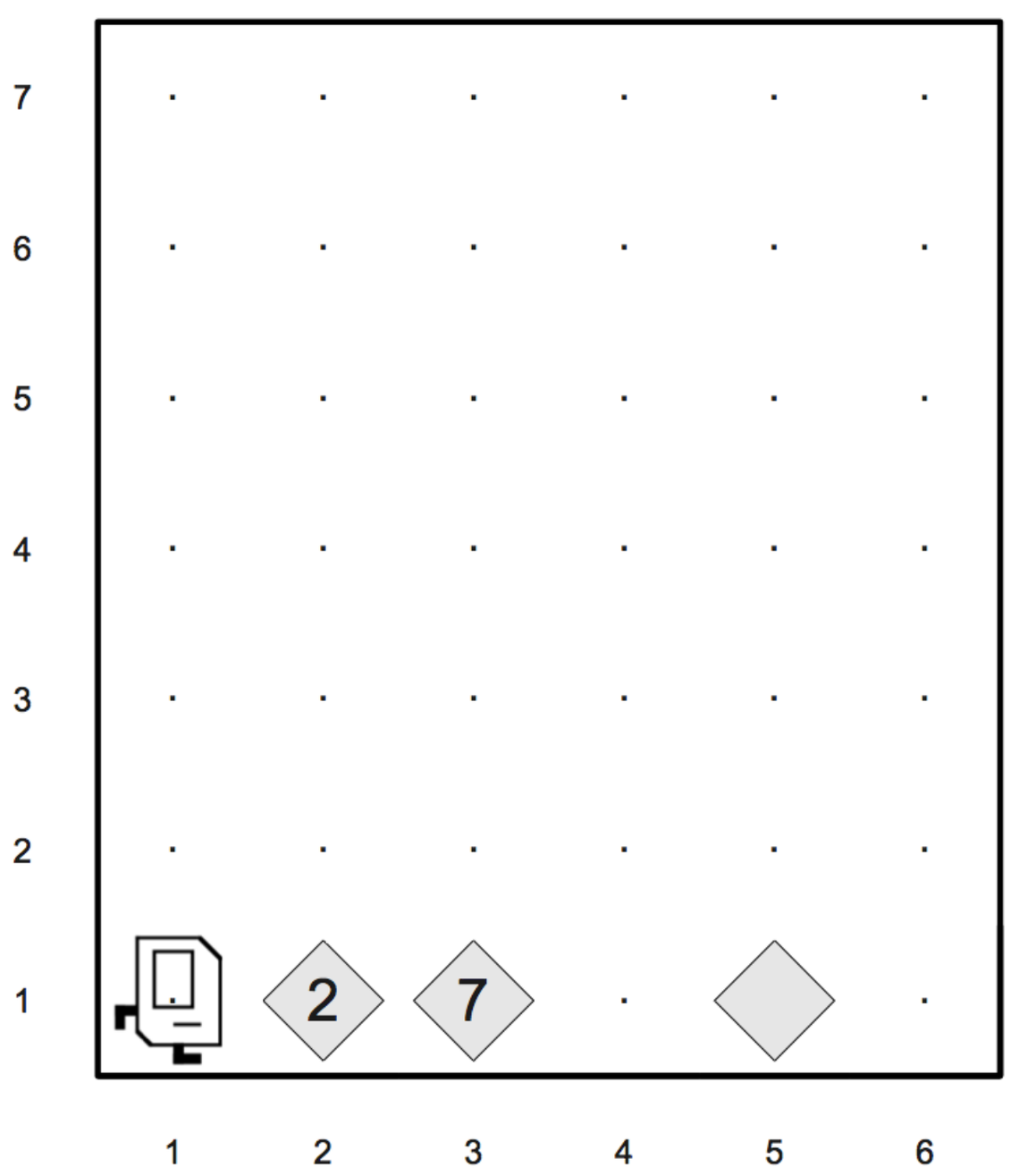
mouseListeners

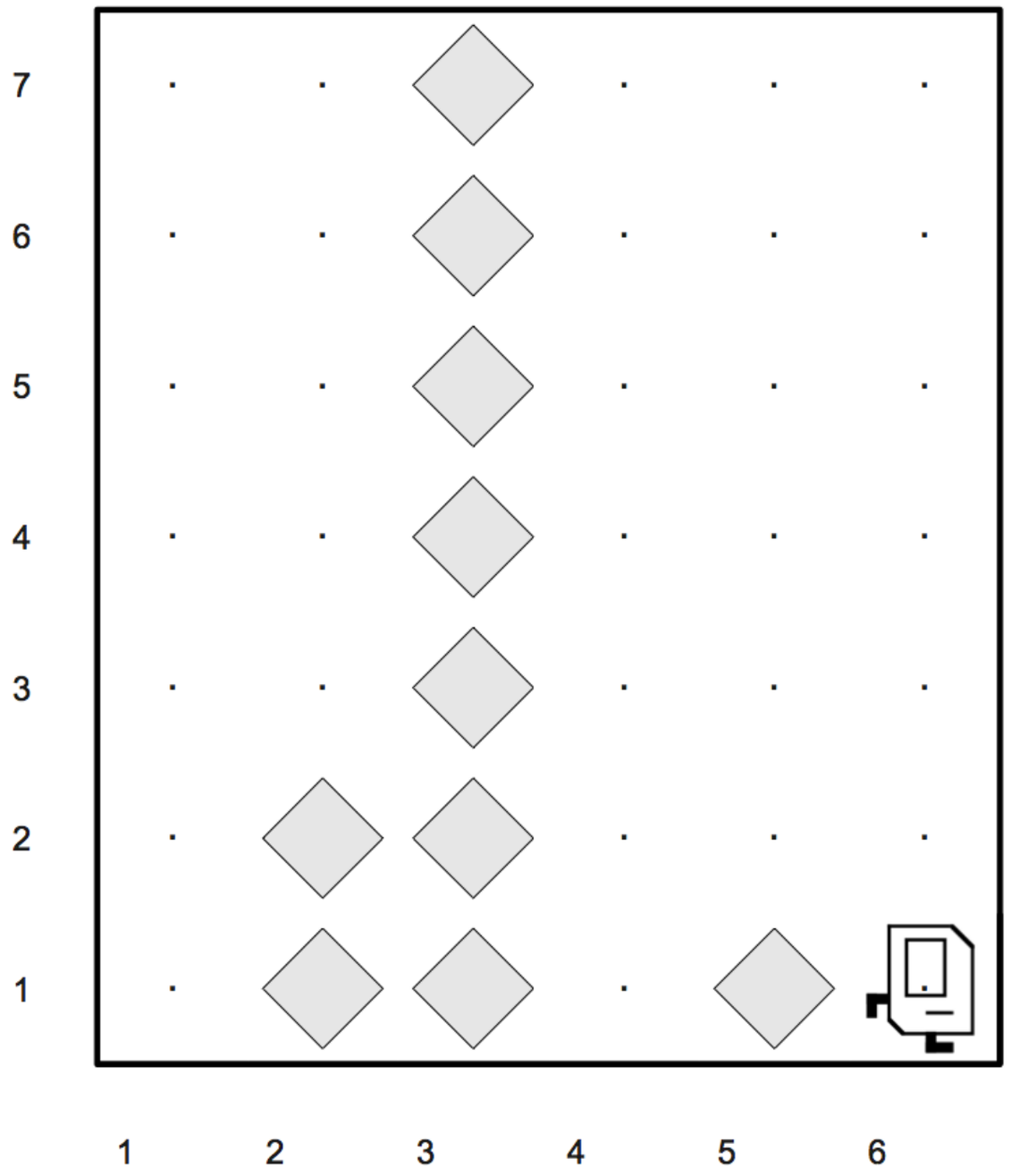
The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

Problem 1: Tower-Building Karel **(20 Points)**

In this problem, Karel begins on the corner of 1st Street and 1st Avenue, facing East. Scattered along 1st Street are piles of beepers representing the raw materials needed to assemble towers. Karel's job is to take the piles of beepers and unstack them to form towers of the appropriate heights. For example, if a pile contains four beepers, Karel would unstack the pile into a tower of four beepers. If the pile contains just one beeper, Karel would build a tower of height one consisting of just that beeper. Here is a sample run of the program:





In solving this problem, you can count on the following facts about the world:

- Karel starts off facing East at the corner of 1st Street and 1st Avenue with no beepers in its beeper bag.
- The world is tall enough to ensure that Karel has enough vertical space to build all the towers. However, some towers might be exactly the height of the world. In the above picture, for example, there is just enough room to build the tower of height seven.
- Each corner on 1st Street may contain a stack of beepers, including the very first and very last corners.

Karel's final location and heading do not matter, and you do not need to worry about efficiency. You are limited to the instructions in the Karel booklet; for example, the only variables allowed are loop control variables used within the control section of a **`for`** loop, and you must not use the **`break`** or **`return`** statements. You may find the method **`beepersInBag()`** useful, though your solution does not need to use this method.

```
import stanford.karel.*;

public class TowerBuildingKarel extends SuperKarel {
    public void run() {
        /* There would normally be a blank page here on which to write your answer. */
```

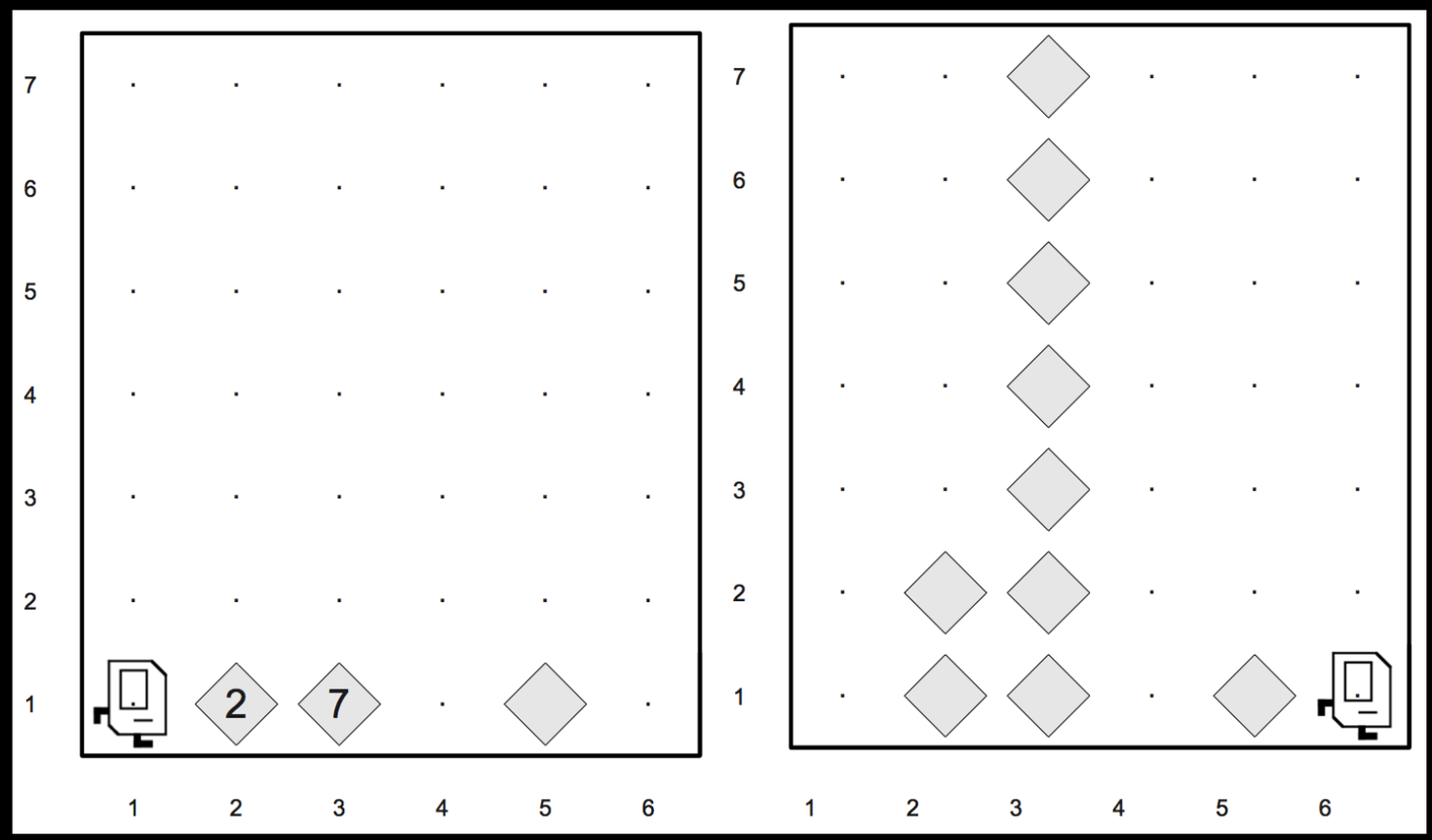
In solving this problem, you can count on the following facts about the world:

- Karel starts off facing East at the corner of 1st Street and 1st Avenue with no beepers in its beeper bag.
- The world is tall enough to ensure that Karel has enough vertical space to build all the towers. However, some towers might be exactly the height of the world. In the above picture, for example, there is just enough room to build the tower of height seven.
- Each corner on 1st Street may contain a stack of beepers, including the very first and very last corners.

Karel's final location and heading do not matter, and you do not need to worry about efficiency. You are limited to the instructions in the Karel booklet; for example, the only variables allowed are loop control variables used within the control section of a **`for`** loop, and you must not use the **`break`** or **`return`** statements. You may find the method **`beepersInBag()`** useful, though your solution does not need to use this method.

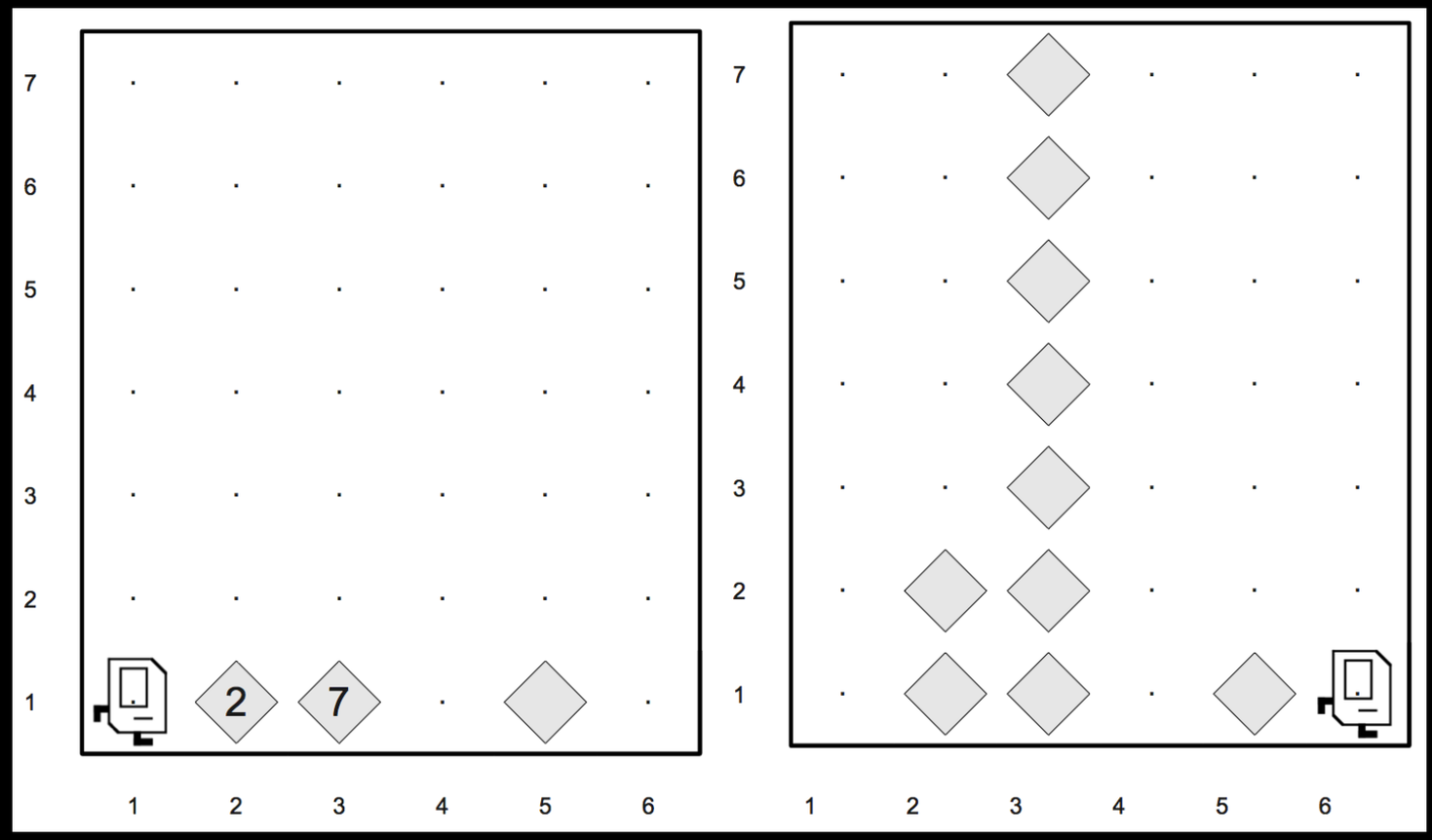
```
import stanford.karel.*;

public class TowerBuildingKarel extends SuperKarel {
    public void run() {
        /* There would normally be a blank page here on which to write your answer. */
```

```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {




    }
```

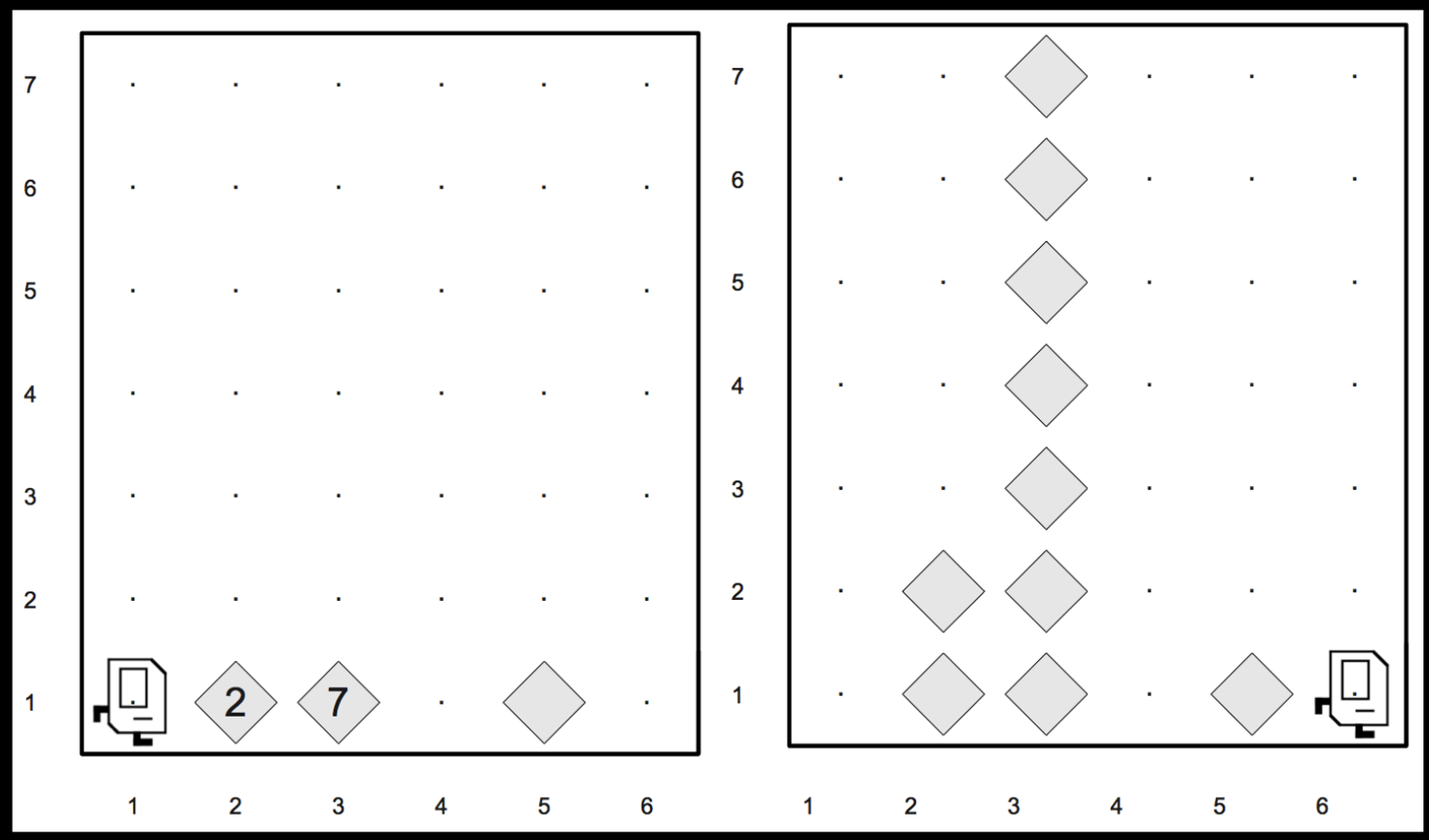



```
}
```

```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {
        while (frontIsClear()) {
            buildOneTower();
            move();
        }
        buildOneTower();
    }
}
```

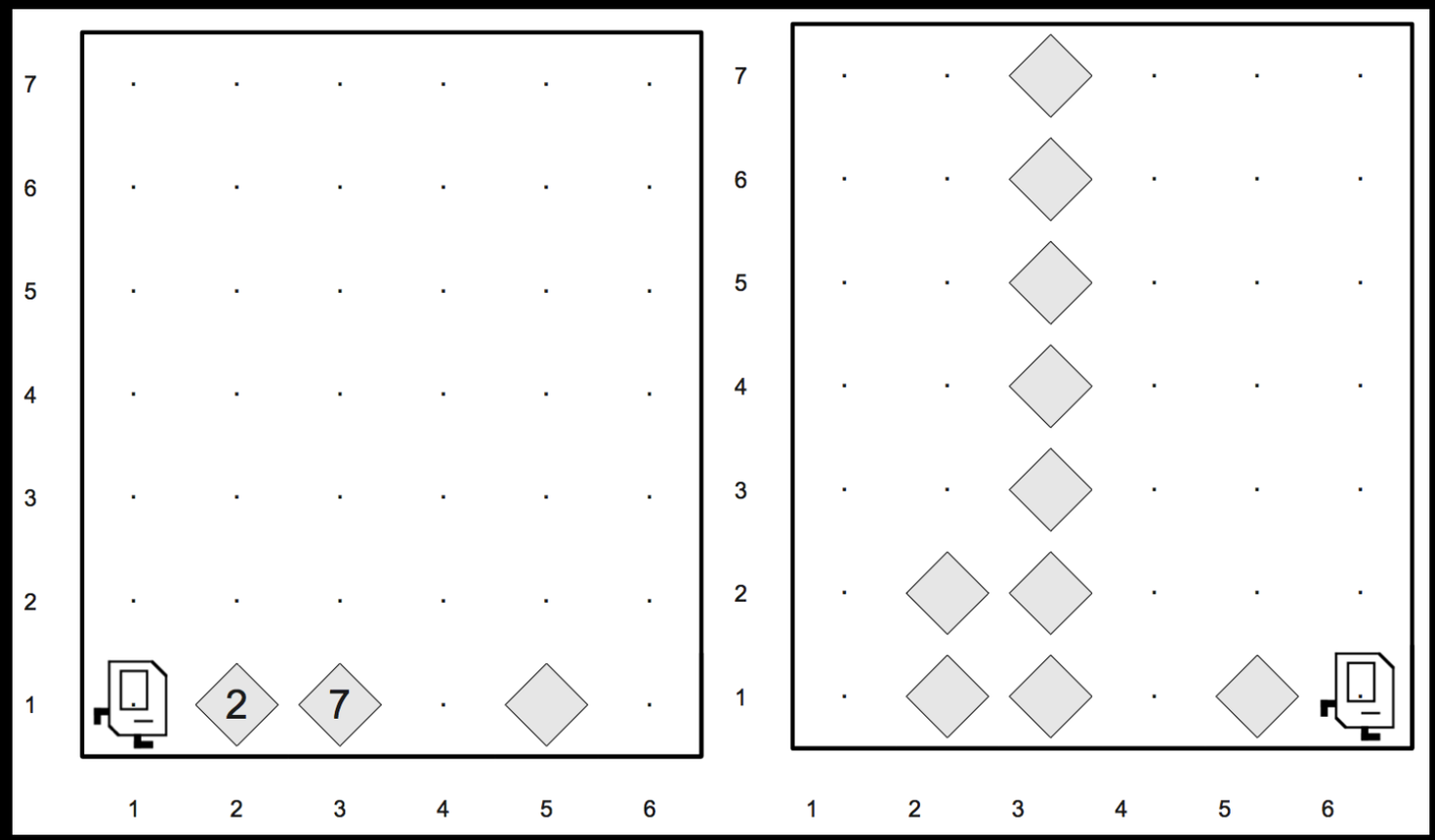


```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {
        while (frontIsClear()) {
            buildOneTower();
            move();
        }
        buildOneTower();
    }

    private void buildOneTower() {
        turnLeft();
        pickUpAllBeepers();
        placeBeepers();
        returnHome();
        turnLeft();
    }
```

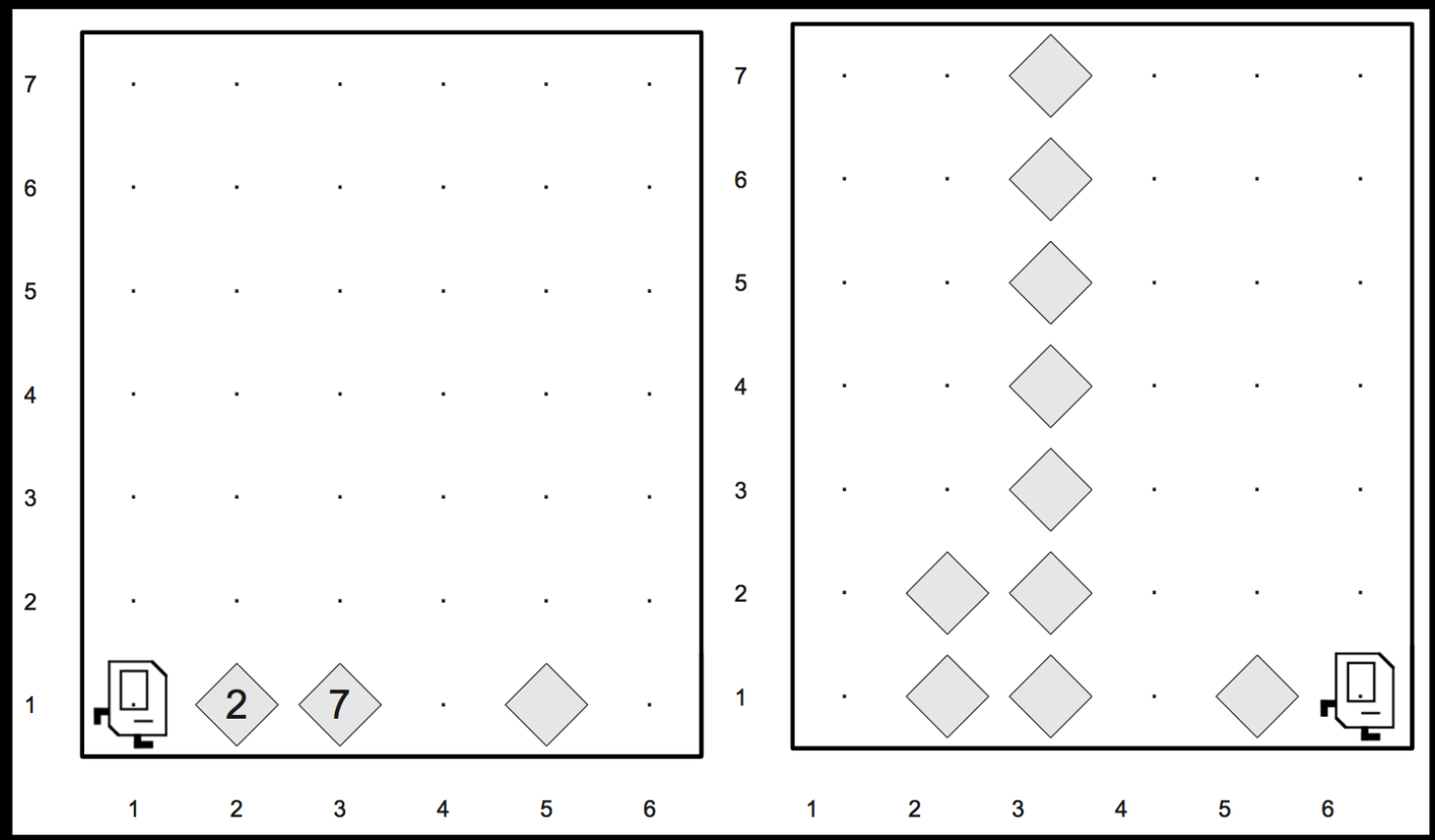



```
}
```

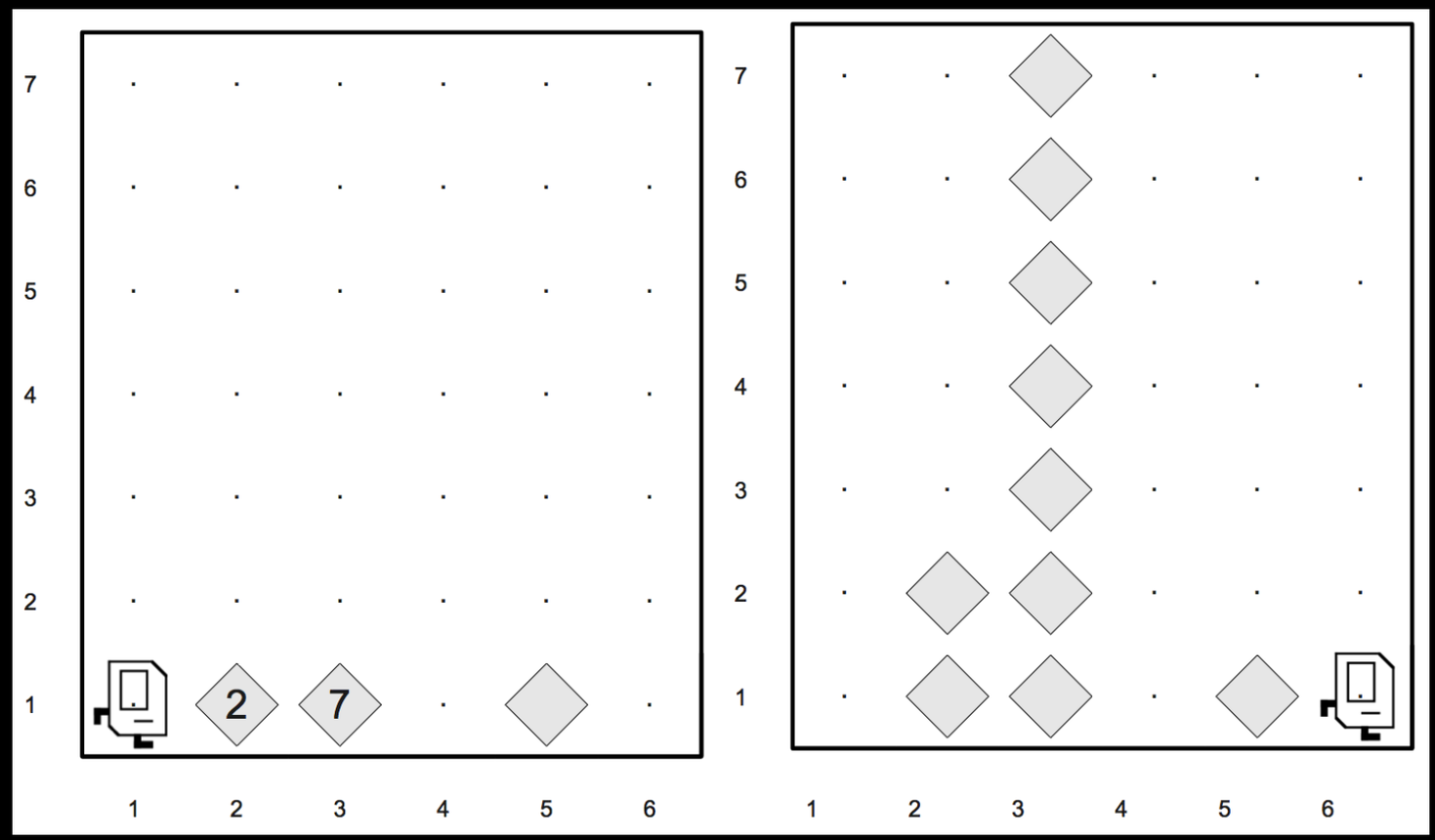
```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {
        while (frontIsClear()) {
            buildOneTower();
            move();
        }
        buildOneTower();
    }

    private void buildOneTower() {
        turnLeft();
        pickUpAllBeepers();
        placeBeepers();
        returnHome();
        turnLeft();
    }

    private void pickUpAllBeepers() {
        while (beepersPresent()) {
            pickBeeper();
        }
    }
}
```



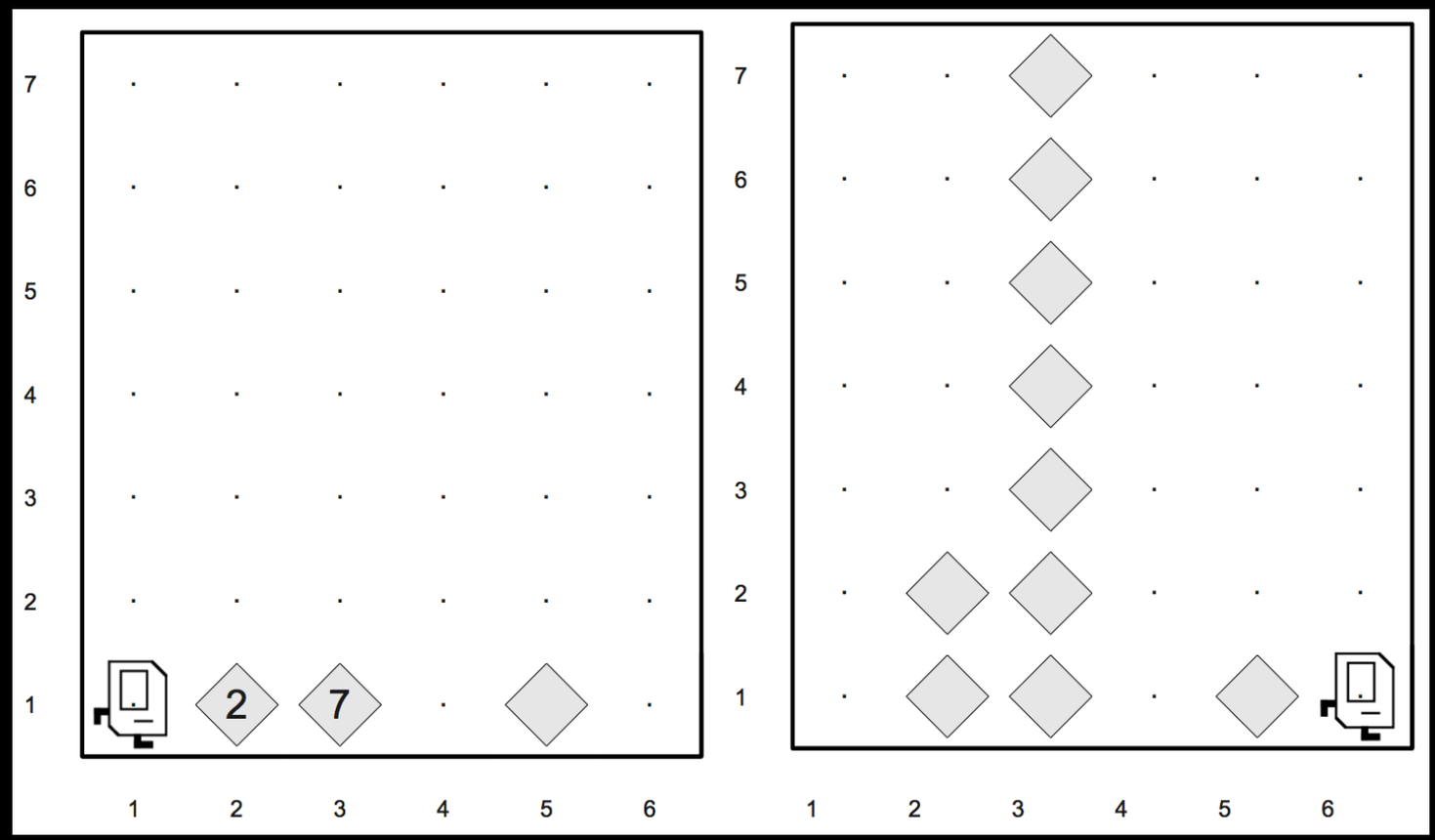
```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {
        while (frontIsClear()) {
            buildOneTower();
            move();
        }
        buildOneTower();
    }

    private void buildOneTower() {
        turnLeft();
        pickUpAllBeepers();
        placeBeepers();
        returnHome();
        turnLeft();
    }

    private void pickUpAllBeepers() {
        while (beepersPresent()) {
            pickBeeper();
        }
    }

    private void placeBeepers() {
        while (beepersInBag()) {
            putBeeper();
            if (frontIsClear()) {
                move();
            }
        }
    }




}
```



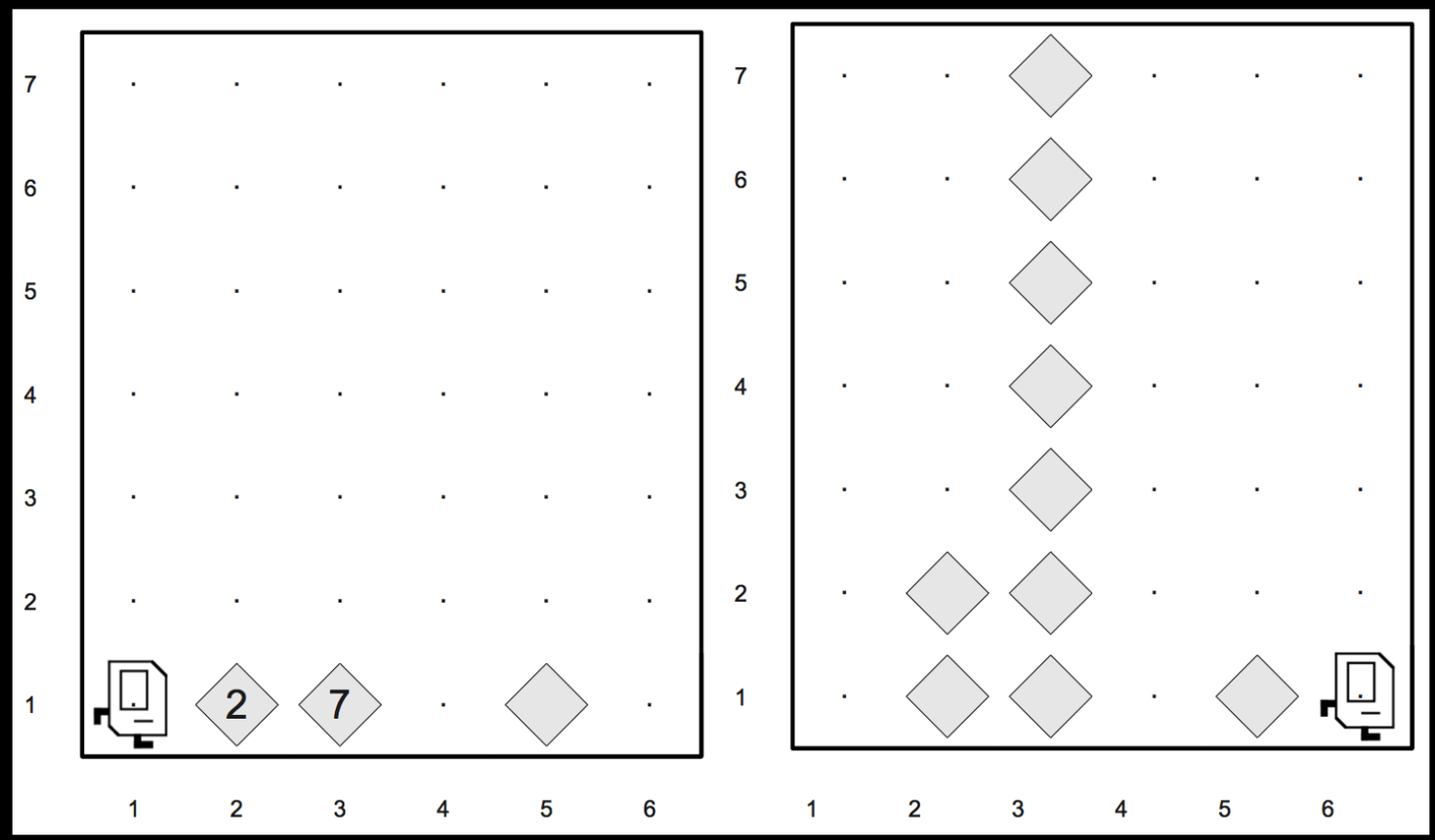
```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {
        while (frontIsClear()) {
            buildOneTower();
            move();
        }
        buildOneTower();
    }

    private void buildOneTower() {
        turnLeft();
        pickUpAllBeepers();
        placeBeepers();
        returnHome();
        turnLeft();
    }

    private void pickUpAllBeepers() {
        while (beepersPresent()) {
            pickBeeper();
        }
    }

    private void placeBeepers() {
        while (beepersInBag()) {
            putBeeper();
            if (frontIsClear()) {
                move();
            }
        }
    }
}
```



```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {
        while (frontIsClear()) {
            buildOneTower();
            move();
        }
        buildOneTower();
    }

    private void buildOneTower() {
        turnLeft();
        pickUpAllBeepers();
        placeBeepers();
        returnHome();
        turnLeft();
    }

    private void pickUpAllBeepers() {
        while (beepersPresent()) {
            pickBeeper();
        }
    }

    private void placeBeepers() {
        while (beepersInBag()) {
            putBeeper();
            if (frontIsClear()) {
                move();
            }
        }
    }




}
```



```
import stanford.karel.*;

public class TowerBuildingKarel extends
SuperKarel {
    public void run() {
        while (frontIsClear()) {
            buildOneTower();
            move();
        }
        buildOneTower();
    }

    private void buildOneTower() {
        turnLeft();
        pickUpAllBeepers();
        placeBeepers();
        returnHome();
        turnLeft();
    }

    private void pickUpAllBeepers() {
        while (beepersPresent()) {
            pickBeeper();
        }
    }

    private void placeBeepers() {
        while (beepersInBag()) {
            putBeeper();
            if (frontIsClear()) {
                move();
            }
        }
    }

    private void returnHome() {
        turnAround();
        while (frontIsClear()) {
            move();
        }
    }
}
```



The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

Problem 2: Simple Java expressions, statements, and methods (20 points)

(2a) Compute the value of each of the following Java expressions. If an error occurs during any of these evaluations, write "Error" on that line and explain briefly why the error occurs.

```
5.0 / 4 - 4 / 5
```

_____

```
7 < 9 - 5 && 3 % 0 == 3
```

_____

```
"B" + 8 + 4
```

_____

Expressions

- Order of operations
- Short circuit evaluation
- Undefined operations
  - divide by zero
  - subtract from a String
  - etc.

Operator Precedence

Operators	Precedence
postfix	*expr*++ *expr*--
unary	++*expr* --*expr* +*expr* -*expr* ~ !
multiplicative	* / %
additive	+ -
shift	<< >> >>>
relational	< > <= >= instanceof
equality	== !=
bitwise AND	&
bitwise exclusive OR	^
bitwise inclusive OR	|
logical AND	&&
logical OR	||
ternary	? :
assignment	= += -= *= /= %= &= ^= |= <<= >>= >>>=

Operators of equal precedence are evaluated left to right

Short circuit evaluation

- **`x || y`** means **`x OR y`**
  - what if **`x`** is **`true`**?
  - **`y`** doesn't matter!
- **`x && y`** means **`x AND y`**
  - what if **`x`** is **`false`**?
  - **`x`** doesn't matter!
- This is important because it can mean error code won't run
  - **`3 == 3 || 4 / 0`**

Expression evaluation

Expression evaluation

- `“E” - “A”`

Expression evaluation

- `“E” - “A”`

`Error: Strings don’t subtract`

Expression evaluation

- `“E” - “A”` `Error: Strings don’t subtract`
- `1 – 2 – 3 - 4.0`

Expression evaluation

- `"E" - "A"` `Error: Strings don't subtract`
- `1 — 2 — 3 - 4.0` `-8.0`

Expression evaluation

- `"E" - "A"` `Error: Strings don't subtract`
- `1 — 2 — 3 - 4.0` `-8.0`
- `(13 / 7) / (7 / 13)`

Expression evaluation

- `"E" - "A"` `Error: Strings don't subtract`
- `1 — 2 — 3 - 4.0` `-8.0`
- `(13 / 7) / (7 / 13)` `Error: Divide by zero`

Expression evaluation

- “E” - “A” Error: Strings don’t subtract
- 1 – 2 – 3 - 4.0 -8.0
- (13 / 7) / (7 / 13) Error: Divide by zero
- "1" + (1 + 1) + 1 + 1 + "1"

Expression evaluation

- `"E" - "A"` `Error: Strings don't subtract`
- `1 — 2 — 3 - 4.0` `-8.0`
- `(13 / 7) / (7 / 13)` `Error: Divide by zero`
- `"1" + (1 + 1) + 1 + 1 + "1"` `"12111"`

Expression evaluation

- `"E" - "A"` `Error: Strings don't subtract`
- `1 — 2 — 3 - 4.0` `-8.0`
- `(13 / 7) / (7 / 13)` `Error: Divide by zero`
- `"1" + (1 + 1) + 1 + 1 + "1"` `"12111"`
- `1 == 2 || 2 / 0 == 3`

Expression evaluation

- `"E" - "A"` Error: Strings don't subtract
- `1 – 2 – 3 - 4.0` -8.0
- `(13 / 7) / (7 / 13)` Error: Divide by zero
- `"1" + (1 + 1) + 1 + 1 + "1"` "12111"
- `1 == 2 || 2 / 0 == 3` Error: Divide by zero

More expressions

More expressions

- `4 / 7 * (double)7 / 4`

More expressions

- `4 / 7 * (double)7 / 4` `0.0`

More expressions

- `4 / 7 * (double)7 / 4` `0.0`
- `137 / 42 == 0 && 137 / 0 == 42`

More expressions

- `4 / 7 * (double)7 / 4` `0.0`
- `137 / 42 == 0 && 137 / 0 == 42` `false`

More expressions

- `4 / 7 * (double)7 / 4` `0.0`
- `137 / 42 == 0 && 137 / 0 == 42` `false`
- `1 + 2 + "1 + 2" + 1 + 2`

More expressions

- `4 / 7 * (double)7 / 4` `0.0`
- `137 / 42 == 0 && 137 / 0 == 42` `false`
- `1 + 2 + "1 + 2" + 1 + 2`

More expressions

- `4 / 7 * (double)7 / 4` 0.0
- `137 / 42 == 0 && 137 / 0 == 42` false
- `1 + 2 + "1 + 2" + 1 + 2`

More expressions

- `4 / 7 * (double)7 / 4` 0.0
- `137 / 42 == 0 && 137 / 0 == 42` false
- `1 + 2 + "1 + 2" + 1 + 2`

More expressions

- `4 / 7 * (double)7 / 4` 0.0
- `137 / 42 == 0 && 137 / 0 == 42` false
- `1 + 2 + "1 + 2" + 1 + 2`

More expressions

- `4 / 7 * (double)7 / 4` `0.0`
- `137 / 42 == 0 && 137 / 0 == 42` `false`
- `1 + 2 + "1 + 2" + 1 + 2` `"31 + 212"`

More expressions

- `4 / 7 * (double)7 / 4` `0.0`
- `137 / 42 == 0 && 137 / 0 == 42` `false`
- `1 + 2 + "1 + 2" + 1 + 2` `"31 + 212"`
- `(char)('3' – '0' + 'A')`

More expressions

- `4 / 7 * (double)7 / 4` 0.0
- `137 / 42 == 0 && 137 / 0 == 42` false
- `1 + 2 + "1 + 2" + 1 + 2` "31 + 212"
- `(char)('3' – '0' + 'A')` ‘D’

More expressions

- `4 / 7 * (double)7 / 4` 0.0
- `137 / 42 == 0 && 137 / 0 == 42` false
- `1 + 2 + "1 + 2" + 1 + 2` "31 + 212"
- `(char)('3' – '0' + 'A')` ‘D’

A B C D

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

(2b) What output is printed by the following program:

```
/*
 * File: Problem2b.java
 * --------------------
 * This program doesn't do anything useful and exists only to test
 * your understanding of method calls and parameter passing.
 */

import acm.program.*;

public class Problem2b extends ConsoleProgram {

  public void run() {
     int num1 = 2;
     int num2 = 13;
     println("The 1st number is: " + Mystery(num1, 6));
     println("The 2nd number is: " + Mystery(num2 % 5, 1 + num1 * 2));
  }


  private int Mystery(int num1, int num2) {
     num1 = Unknown(num1, num2);
     num2 = Unknown(num2, num1);
     return(num2);
  }


  private int Unknown(int num1, int num2) {
     int num3 = num1 + num2;
     num2 += num3 * 2;
     return(num2);
  }

}
```

Answer:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
	public void run() {
		int tinker  = 36;
		int tailor  = 54;
		int soldier = smiley(tailor, tinker);
		println("soldier = " + soldier);
	}

	private int smiley(int tinker, int tailor) {
		int poorMan = guillam(tinker, tailor);
		println("poorMan = " + poorMan);

		int beggarMan = guillam(tailor + 9, tinker / 9);
		println("beggarMan = " + beggarMan);

		return poorMan + beggarMan;
	}

	private int guillam(int karla, int mundt) {
		karla %= 10;
		mundt /= 10;
		return 100 * karla + mundt;
	}
}
```

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

smiley

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker:
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker:
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor:
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor:
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla:
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **54**
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **54**
mundt: **36**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **36**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **36**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan:
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **4**
mundt: **3**

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **45**
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **45**
mundt: **6**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **6**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **6**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan:

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla: **5**
mundt: **0**

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier:

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker: **54**
tailor: **36**
poorMan: **403**
beggarMan: **500**

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500

```
import acm.program.*;
public class JavaLeCarre extends ConsoleProgram {
      public void run() {
            int tinker  = 36;
            int tailor  = 54;
            int soldier = smiley(tailor, tinker);
            println("soldier = " + soldier);
      }

      private int smiley(int tinker, int tailor) {
            int poorMan = guillam(tinker, tailor);
            println("poorMan = " + poorMan);

            int beggarMan = guillam(tailor + 9, tinker / 9);
            println("beggarMan = " + beggarMan);

            return poorMan + beggarMan;
      }

      private int guillam(int karla, int mundt) {
            karla %= 10;
            mundt /= 10;
            return 100 * karla + mundt;
      }
}
```

run

tinker: **36**
tailor: **54**
soldier: **903**

smiley

tinker:
tailor:
poorMan:
beggarMan:

guillam

karla:
mundt:

poorMan = 403
beggarMan = 500
soldier = 903

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

Problem Three: The Saint Petersburg Game **(25 Points)**

The *Saint Petersburg Game* or *Saint Petersburg Lottery* is a hypothetical casino game played by two players (say, you and me) sitting at a table. I begin by putting $1 on the table, and you then repeatedly flip a coin until it comes up tails. Each time the coin comes up heads, I double the amount of money on the table. As soon as the coin comes up tails, the game is over and you win all the money on the table (no strings attached – I'm just feeling really generous!)

One sample run of the game is as follows. I put $1 on the table. You then flip tails, so the game ends and you collect $1. Another run might go like this: I put $1 on the table. You flip heads, so I double the money to $2. You flip heads again, so I double the money to $4. You flip heads again, so I double the money to $8. You then flip tails, so the game ends and you collect $8.

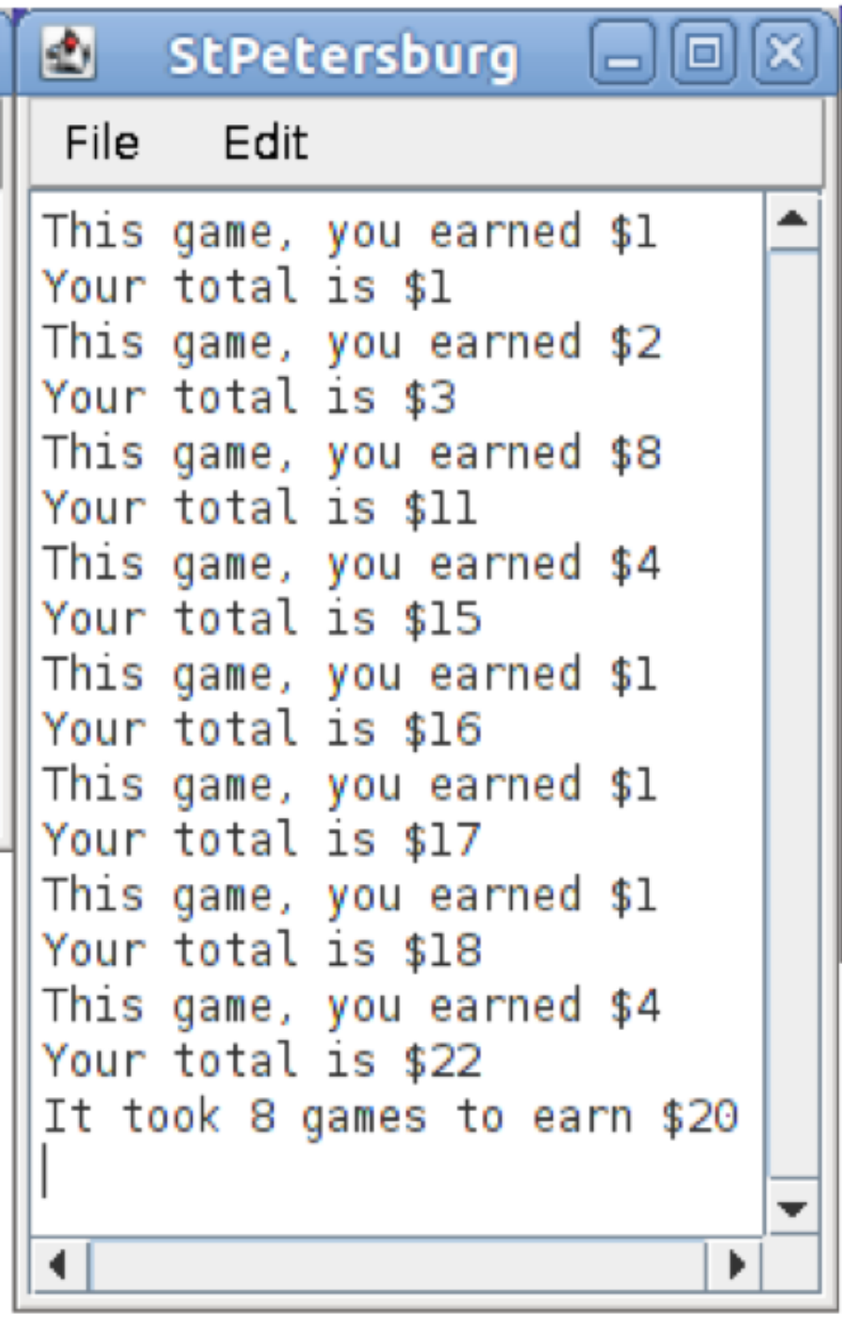
Write a program that plays the Saint Petersburg Game as many times as is necessary to earn a total of at least $20. To clarify – you aren't playing until you win $20 in a single game; instead, you're playing until your total winnings across all games you've played is $20. After each game, your program should print out how much money you won during that game, along with your total winnings. For example, if after the first game you earned $4, you would print out

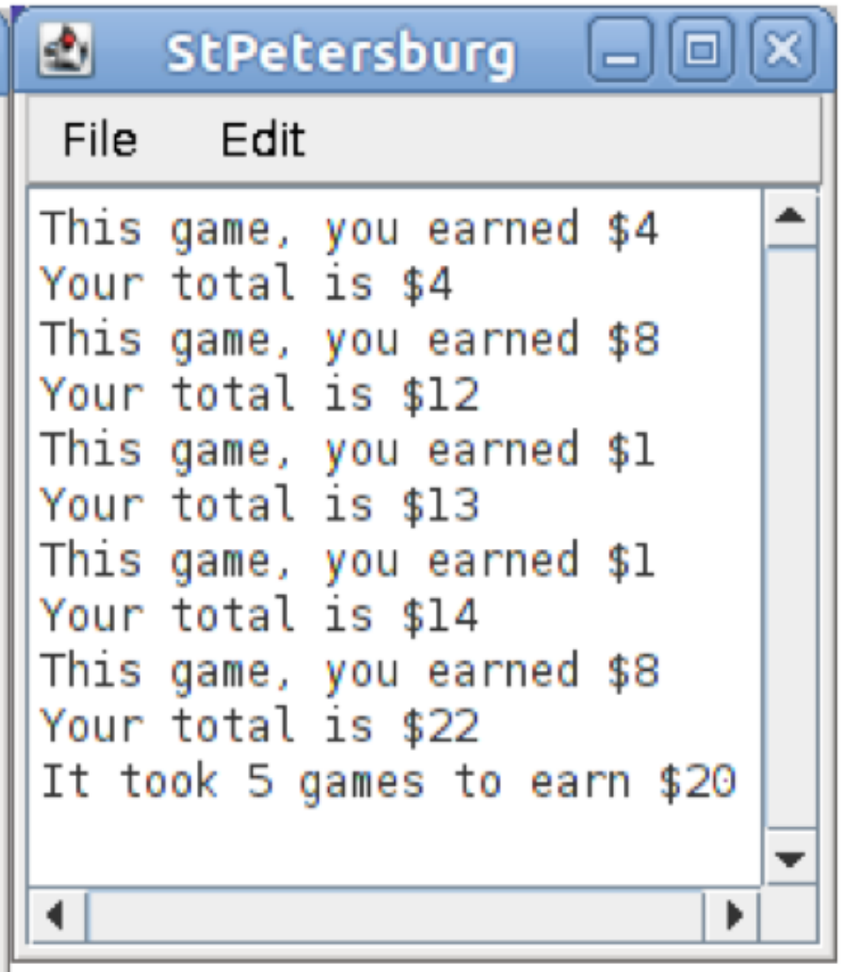
```
This game, you earned $4
Your total is $4
```

Once you've earned at least $20, your program should output how many times you had to play the Saint Petersburg Game to earn $20. For example, if it takes five games to win, at the end you'd print

```
It took 5 games to earn $20
```

Three sample runs of the program are shown below:






```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    /* Again, we would normally provide space for you to write your solution */
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {

    }

}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
```

```
    }

}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;
```

```
        }

}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;
```

```
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }
}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {




        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }




}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {




        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }


}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {




        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }




}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {



            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }



}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();


            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {









    }
}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();
            println("Your total is $" + totalWinnings);

            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {









    }
}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();
            println("Your total is $" + totalWinnings);

            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {

        int winnings = 1;









    }
}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();
            println("Your total is $" + totalWinnings);

            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {

        int winnings = 1;









        return winnings;
    }
}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();
            println("Your total is $" + totalWinnings);

            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {

        int winnings = 1;









        println("This game, you won $" + roundWinnings);
        return winnings;
    }
}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();
            println("Your total is $" + totalWinnings);

            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {
        RandomGenerator rgen = RandomGenerator.getInstance();
        int winnings = 1;









        println("This game, you won $" + roundWinnings);
        return winnings;
    }
}
```

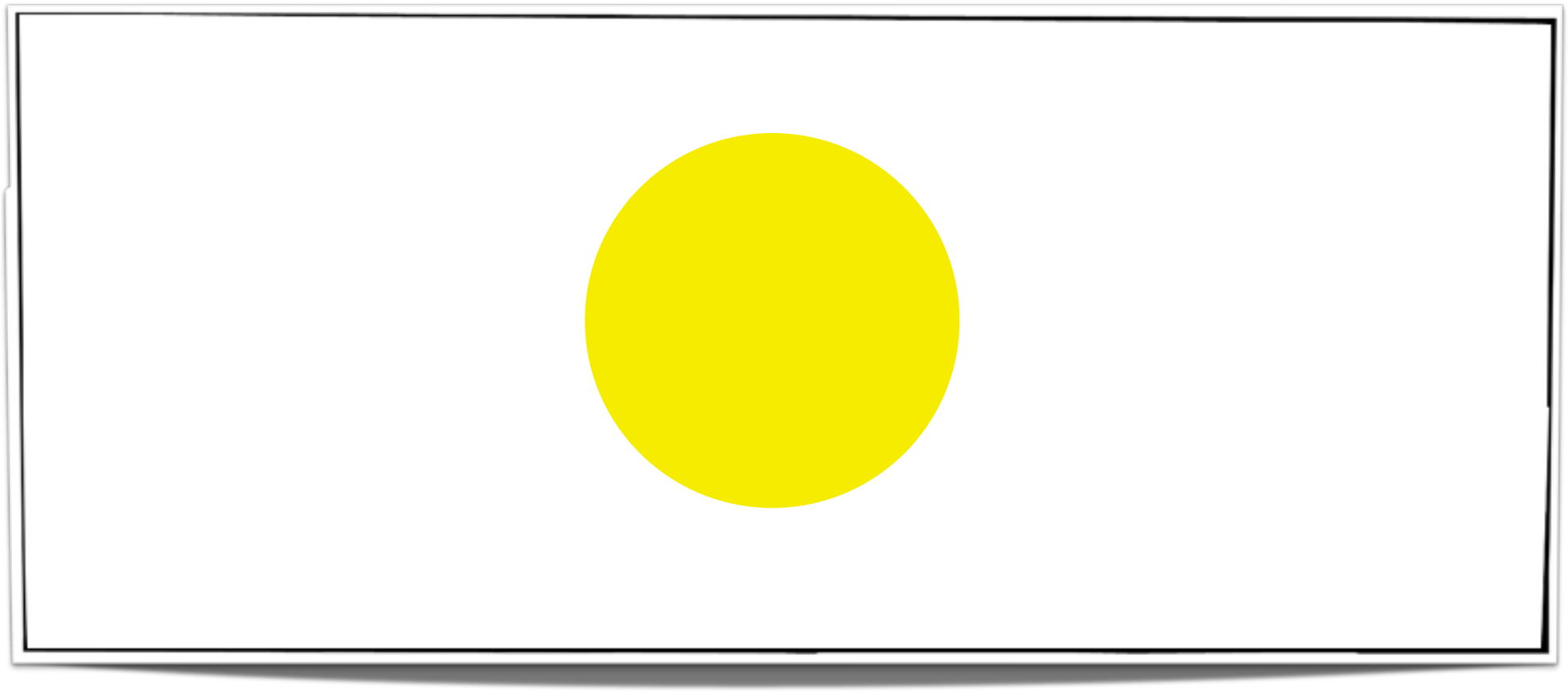
```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();
            println("Your total is $" + totalWinnings);

            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {
        RandomGenerator rgen = RandomGenerator.getInstance();
        int winnings = 1;

        while (rgen.nextBoolean()) {

        }

        println("This game, you won $" + roundWinnings);
        return winnings;
    }
}
```

```
import acm.program.*;
import acm.util.*;

public class SaintPetersburgGame extends ConsoleProgram {
    private static final int TARGET_AMOUNT = 20;

    public void run() {
        int totalWinnings = 0;
        int numRounds = 0;

        while (totalWinnings < TARGET_AMOUNT) {
            totalWinnings += playGame();
            println("Your total is $" + totalWinnings);

            numRounds++;
        }
        println("It took " + numRounds + " games to earn $" + TARGET_AMOUNT);
    }

    private int playGame() {
        RandomGenerator rgen = RandomGenerator.getInstance();
        int winnings = 1;

        while (rgen.nextBoolean()) {
            winnings *= 2;
        }

        println("This game, you won $" + roundWinnings);
        return winnings;
    }
}
```

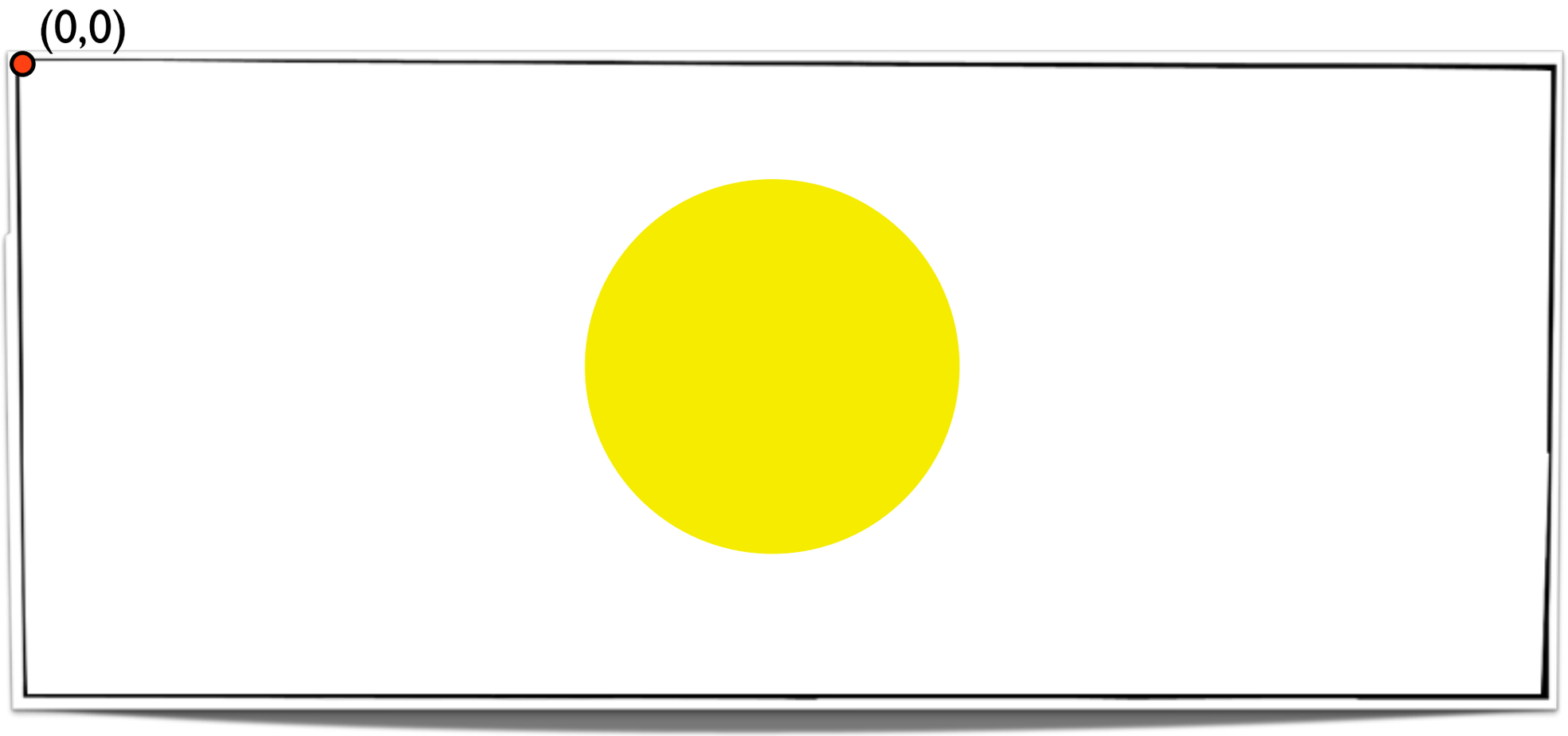
The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

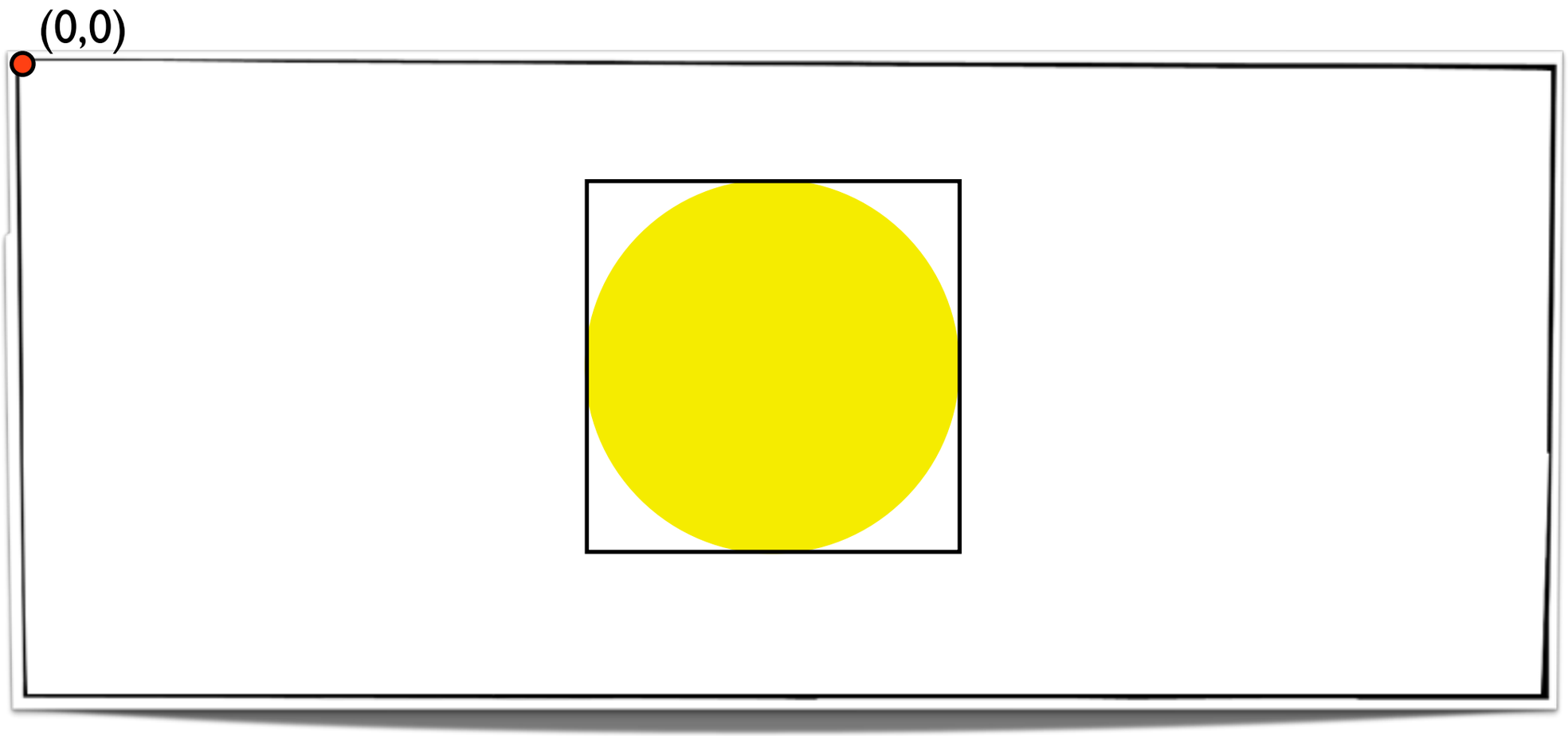
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



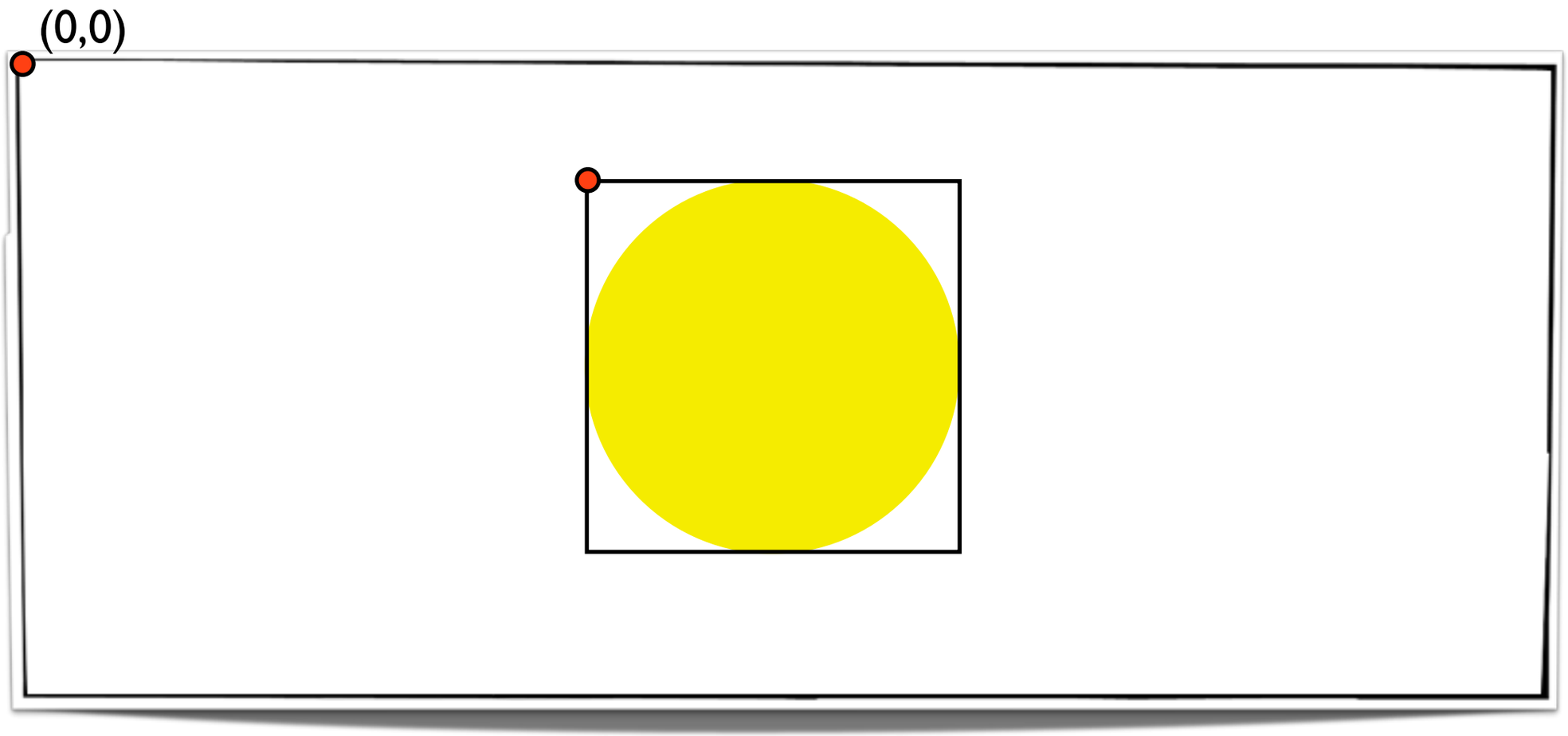
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



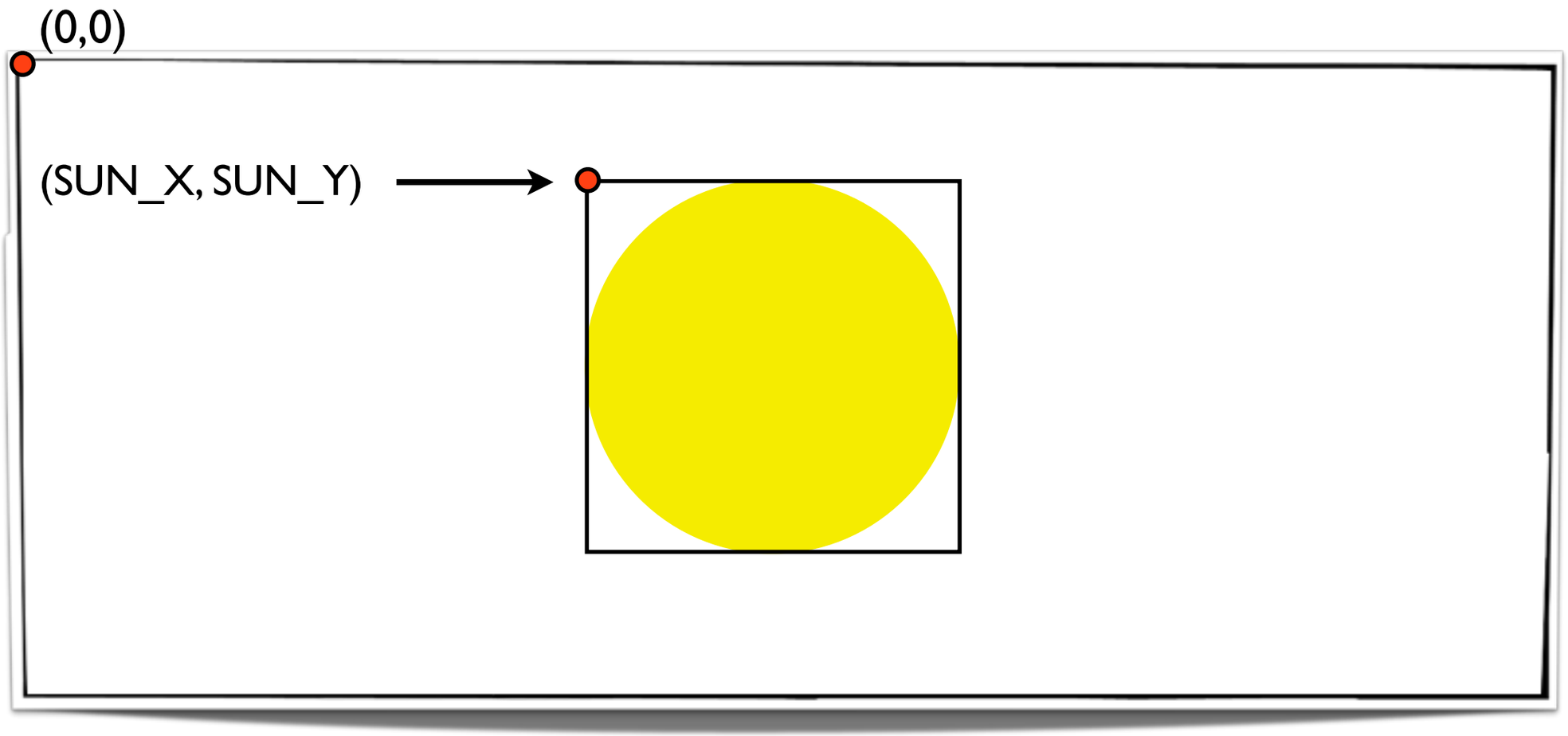
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



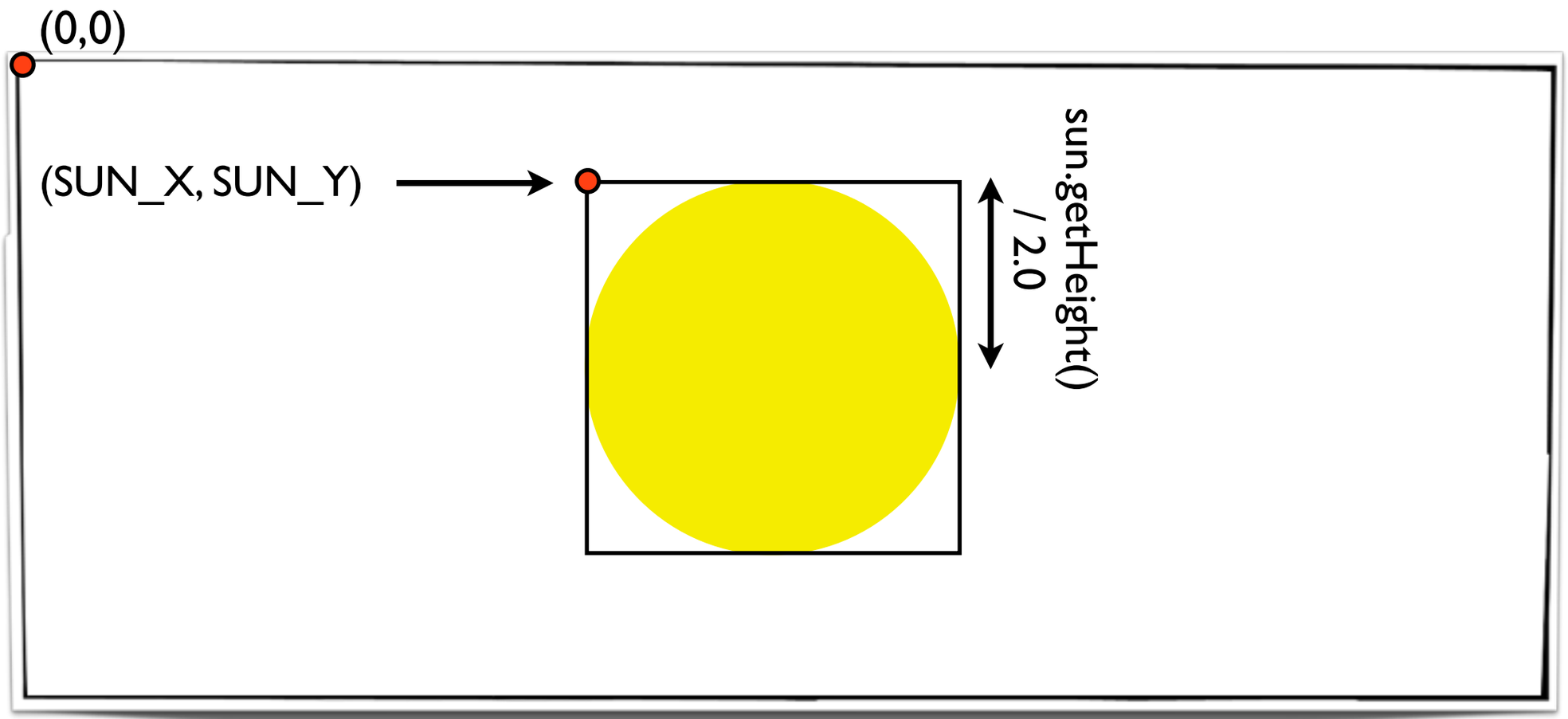
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



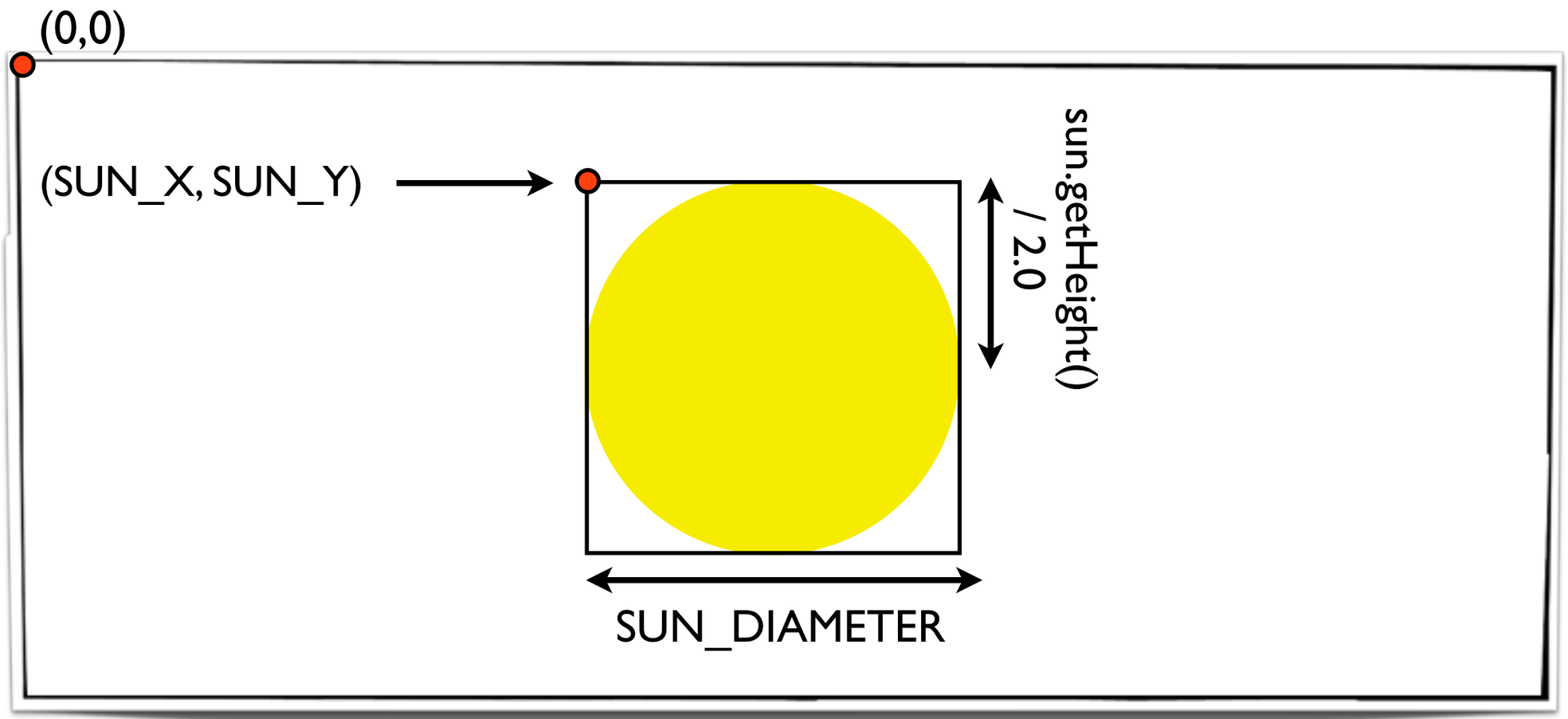
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



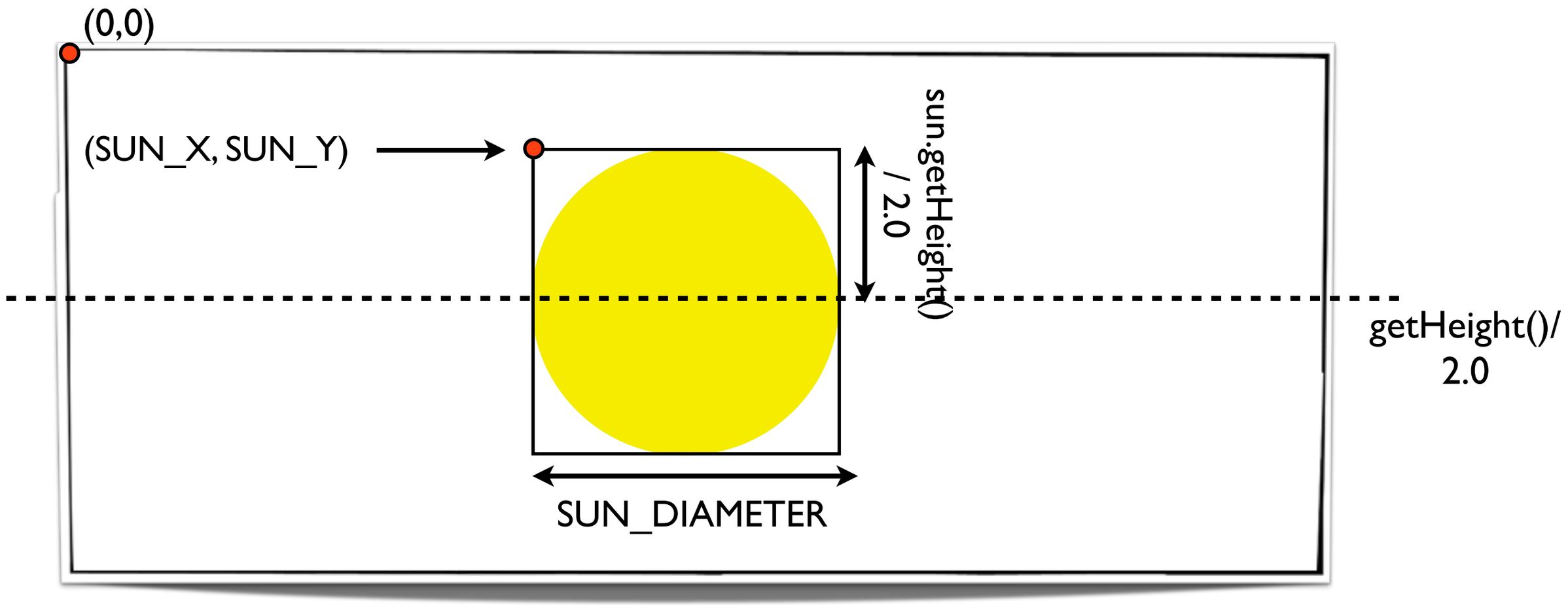
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



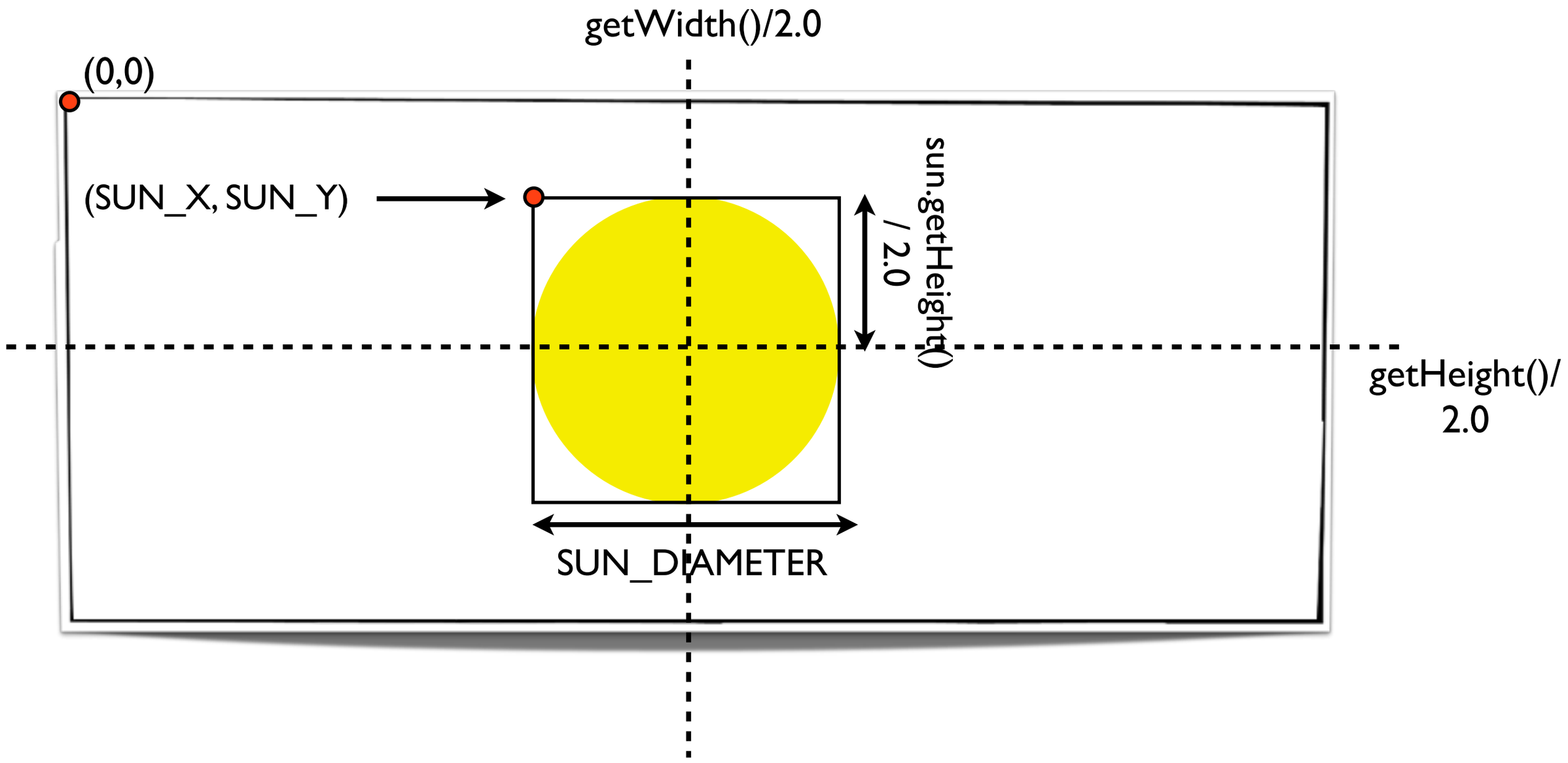
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



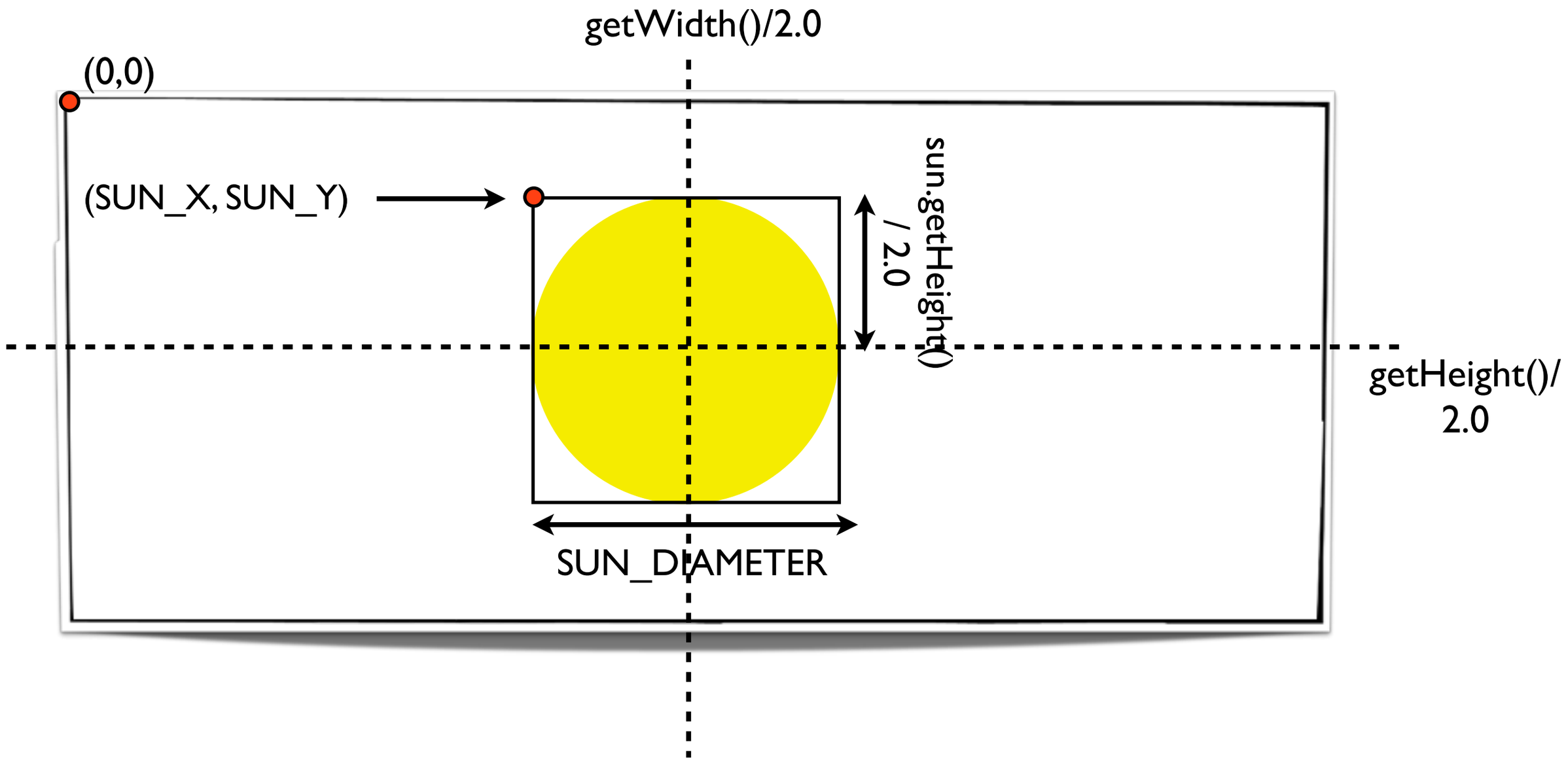
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



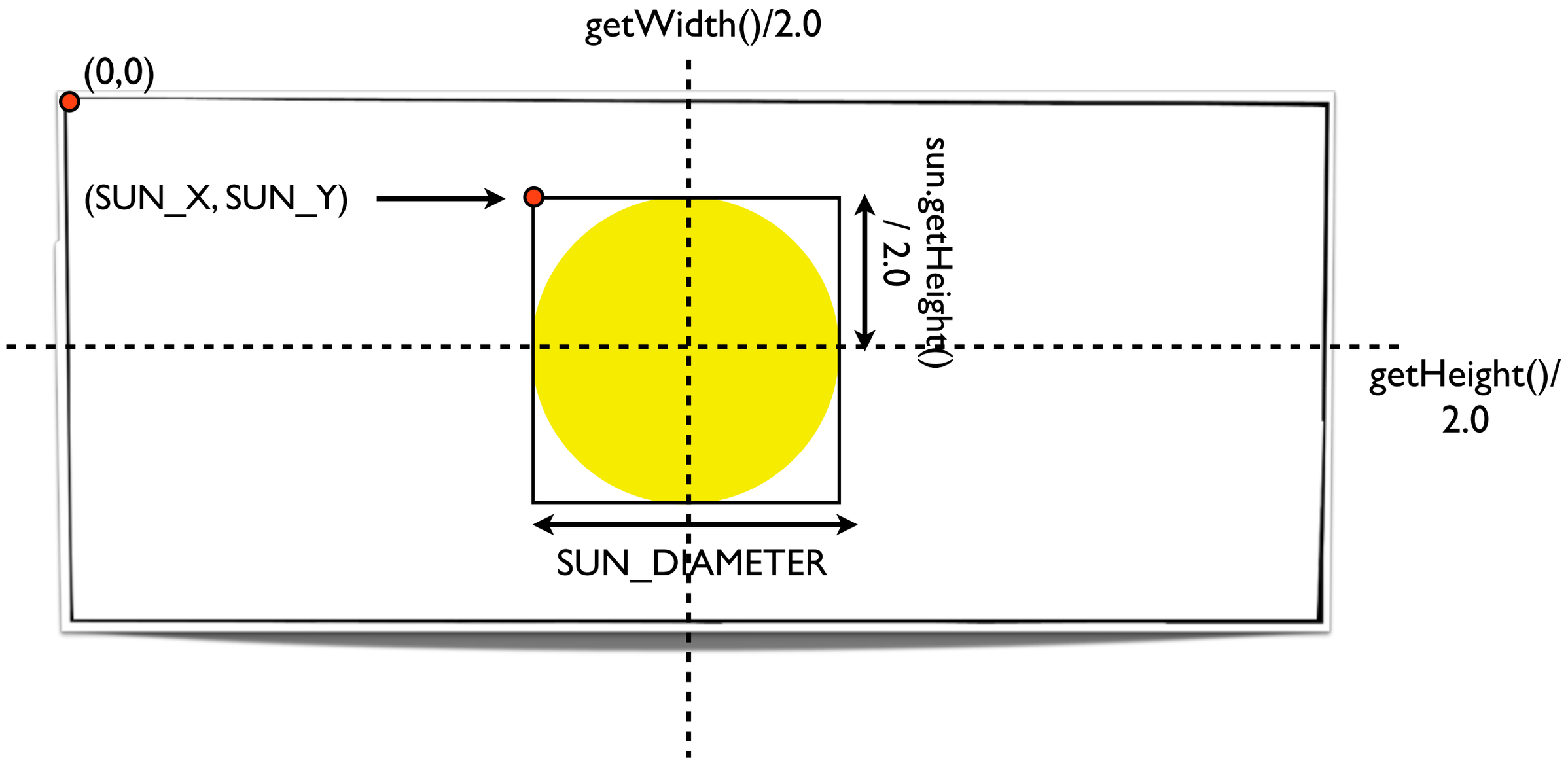
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________;
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________; getWidth()/2.0 - sun.getWidth()/2.0
  double sunY = _________;
  add(sun, sunX, sunY);
}
```



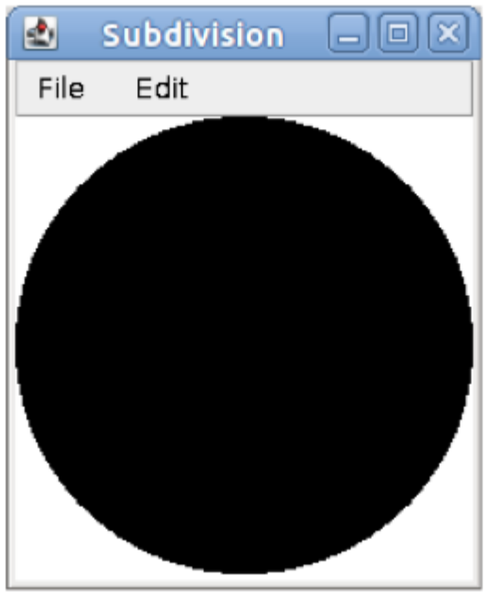
```
private void drawSun() {
  GOval sun = new GOval(SUN_DIAMETER, SUN_DIAMETER);
  sun.setColor(Color.YELLOW);
  sun.setFilled(true);
  sun.setFillColor(Color.YELLOW);
  double sunX = _________; getWidth()/2.0 - sun.getWidth()/2.0
  double sunY = _________; getHeight()/2.0 - sun.getHeight()/2.0
  add(sun, sunX, sunY);
}
```



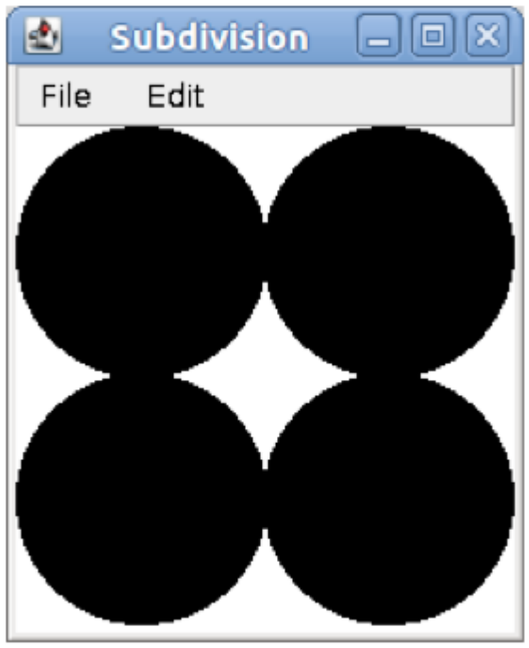
Problem Four: Subdivision **(25 Points)**

In this problem, you'll write a program that lets the user continuously cut an circle into smaller and smaller pieces, leading to aesthetically pleasing pictures.

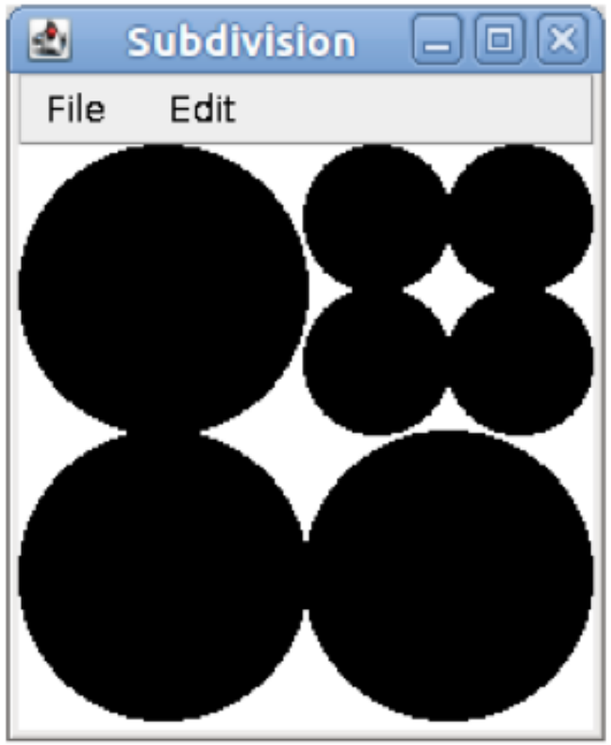
When the program starts up, the user sees a black circle that occupies the entire window, like this:



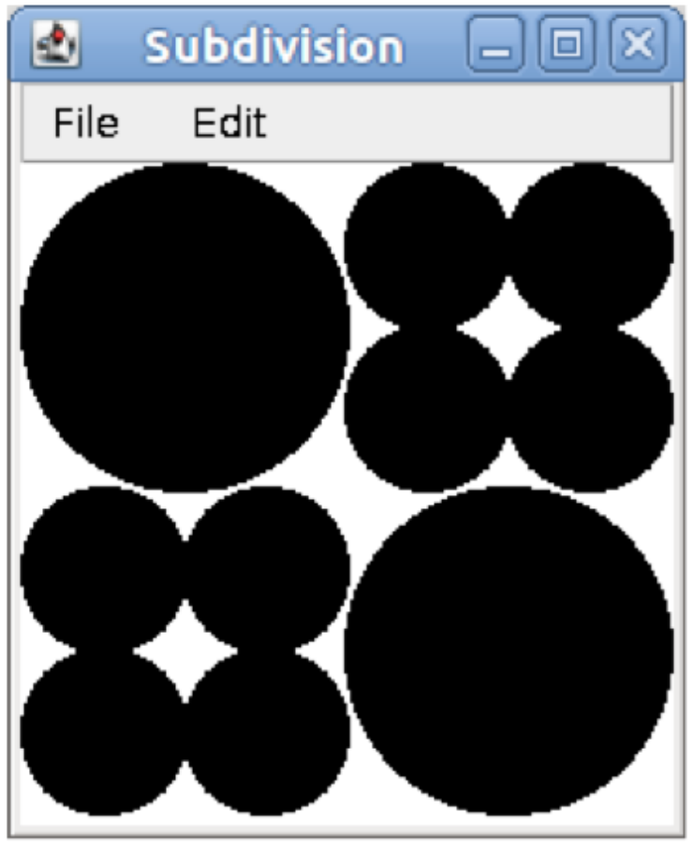
Now, if the user clicks on the circle, that circle is replaced by four smaller circles, each of which is a quarter of the size of the original circle. For example, after clicking on the circle above, the program would display



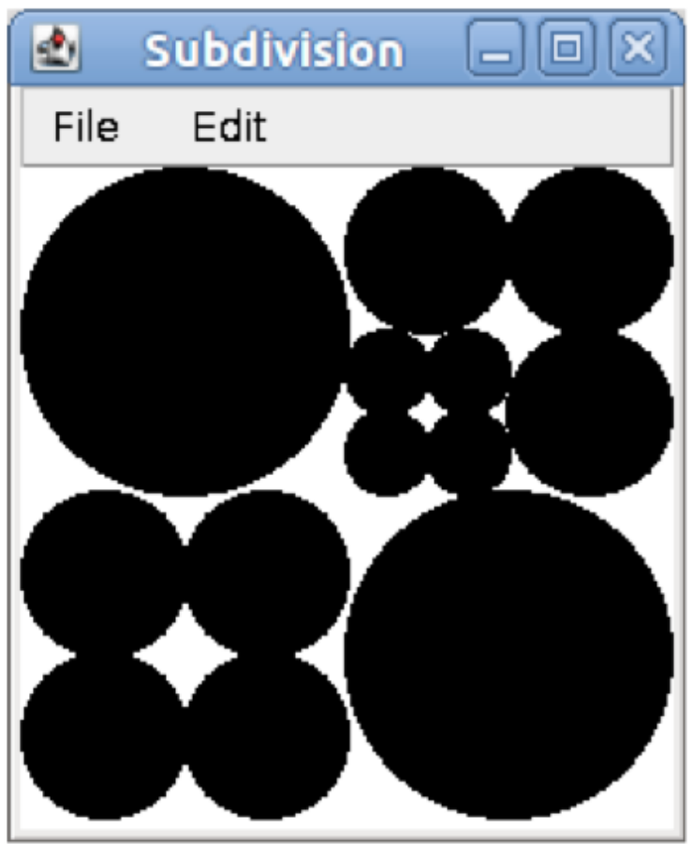
If the user then clicks on any of these four circles, that circle is removed and replaced by four more circles, each of which are one-quarter the size of the chosen circle. For example, if the user clicks on the upper-right circle, the result would be



If the user now clicks the bottom-left circle, the result would be



And finally, if the user clicked the lower-left of the small circles in the upper-right corner, the result would be



Write a program that begins by drawing a filled, black circle whose width and height are the width and height of the window (which you may assume are the same). Whenever the user clicks on an circle, you should remove that circle and add in four new circles such that

- each circle is one-quarter the size of the original circle (half as tall and half as wide),
- each circle is filled black, and
- the four circles are positioned as shown in the above diagrams.

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
```

/* As before, we would normally give you more space to write your answer. */

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {


    }
}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());

    }
}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());

    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }



}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {

        }

}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {
        GObject hit = getElementAt(e.getX(), e.getY());



    }
```



```
}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {
        GObject hit = getElementAt(e.getX(), e.getY());

            replaceObject(hit);

    }
```

```
}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {
        GObject hit = getElementAt(e.getX(), e.getY());
        if (hit != null) {
            replaceObject(hit);
        }
    }



}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {
        GObject hit = getElementAt(e.getX(), e.getY());
        if (hit != null) {
             replaceObject(hit);
        }
    }

    private void replaceObject(GObject hit) {









    }
}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {
        GObject hit = getElementAt(e.getX(), e.getY());
        if (hit != null) {
             replaceObject(hit);
        }
    }

    private void replaceObject(GObject hit) {
        remove(hit);









    }
}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {
        GObject hit = getElementAt(e.getX(), e.getY());
        if (hit != null) {
             replaceObject(hit);
        }
    }

    private void replaceObject(GObject hit) {
        remove(hit);

        double size = hit.getWidth() / 2.0;
        double x = hit.getX();
        double y = hit.getY();









    }
}
```

```
import acm.program.*;
import acm.graphics.*;
import java.awt.*;
import java.awt.event.*;

public class Subdivision extends GraphicsProgram {
    public void run() {
        createCircle(0, 0, getWidth(), getHeight());
        addMouseListeners();
    }

    private void createCircle(double x, double y, double width, double height) {
        GOval circle = new GOval(x, y, width, height);
        circle.setFilled(true);
        add(circle);
    }

    public void mouseClicked(MouseEvent e) {
        GObject hit = getElementAt(e.getX(), e.getY());
        if (hit != null) {
            replaceObject(hit);
        }
    }

    private void replaceObject(GObject hit) {
        remove(hit);

        double size = hit.getWidth() / 2.0;
        double x = hit.getX();
        double y = hit.getY();

        createCircle(x, y, size, size);
        createCircle(x + size, y, size, size);
        createCircle(x, y + size, size, size);
        createCircle(x + size, y + size, size, size);
    }
}
```

The plan

- General info
- Karel
- Expression evaluation
- Program tracing
- Simple Java and randomness
- Graphics and MouseEvents
- String manipulation

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {



}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";




    return result;
}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {


            result += curCh;


    }
    return result;
}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);

            result += curCh;

    }
    return result;
}
```

A warm-up

```
public void run() {
    String str = "1 3 535 2340 29";
    println(removeWhitespace1(str));
    println(removeWhitespace2(str));
}
```

```
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result += curCh;
        }
    }
    return result;
}
```

```java
private String removeWhitespace2(String str) {
    String result = "";
    for (int i = str.length() - 1; i >= 0; i--) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result = curCh + result;
        }
    }
    return result;
}
```

```java
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result += curCh;
        }
    }
    return result;
}
```

```java
private String removeWhitespace2(String str) {
    String result = "";
    for (int i = str.length() - 1; i >= 0; i--) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result = curCh + result;
        }
    }
    return result;
}
```

```java
private String removeWhitespace1(String str) {
    String result = "";
    for (int i = 0; i < str.length(); i++) {
        char curCh = str.charAt(i);
        if (!Character.isWhitespace(curCh)) {
            result += curCh;
        }
    }
    return result;
}
```

More String problem tips

More String problem tips

- First consider how you would do the problem as a human

More String problem tips

- First consider how you would do the problem as a human
  - What do you need to remember between each character?

More String problem tips

- First consider how you would do the problem as a human
    - What do you need to remember between each character?
- Watch out for out-of-bounds!

More String problem tips

- First consider how you would do the problem as a human
    - What do you need to remember between each character?
- Watch out for out-of-bounds!
- Decompose the problem if necessary

Problem Five: Grade-School Arithmetic **(30 Points)**

As mentioned in Chapter 3 of *The Art and Science of Java*, Java's primitive data types (**`int`**, **`double`**, etc.) have ranges on what values they can store. **`int`**, for example, can't hold values greater than 2,147,483,647. Since we may want the computer to work with values larger than this (say, for example, the US national debt or the number of atoms in one gram of carbon), programmers sometimes use other types (such as **`String`**) for large integers. For example:

```
String worldPopulation  = "6993309969";
String rubiksCubeStates = "43252003274489856000";
```

In order to make use of numbers encoded this way, we have to have some way to add them. In grade school, you probably learned how to add two large numbers one digit at a time (a technique appropriately called *grade-school addition*). You write the two numbers out, one on top of the other, then work from the right to the left adding the individual digits of the numbers. When the digits sum up to a value greater than ten, you would write out just the ones digit of their sum, then carry a 1 into the next column. For example, here's 137 + 864:

```
  1   1   1
      1   3   7
  +   8   6   4
-------------------
  1   0   0   1
```

Using your knowledge of Java, you will teach the computer how to add values this way. Your job in this problem is to write a method

```
private String addIntegerStrings(String firstNum, String secondNum)
```

that accepts as input two **`String`**s encoding integers with the same number of digits, then uses grade-school addition to add the two integers represented by those strings. For example:

`addIntegerStrings("44", "99")`	returns `"143"`
`addIntegerStrings("1234", "4321")`	returns `"5555"`
`addIntegerStrings("137", "42")`	cannot happen, since 137 and 42 don't have the same number of digits. You do not need to handle this case in your solution.
`addIntegerStrings("137", "042")`	returns `"179"`
`addIntegerStrings("123123123123123123", "119119119119119119")`	returns `"242242242242242242"`

You may assume the following:

- The strings **`firstNum`** and **`secondNum`** are the same length, meaning that the numbers have the same number of digits.
- The integers encoded by **`firstNum`** and **`secondNum`** are nonnegative, so every encoded integer will be at least 0.
- All the characters in the two input strings are digits, so there are no commas, minus signs, plus signs, etc. Thus you will never get **`"1,000,000"`** or **`"-3"`** as inputs.

Your solution must add the numeric strings using the grade-school algorithm. You cannot assume that the numbers represented by the strings are small enough to be stored as an **`int`**, **`long`**, or **`double`**.

As a hint to make your task a bit easier, the carry from one column to the next can never be anything other than 0 or 1.

```
private String addIntegerStrings(String firstNum, String secondNum) {
```

```
private String addIntegerStrings(String firstNum, String secondNum) {

}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";









    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";
```

left to right or right to left??

```
  1   1   1
      1   3   7
+     8   6   4
-------------------
  1   0   0   1
```

```
    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";



    for (int i = firstNum.length() - 1; i >= 0; i--) {









    }









    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";



    for (int i = firstNum.length() - 1; i >= 0; i--) {
        int firstDigit  = firstNum.charAt(i) - '0';
        int secondDigit = secondNum.charAt(i) - '0';
```

```
    }



    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";



    for (int i = firstNum.length() - 1; i >= 0; i--) {
        int firstDigit  = firstNum.charAt(i) - '0';
        int secondDigit = secondNum.charAt(i) - '0';

        int sum = firstDigit + secondDigit + carry;

    }




    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";
    int carry = 0;

    for (int i = firstNum.length() - 1; i >= 0; i--) {
        int firstDigit  = firstNum.charAt(i) - '0';
        int secondDigit = secondNum.charAt(i) - '0';

        int sum = firstDigit + secondDigit + carry;

    }

    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";
    int carry = 0;

    for (int i = firstNum.length() - 1; i >= 0; i--) {
        int firstDigit  = firstNum.charAt(i) - '0';
        int secondDigit = secondNum.charAt(i) - '0';

        int sum = firstDigit + secondDigit + carry;

        result = (sum % 10) + result;

    }




    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";
    int carry = 0;

    for (int i = firstNum.length() - 1; i >= 0; i--) {
        int firstDigit  = firstNum.charAt(i) - '0';
        int secondDigit = secondNum.charAt(i) - '0';

        int sum = firstDigit + secondDigit + carry;

        result = (sum % 10) + result;
        carry = sum / 10;
    }




    return result;
}
```

```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";
    int carry = 0;

    for (int i = firstNum.length() - 1; i >= 0; i--) {
        int firstDigit  = firstNum.charAt(i) - '0';
        int secondDigit = secondNum.charAt(i) - '0';

        int sum = firstDigit + secondDigit + carry;

        result = (sum % 10) + result;
        carry = sum / 10;
    }

    if (carry != 0) {

    }

    return result;
}
```

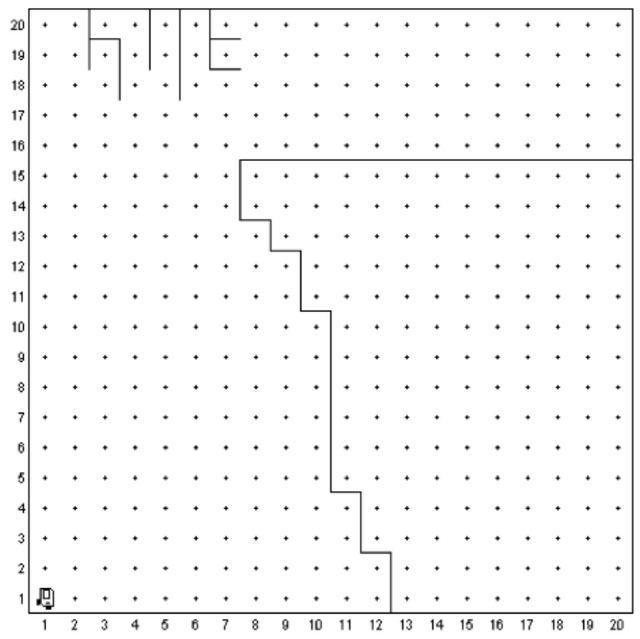
```
private String addIntegerStrings(String firstNum, String secondNum) {
    String result = "";
    int carry = 0;

    for (int i = firstNum.length() - 1; i >= 0; i--) {
        int firstDigit  = firstNum.charAt(i) - '0';
        int secondDigit = secondNum.charAt(i) - '0';

        int sum = firstDigit + secondDigit + carry;

        result = (sum % 10) + result;
        carry = sum / 10;
    }

    if (carry != 0) {
        result = carry + result;
    }

    return result;
}
```

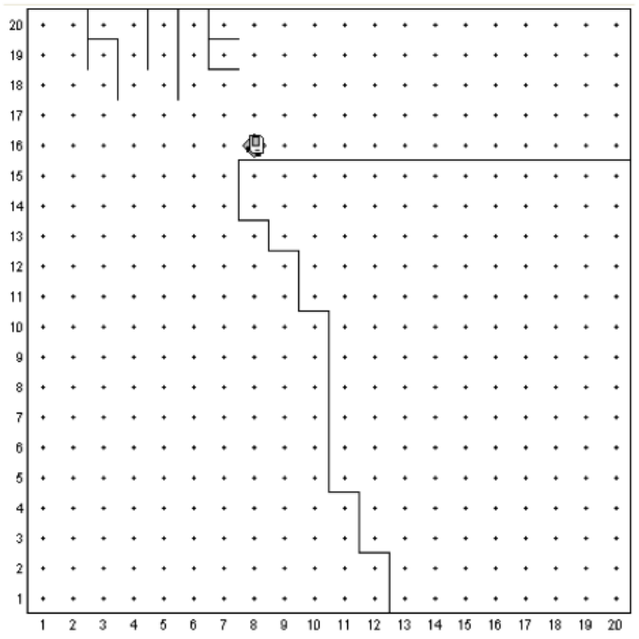
Wish You a Good Luck

Extra problems

RockClimbingKarel

In this problem, you will write a Karel program where Karel climbs a mountain to put a flag at the edge of a cliff. Karel may leave beepers in the process of finding the top, but must clean everything up before it stops. Notice that the cliff can keep growing outwards, but will always grow by one avenue at a time. For example, Karel will transform the world shown below at the left to the world shown below at right:





You can assume the following conditions:

- The world is at least two by two
- There are no beepers on the world
- Karel always starts at (1,1)
- Flying spiders can leave random webs hanging from the top. Actually, this is just for fun and because it's 6am as I'm writing this, but the real condition here is that Karel must climb along the edge of the wall to reach the top.

Here is the starter code:

```
public class RockClimbingKarel extends SuperKarel {

	public void run(){
		//Your code here
	}
}
```

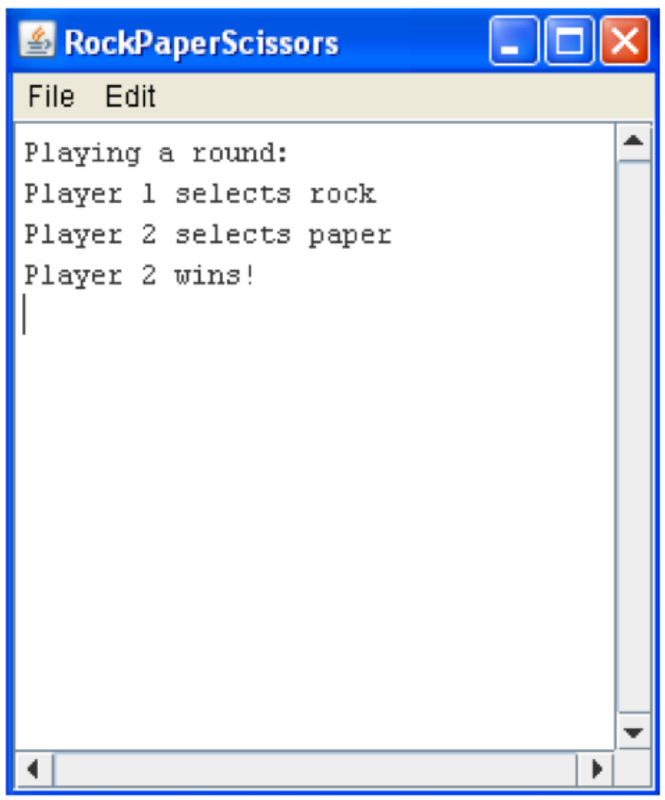
```
public class RockClimbingKarel extends SuperKarel {

	public void run() {
		moveToWall();
		turnLeft();
		climbRock();
		putBeeper();
	}

	private void moveToWall(){
		while(frontIsClear()){
			move();
		}
	}

	private void climbRock(){
		climbToTop();
		positionAtEdge();
	}

	private void climbToTop(){
		while(rightIsBlocked()){
			if(frontIsClear()){
				move();
			}else{
				moveToNextEdge();
			}
		}
	}

	private void moveToNextEdge(){
		turnLeft();
		move();
		turnRight();
		move();
	}

	private void positionAtEdge(){
		turnRight();
		move();
	}
}
```

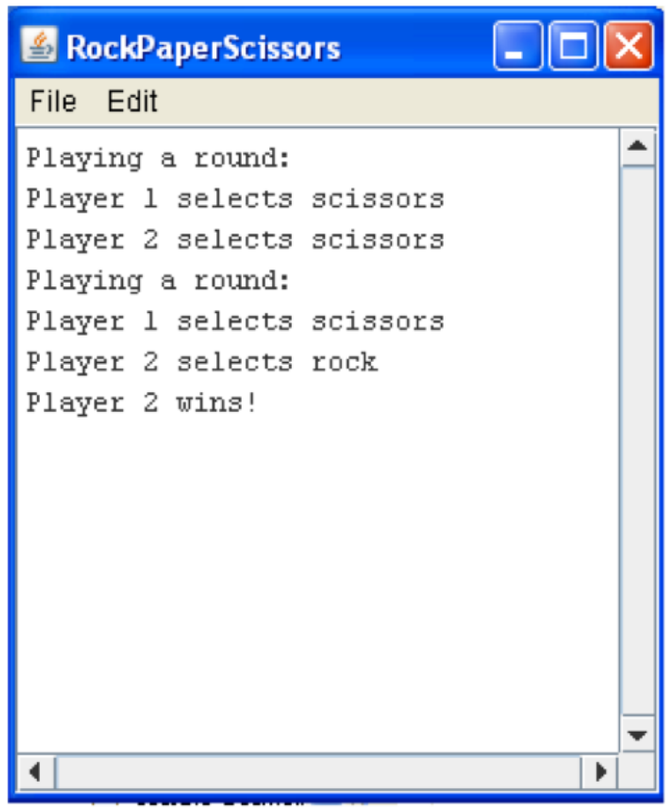
RockPaperScissors

For this program you will play a game of "Rock Paper Scissors". The program will play a game and print the winner. In case of a tie, another turn is played, until one of the players has won. You can assume you have a method that returns the String corresponding to the numeric constant for a selection. The method looks as shown below:

```
private String resultString(int play){}
```

So,
resultingString(ROCK) will return "rock"





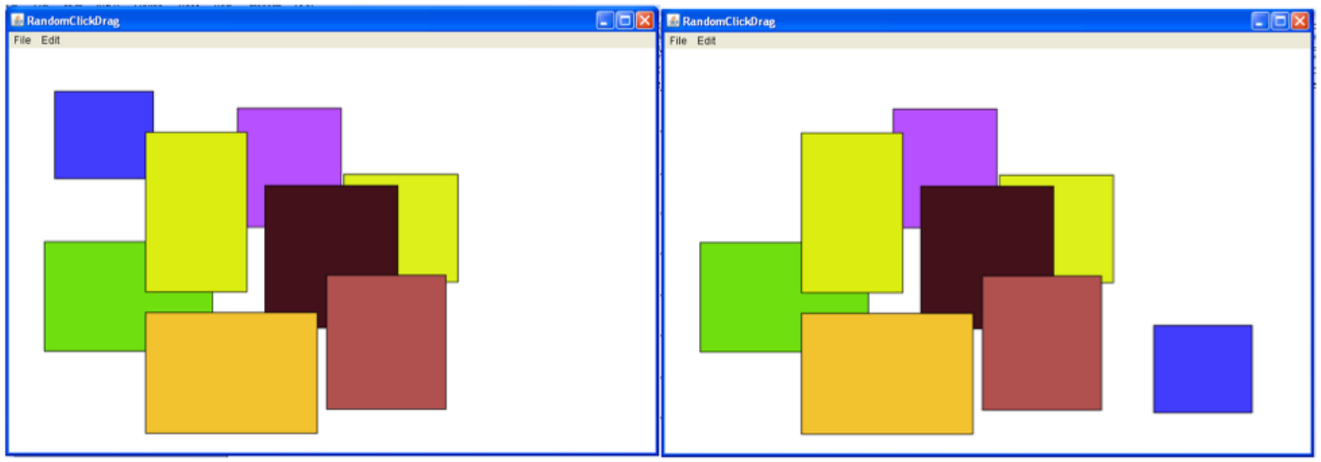
Here is the starter code:

```
public class RockPaperScissors extends ConsoleProgram {

	private static final int ROCK = 0;
	private static final int PAPER = 1;
	private static final int SCISSORS = 2;

	public void run() {
		//Your code here
	}
}
```

```
public class RockPaperScissors extends ConsoleProgram {

	private RandomGenerator rgen = new RandomGenerator();

	private static final int ROCK = 0;
	private static final int PAPER = 1;
	private static final int SCISSORS = 2;

	/* Keep playing rounds until a winner is determined. */
	public void run() {
		int winner = 0; // No winner to begin with.
		while (winner == 0) {
			winner = playRound();
		}
		println("Player " + winner + " wins!");
	}

	/* Determine selections for each player and then return winner (or tie)
*/
	private int playRound() {
		println("Playing a round:");
		int player1 = rgen.nextInt(ROCK, SCISSORS);
		println("Player 1 selects " + resultString(player1));
		int player2 = rgen.nextInt(ROCK, SCISSORS);
		println("Player 2 selects " + resultString(player2));
		return (determineWinner(player1, player2));
	}

	/* Return number of winning player (1 or 2), or 0 to indicate a tie. */
	private int determineWinner(int player1, int player2) {
		if (player1 == player2) {
			return 0;
	/* Player 1 wins if selection is 1 less than player 2's selection,
	 * assuming that we "wrap around" given the 3 values we have.
	 */
		} else if (player1 == (player2 + 1) % 3) {
			return 1;
		} else {
			return 2;
		}
	}

	/* Return string corresponding the numeric constant for a selection. */
	private String resultString(int play) {
		switch (play) {
		case ROCK: return "rock";
		case PAPER: return "paper";
		default: return "scissors";
		}
	}
}
```

RandomClickDrag

In this problem you will write a program that draws GRects of random sizes on the screen at the click of the mouse and allows the user to drag them around the canvas. The GRects can change colors and be dragged around the screen. For example, after a few clicks, your canvas will look as shown below at the left. However, if you press the mouse down on top of the square at the top left, hold, and move it to the bottom right corner, the square will move and the image will now look as shown below at the right.



- Length and width are randomly chosen, as well as the initial color
- Clicking on an existing piece will simply choose a new color for that GRect
- Size is bounded by MAX_LEN and MIN_LEN

Here is the starter code:

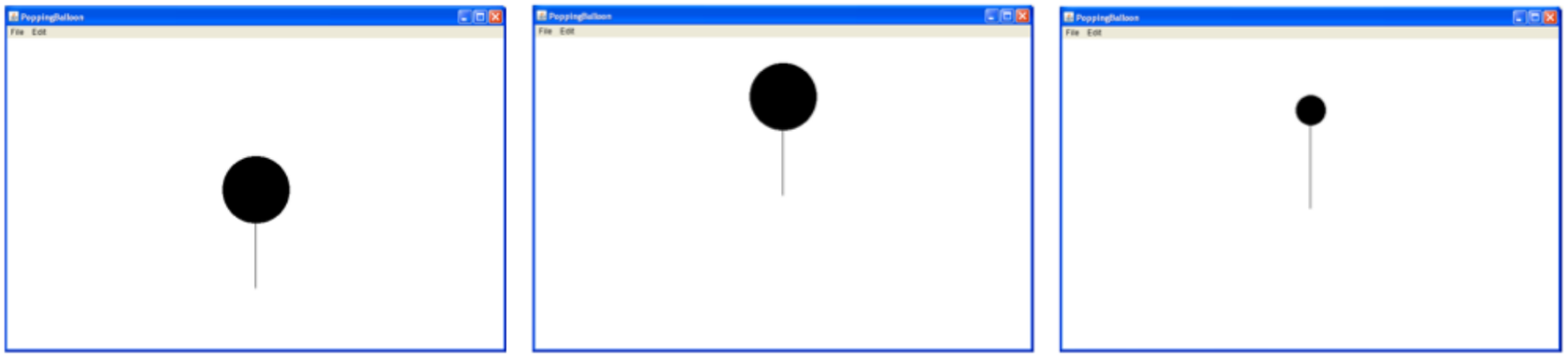
```
public class RandomClickDrag extends GraphicsProgram{

	private static final double MAX_LEN = 200;
	private static final double MIN_LEN = 100;

	private RandomGenerator rgen = new RandomGenerator();

	public void run(){
		//Your code here
	}
}
```

```
public class RandomClickDrag extends GraphicsProgram{

	private static final double MAX_LEN = 200;
	private static final double MIN_LEN = 100;

	private double lastX;
	private double lastY;
	private GObject currentRect;
	private RandomGenerator rgen = new RandomGenerator();

	public void run(){
		addMouseListeners();
	}

	public void mouseClicked(MouseEvent e){
		currentRect = getElementAt(e.getX(), e.getY());
		if(currentRect != null){
			((GRect)currentRect).setFillColor(rgen.nextColor());
		}else{
			GRect rect = new GRect(rgen.nextDouble(MIN_LEN, MAX_LEN),
					rgen.nextDouble(MIN_LEN, MAX_LEN));
			rect.setLocation(e.getX() - (rect.getWidth() / 2),
					e.getY() - (rect.getHeight() / 2));
			rect.setFilled(true);
			rect.setFillColor(rgen.nextColor());
			add(rect);
		}
	}

	public void mousePressed(MouseEvent e){
		lastX = e.getX();
		lastY = e.getY();
		currentRect = getElementAt(e.getX(), e.getY());
	}

	public void mouseDragged(MouseEvent e){
		if(currentRect != null){
			currentRect.move(e.getX() - lastX, e.getY() - lastY);
			lastX = e.getX();
			lastY = e.getY();
		}
	}
}
```

PoppingBalloon

For this problem you will be making a balloon that slowly moves up. When you click on the balloon, it will slowly start deflating and disappear along with the string holding onto it. The balloon should start at the center of the screen. The images below show:

(1) balloon at start
(2) balloon after a few seconds
(3) balloon deflating



Here is the starter code:

```
public class PoppingBalloon extends GraphicsProgram {

	private static final int BALLOON_WIDTH = 100;
	private static final int TAIL_LENGTH = 150;
	private static final int FLOAT_SPEED = 1;
	private static final int SHRINK_SPEED = 2;
	private static final int PAUSE_TIME = 30;

	public void run() {
		//Your code here
	}

}
```

```
public class PoppingBalloon extends GraphicsProgram {

	private static final int BALLOON_WIDTH = 100;
	private static final int TAIL_LENGTH = 150;
	private static final int FLOAT_SPEED = 1;
	private static final int SHRINK_SPEED = 2;
	private static final int PAUSE_TIME = 30;

	private GOval balloon;
	private GLine tail;
	private int dy = -FLOAT_SPEED;
	private int deflation = 0;

	public void run() {
		int cx = getWidth()/2;
		int cy = getHeight()/2;
		balloon = new GOval(cx - (BALLOON_WIDTH / 2),
				cy - (BALLOON_WIDTH / 2),
				BALLOON_WIDTH, BALLOON_WIDTH);
		balloon.setFilled(true);
		add(balloon);
		tail = new GLine(cx, cy + (BALLOON_WIDTH / 2),
				cx, cy + (BALLOON_WIDTH / 2) + TAIL_LENGTH);
		add(tail);
		addMouseListeners();
		animateBalloon();
	}

	private void animateBalloon() {
		while(balloon.getWidth() > 0) {
			balloon.move(0, dy);
			tail.move(0, dy);
			balloon.setSize(balloon.getWidth() - deflation,
					balloon.getHeight() - deflation);
			//Now adjust balloon after click
			balloon.move(deflation / 2.0, deflation);
			pause(PAUSE_TIME);
		}
		remove(balloon);
		remove(tail);
	}

	public void mouseClicked(MouseEvent e){
		if(getElementAt(e.getX(), e.getY()) != null) {
			dy = 0;
			deflation = SHRINK_SPEED;
		}
	}
}
```