Producing Shapes on Screen

Mitsu Ogihara

Department of Computer Science
University of Miami

Table of Contents

Printing a Triangle

Use a Java program to print the following on the screen

```
######/|
#####/#|
####/##|
###/###|
##/####|
#/#####|
/______|
```

Solution

We can use the following code to print the shape:

Solution

We can use the following code to print the shape:

```
1  //-- print a triangle
2  public class Triangle {
3    public static void main( String[] args ) {
4      System.out.println( "######/|" );
5      System.out.println( "#####/#|" );
6      System.out.println( "####/##|" );
7      System.out.println( "###/###|" );
8      System.out.println( "##/####|" );
9      System.out.println( "#/#####|" );
10     System.out.println( "/______|" );
11   }
12 }
```

Solution

We can use the following code to print the shape:

```
1  //-- print a triangle
2  public class Triangle {
3    public static void main( String[] args ) {
4      System.out.println( "######/|" );
5      System.out.println( "#####/#|" );
6      System.out.println( "####/##|" );
7      System.out.println( "###/###|" );
8      System.out.println( "##/####|" );
9      System.out.println( "#/#####|" );
10     System.out.println( "/______|" );
11   }
12 }
```

Recall that the statements in a method are executed from top to bottom

Printing an Isosceles

How about the following shape?

```
######/\######
#####/##\#####
####/####\####
###/######\###
##/########\##
#/##########\#
/____________\
```

Solution

We can use the following code to print the shape:

```
1  //-- print an isosceles
2  public class Isosceles {
3    //-- main method
4    public static void main( String[] args ) {
5      System.out.println( "######/\\######" );  // line 1
6      System.out.println( "#####/##\\#####" );  // line 2
7      System.out.println( "####/####\\####" );  // line 3
8      System.out.println( "###/######\\###" );  // line 4
9      System.out.println( "##/########\\##" );  // line 5
10     System.out.println( "#/##########\\#" );  // line 6
11     System.out.println( "/____________\\" );  // line 7
12   }
13 }
```

Upside Down

We can reverse the order to print an upside down isosceles

```
1  //-- print an isosceles upside down
2  public class UpsideDownIsoscelesCorrect {
3    //-- main method
4    public static void main( String[] args ) {
5      System.out.println( "\\----------/" );  // line 7
6      System.out.println( "#\\########/#" );  // line 6
7      System.out.println( "##\\######/##" );  // line 5
8      System.out.println( "###\\####/###" );  // line 4
9      System.out.println( "####\\##/####" );  // line 3
10     System.out.println( "#####\\/#####" );    // line 2
11     System.out.println( "######\\/######" );    // line 1
12   }
13 }
```

Printing an Upside Down Isosceles

```
% java UpsideDownIsoscelesCorrect
\------------/
#\##########/#
##\########/##
###\######/###
####\####/####
#####\##/#####
######\/######
```

There is no letter for “over-score”

Printing a Quadrant

```
1  public class Quadrant01 {
2    public static void main( String[] args ) {
3      System.out.println( "+--------+--------+" );
4      System.out.println( "|########|########|" );
5      System.out.println( "|########|########|" );
6      System.out.println( "|########|########|" );
7      System.out.println( "|########|########|" );
8      System.out.println( "|########|########|" );
9      System.out.println( "|########|########|" );
10     System.out.println( "+--------+--------+" );
11     System.out.println( "|########|########|" );
12     System.out.println( "|########|########|" );
13     System.out.println( "|########|########|" );
14     System.out.println( "|########|########|" );
15     System.out.println( "|########|########|" );
16     System.out.println( "|########|########|" );
17     System.out.println( "+--------+--------+" );
18   }
19 }
```

Printing a Quadrant

```
% java Quadrant01
+--------+--------+
|########|########|
|########|########|
|########|########|
|########|########|
|########|########|
|########|########|
+--------+--------+
|########|########|
|########|########|
|########|########|
|########|########|
|########|########|
|########|########|
+--------+--------+
```

Table of Contents

Method Specification

A method of a Java class is a group of statements that work together to perform a certain task

Method Specification

A method of a Java class is a group of statements that work together to perform a certain task

The code for a method consists of two parts:

1. **Method Declaration**:
   <attributes> <name>(<parameters>)
2. **Method Body**:
   a series of statements flanked by a pair of { and }

Method Attributes and Parameters

1. Attributes appear in the following order:
   1. Visibility: possibilities are “public”, “private”, “protected”, and unspecified (called “package”), and we will use “public” for now
   2. Static/Instance: possibilities are “static” and unspecified (called “instance”), and we will use “static” for now
   3. Return type: a method can produce a value and hand it over to whoever called the method
      “void” means no value is returned
2. Parameters are the things that the method may refer to in performing its task, specified with a comma in between

Rules about Methods

1. Any number of methods can be defined in a class
2. For a class to be executable (i.e., by way of the command `java <class>`, the class must have a method `public static void main( String[] args )`
3. The order in which the methods appear in a Java file does not matter

Class Without main

The following Java file compiles okay, but cannot be run

```
1  public class HelloWorldNoMain {
2  //  public static void notMain( String[] args ) {
3    public void main( String[] args ) {
4      System.out.println( "Hello, World! This is not \"main.\"" );
5    }
6  }
```

Class Without main

The following Java file compiles okay, but cannot be run

```
1  public class HelloWorldNoMain {
2  //  public static void notMain( String[] args ) {
3    public void main( String[] args ) {
4      System.out.println( "Hello, World! This is not \"main.\"" );
5    }
6  }
```

```
% javac HelloWorldNoMain.java
% java HelloWorldNoMain
Error: Main method not found in class HelloWorldNoMain, please define
the main method as:
   public static void main( String[] args )
or a JavaFX application class must extend javafx.application.Application
```

Table of Contents

Procedural Decomposition

Decomposition ... cutting out a series of statements that collectively perform a certain task into a new method

- The method should be given a unique name <name>
- The series in the original part can be substituted with
  `<name>();`
  We call this a **method call**
- Some methods can be defined to receive values as parameters (such as `System.out.println`) or to return a value
- Decomposition can be used to remove redundancy
- Decomposition introduces some structure inside the class

An example

For example, in:

```
1  public class PrintMessages {
2    public static void main( String[] args ) {
3      System.out.println( "ABC" );
4      System.out.println( "DEF" );
5      System.out.println( "ABC" );
6      System.out.println( "DEF" );
7    }
8  }
```

`System...( "ABC" ); System...( "DEF" );` are repeated twice

An example

For example, in:

```
1  public class PrintMessages {
2    public static void main( String[] args ) {
3      System.out.println( "ABC" );
4      System.out.println( "DEF" );
5      System.out.println( "ABC" );
6      System.out.println( "DEF" );
7    }
8  }
```

`System...( "ABC" ); System...( "DEF" );` are repeated twice

We may bundle the two lines together as a method by the name of `printMessages`

After decomposition

```
1  public class PrintMessages {
2    public static void printMessages() {
3      System.out.println( "ABC" );
4      System.out.println( "DEF" );
5    }
6    public static void main( String[] args ) {
7      printMessages();
8      printMessages();
9    }
10 }
```

Can we decompose this code further?

Quadrant.java

```
1  public class Quadrant01 {
2    public static void main( String[] args ) {
3      System.out.println( "+--------+--------+" );
4      System.out.println( "|########|########|" );
5      System.out.println( "|########|########|" );
6      System.out.println( "|########|########|" );
7      System.out.println( "|########|########|" );
8      System.out.println( "|########|########|" );
9      System.out.println( "|########|########|" );
10     System.out.println( "+--------+--------+" );
11     System.out.println( "|########|########|" );
12     System.out.println( "|########|########|" );
13     System.out.println( "|########|########|" );
14     System.out.println( "|########|########|" );
15     System.out.println( "|########|########|" );
16     System.out.println( "|########|########|" );
17     System.out.println( "+--------+--------+" );
18   }
19 }
```

We will factor out the highlighted part as a method

Quadrant.java

```
1  public class Quadrant02 {
2    public static void topSides() {
3      System.out.println( "|########|########|" );
4      System.out.println( "|########|########|" );
5      System.out.println( "|########|########|" );
6      System.out.println( "|########|########|" );
7      System.out.println( "|########|########|" );
8      System.out.println( "|########|########|" );
9    }
10   public static void main( String[] args ) {
11     System.out.println( "+--------+--------+" );
12     topSides();
13     System.out.println( "+--------+--------+" );
14     topSides();
15     System.out.println( "+--------+--------+" );
16   }
17 }
```

The component

Quadrant.java

```
1  public class Quadrant02 {
2    public static void topSides() {
3      System.out.println( "|########|########|" );
4      System.out.println( "|########|########|" );
5      System.out.println( "|########|########|" );
6      System.out.println( "|########|########|" );
7      System.out.println( "|########|########|" );
8      System.out.println( "|########|########|" );
9    }
10   public static void main( String[] args ) {
11     System.out.println( "+--------+--------+" );
12     topSides();
13     System.out.println( "+--------+--------+" );
14     topSides();
15     System.out.println( "+--------+--------+" );
16   }
17 }
```

The use of the component

Further Decomposition

```
1  public class Quadrant03 {
2    public static void hLine() {
3       System.out.println( "+--------+--------+" );
4    }
5    // the side line
6    public static void side() {
7       System.out.println( "|########|########|" );
8    }
9    // the middle block between the horizontal lines
10   public static void theMiddle() {
11      side();
12      side();
13      side();
14      side();
15      side();
16      side();
17   }
18   // the main
19   public static void main( String[] args ) {
20      hLine();
21      theMiddle();
22      hLine();
23      theMiddle();
24      hLine();
25   }
26 }
```

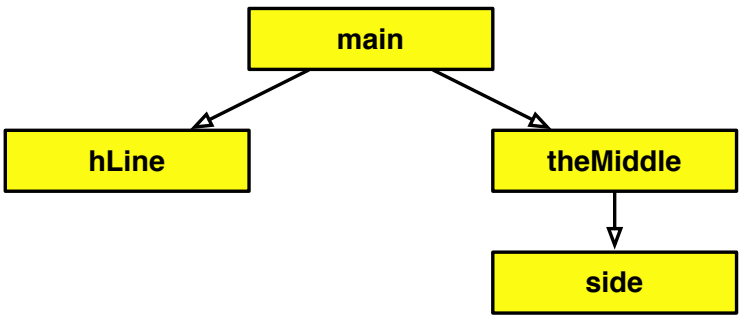
The top, the middle, and the bottom horizontal line

Further Decomposition

```
1  public class Quadrant03 {
2    public static void hLine() {
3       System.out.println( "+--------+--------+" );
4    }
5    // the side line
6    public static void side() {
7       System.out.println( "|########|########|" );
8    }
9    // the middle block between the horizontal lines
10   public static void theMiddle() {
11      side();
12      side();
13      side();
14      side();
15      side();
16      side();
17   }
18   // the main
19   public static void main( String[] args ) {
20      hLine();
21      theMiddle();
22      hLine();
23      theMiddle();
24      hLine();
25   }
26 }
```

The individual rows within the boxes

Method Dependency



Bad Case of Decomposition - Calling Itself

If a method calls itself, a weird situation occurs in which the method keeps calling itself – infinite loop

```
1  public class InfiniteCalls {
2    public static void partOne() {
3      System.out.println( "One" );
4      partTwo();
5    }
6    public static void partTwo() {
7      System.out.println( "Two" );
8      partThree();
9    }
10   public static void partThree() {
11     System.out.println( "Three" );
12     partOne();
13   }
14   public static void main( String[] args ) {
15     partOne();
16   }
17 }
```

Method Call Diagram