Classes and Objects I

Zbigniew Dendzik
Institute of Physics, University of Silesia

2025

Introduction

Below you will find instructions for exercises whose goal is to help you master the basic skills related to programming in the Java language. These skills form the foundation for further exercises, and therefore I treat mastering them within the specified deadline very seriously ;-|. Some exercises are illustrated with code fragments, which are usually meant to be extended as part of a given exercise. If a library class appears in an exercise, then its first occurrence will at the same time be a link to the Java API documentation, where you can find information needed to use it.

Example I

Below you will find the source code of the `Rectangle` class – a compound data structure that we will use to create `Rectangle` objects. The class contains two fields (member variables) of type `double`, two constructors and a method (member function) used to compute the area of a rectangle. Further on you will also find the source code of the `Program` class, which contains the method named `main()`, invoked when execution of class `Program` is started. Within this method the following steps will be performed: declaration of an object of class `Rectangle`, creation of a `Rectangle` object, invocation of the `area()` method on the created object, and printing to standard output of the variable `x` which stores the value returned by this method.

```
1  class Rectangle
2  {
3     double width;
4     double height;
5
6     Rectangle()
7     {
8        this.width=0.0;
9        this.height=0.0;
10    }
11
12    Rectangle(double width,double height)
13    {
14       this.width=width;
15       this.height=height;
```

```
16      }
17
18      double area()
19      {
20          return width*height;
21      }
22  }
23
24  public class Program
25  {
26      public static void main(String[] args)
27      {
28          Rectangle obj;
29          obj=new Rectangle(3,4);
30          double x=obj.area();
31
32          System.out.println("Rectangle area: "+x);
33      }
34  }
```

Listing 1: Example I: classes Rectangle and Program

Ex. 1.1

Using the information from the lecture, identify these elements, analyse the purpose of their use, and their syntax.

Ex. 1.2

Using the information from the lecture, prepare an appropriate compilation unit (you may put both class definitions in a single file whose name is the same as the public class name and with the "`.java`" extension). Compile the project and analyse the compilation results – the number and names of files containing bytecodes for the Java virtual machine. Use any editor (`vim`, `JCreator`, `NetBeans`) and the Java language compiler (`javac`), which is part of the Java SE Development Kit (JDK) development environment provided by Oracle. Then run the program using the interpreter (`java`), which is part of the Java SE Runtime Environment (JRE) that is also included in the JDK package. On web pages you can also find instructions on how to install and use the above tools.

Ex. 1.3

Add to the `Rectangle` class an implementation of the `perimeter()` method returning the perimeter of the rectangle, as well as an appropriate invocation of this method in the program which will allow you to test your implementation. Compile and test the example.

Example II

Write an implementation of the `Point` class used to represent points on the plane. Add to the `Rectangle` class a `center` field, which is an object of class `Point`. Add an appropriate

constructor that allows you to initialise the additional field with information specifying the position of the centre of a given figure on the plane. You can use the example below. Identify the class members, analyse the purpose of their use and their syntax. Prepare an appropriate compilation unit and test the example.

```
class Point
{
   double x;
   double y;

   Point(double x,double y)
   {
      this.x=x;
      this.y=y;
   }

   public String toString()
   {
      return "[x: "+x+", y: "+y+"]";
   }
}

class Rectangle
{
   double width;
   double height;
   Point vertex;

   Rectangle(double width,double height)
   {
      this.width=width;
      this.height=height;
      this.vertex=new Point(0,0);
   }

   Rectangle(double width,double height, Point vertex)
   {
      this.width=width;
      this.height=height;
      this.vertex=vertex;
   }

   public String toString()
   {
      return "[w: "+width+", h: "+height+"]" + vertex.toString();
   }

   double area()
   {
      return width*height;
   }
}
```

```
48
49 public class Program
50 {
51    public static void main(String[] args)
52    {
53       Point obj1;
54       obj1=new Point(-1,1);
55       System.out.println("point: "+obj1);
56
57       Rectangle obj2;
58       obj2=new Rectangle(3,4,obj1);
59       System.out.println("rectangle: "+obj2);
60
61       double p=obj2.area();
62       System.out.println("area: "+p);
63    }
64 }
```

Listing 2: Example II: classes Point and Rectangle

Ex. 1.4

Answer the question: what is the `public String toString()` method used for?

Ex. 1.5

In the `Point` class implement the method `void move(double dx,double dy)` which performs a translation of the point by the vector `(dx,dy)`. In the program, test this method call using any point.

```
1 Point obj;
2 obj = new Point(0,0);
3 System.out.println(obj);
4
5 obj.move(-1,3);
6 System.out.println(obj);
```

Listing 3: Example use of the move method in class Point

Ex. 1.6

Also in the `Rectangle` class implement the method `void move(double u,double v)` which performs a translation of the figure (moves the rectangle vertex) by the vector `(u,v)`. In the definition of this method use the `move()` method from class `Point`. In the program, test this method call using any rectangle.

```
1 Point obj;
2 obj = new Point(0,0);
3
4 Rectangle obj2;
5 obj2=new Rectangle(3,4,obj);
```

```
6 System.out.println(obj2);
7
8 obj2.move(7,-3);
9 System.out.println(obj2);
```

Listing 4: Example use of the move method in class Rectangle

Ex. 1.7

Add to your project the definition of the `Circle` class containing the fields `radius` and `center`. Write implementations of the appropriate constructors and methods, similarly to the `Rectangle` class. Test the example.

Ex. 1.8

Add to the `Rectangle` and `Circle` classes implementations of the method `boolean contains(Point obj)`. In the program, test it using several different rectangles and points.

Ex. 1.9

Add to the `Circle` class an implementation of the method `boolean intersects(Circle obj)`. In the program, test it using several different circles.

Ex. 1.10

Add to the `Rectangle` class an implementation of the method `boolean intersects(Circle obj)`. In the program, test it using several different rectangles and circles. Hint: consider the projections of the circle and rectangle onto the X and Y axes.

Ex. 1.11

Using the `javadoc` tool (a standard tool provided with the JDK package) create standard HTML documentation for the `Rectangle`, `Circle` and `Point` classes, containing descriptions of all fields and methods of these classes.