A Very Short Intro to Java

Clément Aubert.

November 3, 2021

Rectangle.java

```java
public class Rectangle extends Object {
// We have a notion of inheritance, and of Object class

 /*
  * Fields
  */

 private double width;
 private double length;

 /*
  * Constructors
  */

 public Rectangle(double width, double length) { // A custom constructor
  this.width = width;
  this.length = length;
 }

 public Rectangle() { // A second, "no arg" constructor
  this.width = 0;
  this.length = 0;
 }

 // Java will know which one we are calling using the signature
 // (name + input type and order of the method)

 /*
  * Getters and setters
  * There are no properties in Java.
  */

 // Getters, defined manually
 public double getLength() {
  return length;
 }
 public double getWidth() {
```

```java
  return width;
 }

 // Setters, defined manually
 public void setLength(double l) {
  length = l;
 }
 public void setWidth(double width) {
  this.width = width; // We can use the "this" keyword to overcome shadowing.
 }

 /*
  * Other methods
  */

 public final double getArea() {
 // final makes that this method can't be overriden, even by methods in class extending this one
  return length * width;
 }


 public String toString() {
  return ("This rectangle is " + width + " x " + length + " (" + getArea() + ")");
 }

 // This is a really compact equals method.
 public boolean equals(Rectangle r) {
  return r.length == this.length && r.width == this.width;
 }
}
```

Demo.java

```java
/*
 * Clément Aubert
 * https://spots.augusta.edu/caubert/teaching/general/java/
 * 5th November 2019
 */

import java.util.Scanner;   // Importing a java API to read from the keyboard.
import java.io.File;        // Importing a java API to manipulate files.

public class Demo { // Class header
 public static void main(String[] args) { // Method name
  System.out.println("Hi"); // "System" is a Class, "out" an object in this class, "println" a method, and "Hi" a String literal.

  /*
   * Datatypes
   */

```

```java
  int number_of_students = 6; // Variables start with lower case.
  boolean myFlag = true;
  char letter = 'c';
  double tax = 3.2;
  // There is no "decimal" datatype in Java: for monetary amounts, use the BigDecimal
   ↪  class.


  /*
   * If-else-if
   */

  if (!myFlag) { // The boolean operators are !, && and ||.
   System.out.println("Hi to all " + number_of_students + " of you" + "\n \t" + letter);
   // Escape sequences: \n for new line, \t for tabulation
  } else if (tax > 4) {
   System.out.printf("%d, %f, %s, %c \n", 3, 2.5, "Cle", 'C');
   // Cf. below for string formatting
  } else {
   System.out.printf("%f \n", (++tax));
   // prefix increment operator: increment, and then display on the screen
  }

  /*
   * Displaying on the screen
   * The syntax is %[flags][width][.precision]conversion
   * Where a flag is
   *  • , to use comma separators
   *  • 07, to pad with e.g. seven "0"s.
   *  • -, to left justify
   * width is the width of the field printed
   * precision is when the value should be rounded
   * and the conversion is
   *  • f for floating point
   *  • d for integer
   *  • s for string
   *  • c for character
   */


  System.out.printf("%07d \n", 1234);                 // 0001234
  System.out.printf("%09.2f \n", 1.234);              // 000001.23
  System.out.printf("%.1f \n", 78.427);               // 78.4
  System.out.printf("%20f \n", 12.4);                 //            12.400000
  System.out.printf("%-10s %10s \n", "Bob", "Jane");  // Bob              Jane
  System.out.printf("%s \n", 1234);                   // 1234 -- We can use type casting.

  /*
   * String
   */

  String name = "Totoro";
  int size = name.length(); // length is a method in the String class.

```

```java

  if (name.equals("Clément")) { // We can't use equal sign to compare strings, we have to use a method.
   System.out.print("We have the same name!");
  }

  /*
   * Reading from keyboard
   */

  Scanner key = new Scanner(System.in);      // We first create a Scanner object.
  System.out.print("Enter your name:\n");
  String your_name = key.nextLine();         // And then use the nextLine() method to read a String.
  System.out.print("Enter your age:\n");
  int your_age = key.nextInt();              // Reading an Int can be done with nextInt()
  System.out.print(your_name + " entered " + your_age + ".\n");

  /*
   * While loops
   */

  int y = 0;
  while (y < 3 && myFlag) {
   System.out.printf("y is %d, let's increment it.\n", y);
   y++;
  }

  /*
   * Array and for loops.
   */

  int[] numbers; //Declares array reference variable.
  numbers = new int[6]; // Actually creates the array and assign its adress to numbers.

  for (int i = 0; i < numbers.length; i++) {
   numbers[i] = i;
   System.out.println(numbers[i]);
  }

  for (int z = 4; z > 2; z--) {
   System.out.printf("z is %d, let's decrement it.\n", z);
  }

  // We can use the shorthand notation:
  int[] intArray = {
   1,
   2,
   3,
   4,
   5,
   6,
   7,
   8,
```

```
123     9,
124     10
125   };
126
127   // We have exceptions, like:
128   try {
129     for (int i = 0; i < intArray.length; i++) // try to replace the < with <= to raise the
        ↪  exception.
130       System.out.print(intArray[i] + " ");
131   } catch (StringIndexOutOfBoundsException e) {
132     System.out.print("StringIndexOutOfBoundsException");
133   }
134   System.out.println("\n");
135
136   /*
137       try-with-ressources is close to C#'s using statements.
138       The main idea is that:
139         - Java will try to execute the statements between the parenthesis, and create
      ↪  objects in that part,
140               we call those objects "resources".
141         - If everything goes fine, it will execute the body of the try-with-ressource, and,
      ↪  once it is done,
142               it will "destroy" the resources.
143         - If something goes wrong, it will raise the exception.
144       You can read more at
145       <https://docs.oracle.com/javase/tutorial/essential/exceptions/tryResourceClose.html>
146       to be a resource, the class must implement java.lang.AutoCloseable.
147
148       Note that there are many benefits to using try-with-ressources instead of "simply" a
      ↪
149       try-catch-finally block. You can find them discussed at
150       https://stackoverflow.com/a/17739403
151       and
152       https://www.journaldev.com/592/java-try-with-resources
153       In short:
154           - More readable and shorter code,
155           - Automatic resource management,
156           - Support multiple ressources.
157     */
158
159   try (
160     Scanner scanner = new Scanner(new File("test.txt")); // This statement opens the file
        ↪  "test.txt".
161   ) {
162
163     while (scanner.hasNext()) // as long as there is a line that have not been read, move
        ↪  the cursor…
164     {
165       System.out.println(scanner.nextLine()); // read it.
166     } // and loop.
167   } catch (Exception e) { // If opening the file did not work, execute this block.
168     e.printStackTrace();
169   }
170
```

```java
    // When we reach this point, the scanner object does not exist anymore!


    /*
     * Creating objects created in a different class.
     * You have to compile Rectangle.java in the same folder for this to work.
     */

    Rectangle box1 = new Rectangle(2, 4);
    Rectangle box2 = new Rectangle();

    /*
     * Using methods.
     */

    System.out.print(box1 + "\n"); // Implicitely call the toString() method.
    System.out.print(box1.equals(box2) + "\n"); // Gets evaluated to false. We can't use equal sign to compare those objects.
    System.out.print(box1.getArea() + "\n"); // Example of another method.

    /*
     * Of course, you can also have lambda expressions, tuples, ArrayList ("resizable arrays"), interfaces, polymorphism, etc.
     * If you can write a program in an object-orinted programming language, then you can write it in Java.
     */

  }
}
```