Lesson 3: Exercises

Java language: primitive types and objects, basic instructions, assignments, conditionals, loops, strings, arrays, simple input / output, classes and subclasses, constructors, inheritance, polymorphism, visibility, abstract and final classes.

Start: Week of February 17th.
Duration: 4 weeks.

ADVICE: In general it is a good idea to think, discuss and solve each exercise before compiling and executing it. However, after creating each program, it is very convenient to see its actual output.

1) Why the following Java program does not print `Two plus two are 4`?

```
int x = 2;

System.out.println("Two plus two are " + x + x );
```

2) In the following Java code snippet, the second line has been commented out because it gave a compilation error. What error was that? An attempt has been made to fix it in the second line, but the compilation error still remains, and the same happened in the third line. Finally, two equivalent and correct assignments have been achieved. Explain the errors, the wrong attempts to correct them, the two correct instructions, and the final result, 9.

```
byte x = 3;
// error x =  2 + x;
// error x =  2 + (int) x;
// error x =  (byte) 2 + (int) x;
x =  (byte) (2 + (int) x);
x =  (byte) (2 + x);

System.out.println(x + 2);
```

3) Which compilation errors can you detect in the following Java code snippet (assuming these are the only instructions in the main program)? Explain and correct them all, and indicate the values that are finally printed on the output. Please note that the compiler will not detect all errors at once. It is very likely that the compiler first detects two or three errors and, after these are corrected, it detects another two or three different errors in lines where no error was detected before. Why do you think this behavior happens?

```
int x = x + 1;
int y = 2 * x;
int y = y / 3;
int z;

System.out.println("x: " + x
                   + " y: " + y
                   + " z: " + z
                   + " w: " + w
                  );
```

4) Fix the compilation errors in this Java program and explain the difference between the values that appear in the output when executing it.

```
float  a = 2;
float  b = 2.46;
float  c = 2.4618483617391846291736F;
double d = 2.461848361739284629174 6;
double e = 2.461848361739284629174 6F;

System.out.println("c: " + c + " d: " + d + " e: " + e);
```

5) Considering the variables declared in the previous exercise, why, of the following assignments, only those that have a sum are correct? Correct the wrong ones and explain the correct ones.

```
a = c * d;
b = (float) b - e;
c = (float) d / e;
a = (float) d + a;
b = d * 2;
c = (float) 2.0 + b;
a = 2.0 - (float) b;
```

6) Consider the variables declared below, and assume that the first six have already been assigned with the values corresponding to the moments when a car enters and exits a road tunnel. Write the Java instructions needed to calculate the number of seconds elapsed between both moments and assign them to the variable declared in the last declaration.

```
int enterHour, enterMinute, enterSecond;
int exitHour, exitMinute, exitSecond;
long elapsedSeconds = 0;
```

7) Complete the program of the previous exercise to calculate the average speed and km/h of the vehicle inside the tunnel, knowing the length of the tunnel. The program should print a message in case the car exceeds the maximum speed declared below.

```
final float TUNNEL_LENGTH_IN_KM = 3.47F;
final int SPEED_MAX_IN_KMH = 100;
double vehicleSpeed;
```

8) Have you tested your program with cases like those below? Create a complete program, execute these test cases, and explain the results.

```
// case 1:
enterHour =11; enterMinute =59; enterSecond =42;
exitHour =12;  exitMinute =0;   exitSecond =09;

// caso 2:
enterHour =08; enterMinute =58; enterSecond =07;
exitHour =09;  exitMinute =01;  exitSecond =02;

// caso 3:
enterHour =23; enterMinute =59; enterSecond =42;
exitHour =0;  exitMinute =1;    exitSecond =9;
```

9) Consider the variables declared below, and assume that the first one has already been assigned with the value corresponding to the CPU time (in milliseconds) consumed to execute a program. Write the necessary Java instructions to: calculate the number of hours, minutes and seconds consumed, assign them to the variables declared below for that purpose, and print their values at the output.

```
long cpuMilliseconds = 0;
long execHour = 0, execMin = 0, execSecs = 0;
```

10) Extend the previous exercise to calculate the total amount to be invoiced (with 18% VAT) for CPU consumption, taking into account that a fixed cost is charged for the first hour (or fraction), half of the fixed cost is charged for each of the following 2 periods of ½ hour (or fraction), a ¼ of the fixed cost is charged for each of the following 4 periods of ¼ hour (or fraction), ⅛ of the cost is charged for each of the next 8 periods of ⅛ hour (or fraction), and so on.

```
final long FIXED_COST = 10;
double totalInvoice;
```

11) The following code does not print any "true"? Would it print some "true" if we initialized y with 9 instead of 3? Nope. And what would happen if we exchanged the names of the variables x and y in their respective declarations in the initial version of the program fragment? Nope!

```
short x = 9;
```

```
byte  y = 3;

System.out.println( x < y );
System.out.println( x = y );
System.out.println( x > y );
```

12) In the following code, we have corrected a compilation error, and now it can execute, but we obtain an unexpected result. Why the second comparison is false and true the first one?

```
int n1 = 1234567890;
// float n2 = n1 + 0.3; Compilation error. Why?
float n2 = n1 + 0.3F;

System.out.println( n1 < n2 );
System.out.println( n1 < n1 + 0.3 );
```

13) In the following code, we combine the `char` type (2 bytes) with two numeric types: `short` (2 bytes) and `int` (4 bytes). We have added all needed *castings* to avoid compilation errors, but we obtain an unexpected results. Why `s - i` is not zero, but `c2 - c3` is zerp?

```
char c1 = '\u8001';
short s = (short) c1;
char c2 = (char) s;
int i = c1;  // Casting not required, why?
char c3 = (char) i;

System.out.println( i - s );
System.out.println( c2 - c3 );
```

14) In the following program, we wanted to print a String, and in the next line, a String with spaces and an special character when the first occurrence of soughtLetter os found. Hence, we wanted to obatin this output when lookgin for a "g":

```
My glass is empty.
   *
```

But we obtain this one

```
My glass is empty.
My *
```

Fix the program to obtain the desired output. Check https://docs.oracle.com/javase/9/docs/api/java/lang/String.html for details of the available String operations. Remember that a regular expression like "." matches any character.

```
public class LetterMarker {
      public static void main(String[] args) {

            String text = "Mi glass is empty.";
            char soughtLetter = 'g';
            char marker = '*';

            String copy = text;
            text.replaceAll("."," ");

            int pos = text.indexOf(soughtLetter);
            copy = copy.substring(0,pos) + marker;

            System.out.println(text);
            System.out.println(copy);
      }
}
```

What happens if the `soughtLetter` does not appear in `text`? Fix that problem.

15) Extend the previous program to produce markers in all appearances of the sought letter, so that we obtain the following output:

```
My glass of ginger ale is green
   *        *  *          *
```

Hint: Since we do not know how many occurrences of the sought letter will have to be marked, it seems necessary to use some kind of loop. It can also be useful to check if there are several forms of the `indexOf` method and if some of them allows to not always search from the beginning of the string.

16) Modify the previous program to look for occurrences of substrings. Hence, if we look for "gr", we should obtain the following output

```
My great glass of ginger ale is green
   *                            *
```

17) Extend the previous program to read the strings from a text file. Both the substring to be located and the file name should be given as command line arguments. Hence, if the content of file base.txt is

```
My great glass of ginger ale is green
Grand challenges for grannies
Lettuces grow in the ground
```

Executing **`java Exercise17 gr base.txt`** should produce the following output

```
My great glass of ginger ale is green
   *                            *
Grand challenges for grannies
                        *
Lettuces grow in the ground
         *           *
```

18) For this exercise you have two arrays storing information corresponding to various intervals of a trip. The i-th element of the minutes array indicates how many minutes that interval lasted and the i-th element of the speed array indicates the average speed recorded in that interval. Based on this information, you must calculate the average global travel speed in km / hour, and you have done it using the following program, accumulating the kilometers traveled in each interval and dividing that result by the total number of minutes spent on the trip.

```
public class GlobalAverageSpeed {
    public static void main(String[] args) {

        int[] minutes =  { 4, 8,10, 6, 5,12, 2};  // minutes
        int[] speed   =  {30,50,90,80,70,50,20};  // km/hour

        double distance = 0.0;
        int    totalMinutes = 0;

        for (int i=0; i<minutes.length; i++) {
          totalMinutes += minutes[i];
          distance += minutes[i] * speed[i] / 60;  // Line *1*
        }

        System.out.println(distance / (totalMinutes / 60)); // Line *2*
    }
}
```

However, the result that comes out of the program is close to 0.016312 km / hour, which is obviously incorrect. Therefore, you ask a friend, who suggests that in the expression of line * 2 * it might be better to use parentheses to first divide totalMinutes by 60 and then make the other division of the same expression. You made this change, and the result is Infinity. Without thinking much again you leave line * 2 * as it was and test, just in case, putting the parentheses in the expression of line * 1 * to make the division by 60 first. The result is now close to 0.0074468 km / hour, or even less than at the beginning. So you leave the program again as it was at the beginning, and then another friend tells

you that putting brackets on the line * 2 * had told you well but that you also have to divide by 60.0 instead of 60 and with these two changes in the line * 2 * obtain an average global speed of approximately 58.7234 km / hour that seems much more reasonable than the other values.
Reasonably explain the mistakes that you have been trying to correct with all those changes that you have tried, and please, fix the program ASAP because the teacher says that he does not believe the result is less than 60 km / hour .

19) Write a Java program that accepts an odd number as the unique argument of the command line and use it as a diagonal size to print a rhombus like the following (printed when given 9 as an argument):

20) The following program (though not really perfect) aims at detecting if the integer that is passed as the first argument on the command line is higher than a certain threshold. What can the compiler complain about? Propose two ways to make it compile, maintaining its apparent logic.

```
public class Thresholds {
    public static void main(String[] args) {
        final int THRESHOLD = 9;
        int n = Integer.parseInt(args[0]);
        String result;

        if (n > THRESHOLD) { result  = "mayor"; }
        if (n <= THRESHOLD) { result  = "no mayor"; }
        System.out.println(result);

    }
}
```

21) Write a Java program that guesses the natural number (less than 1000) that the user has thought, asking a series of questions about whether it is less than or greater than a candidate number provisionally chosen by the program.

The program can use binary search in the range of possible values, provisionally choosing the midpoint of the interval as a candidate to ask the user. Use `System.in.read()` to find out if the user presses the '<' or '>' key to indicate if their number is less than or greater than the candidate presented, or press '=' if the proposed candidate matches her number. Please note that the console input (in) requires the user to press the return (enter) key after the chosen key, either <,>, or =. This generates additional characters that your program should ignore. Check https://docs.oracle.com/javase/9/docs/api/java/lang/System.html#in for details of System.in.

22) Write a program that takes two file names as command line arguments and generates a file with the second name and whose content is the result reversing line by line and word by word the content of the first file (where a word is any string separated by one or more White spaces). That is, if the first file contains

```
My glass   of wine is empty.
I like better a glass of water
Good day!
```

The generated file should contain the following

```
day! Good
```

```
water of glass a better like I
empty. is wine of glass My
```

Hints: A posible strategy is storing the entire input file in memory in a list of lines where each line is stored as a list of Strings (words), and then process the arrays in reverse order.

23) Without using the compiler, find and explain the 6 changes that must be made in this program so that it does not produce compilation errors (but without completely changing its structure). After that, use the compiler to verify that it is also unable to detect all errors on the first attempt. Fix the ones the compiler detects and repeat the compilation. How many times did you had to run the compiler to eliminate all the errors?

```
public class Errors1 {
	String greet = "Hello world!;"

	public static void main(String[] args) {
		greet("");
	}// main

	public static greet(String extraGreet);
	{
		System.println(greet, extraGreet);
	}
} // class
```

24) This program has a compilation error that can be fixed immediately. However, it is related to another more serious error: the program does not produce the desired result. What is the compilation error and what is the other most serious error? Correct them and try your program.

```
public class Temperatures {
	public static void main(String[] args) {
		double fahrenheitTemp = 103.5;
		double celsiusTemp;

		toCelsius(celsiusTemp, fahrenheitTemp);
		System.out.println(fahrenheitTemp + "ºF are "
				+ celsiusTemp + "ºC");
	}// main

	public static void toCelsius(double c, double f) {
		c =  5.0 / 9.0 * (f - 32.0);
	}
}// class
```

25) Expand the previous program, adding another method to convert from Celsius to Fahrenheit and prove that applying the two conversions gives the same original temperature. Repeat the conversion of the same temperature in both directions a very high number of times, checking in each repetition that we continue to have an identical value to the original temperature, or print a message with the difference found and the iteration number in which was found.

26) The following Java fragment defines a class Book. Complete the class with three public constructors: one for books of which we only know title, author and publisher (but asume it has not been published yet); another for when we also know the price (and we still assume it has not yet been published); and the third for when we also know the number of copies sold (and therefore, we asume the book has been published). Try to reuse the code of the constructors as much as possible.

```
public class Book {

  private String title;
  private String author;
  private String publisher;
```

```
    private double price;
    private int soldCopies;
    private boolean sinPublicar;
}// class
```

27) Find the three compilation errors in this class that manipulates two scale factors for calculating areas of simple geometric figures.

```
public class Scale {
	private int scaleX;
	private int scaleY;

	private Scale() {
		scaleX = 1;
		scaleY = 1;
	}
	public Scale(int x, int y) {
		scaleX = x;
		scaleY = y;
	}
	public Scale(int x) {
		this();
		this.scaleX = x;
	}
	public Scale(int y) {
		this(1,y);
		this.scaleY = y;
	}

	public int rectangleArea(int base, int height) {
		return scaleX * base * scaleY * height;
	}

	public double rectangleArea(int base, int height) {
		return (double) (scaleX * base * scaleY * height);
	}

	public double squareArea(double side) {
		this.rectangleArea((int) side, (int) side);
	}
}
```

28) Check the following Java classes, just in case they contain errors, and fix them before answering the questions below.

```
public final class FinalSubclass extends IntermediateSubclass {
	public FinalSubclass() {
		System.out.println("finally");
	}
	public FinalSubclass(String e) {
		System.out.println(e);
	}

	public int strange(int x, int y) {return super.strange(x,x) + y;}
} // class FinalSubclass

public class IntermediateSubclass extends Abstract {
	IntermediateSubclass() {
		this("but OK");
		System.out.println("and happy");
```

```
    }
    private IntermediateSubclass(String e) {
        System.out.println(e);
    }

    public int strange(int x, int y) {return x + y;}
} // class IntermediateSubclass

public abstract class Abstract {
    protected Abstract() {
        System.out.println("abstract class");
    }

    abstract public int strange(int x, int y);
}  // class Abstract
```

Question 1: What is printed when making each of these object declarations? Match the cases that are similar to each other and explain why they are.

```
FinalSubclass        a = new FinalSubclass ();
FinalSubclass        b = new FinalSubclass ("===");
IntermediateSubclass c = new IntermediateSubclass ();
Abstract             d = new IntermediateSubclass ();
Abstract             e = new FinalSubclass ("...");
```

Question 2: Assumming the previous declarations. What do the next instructions print? Match the cases that are similar to each other and explain why they are.

```
System.out.println(a.strange (2,10));
System.out.println(b.strange (2,10));
System.out.println(c.strange (2,10));
System.out.println(d.strange (2,10));
System.out.println(e.strange (2,10));
```

29) Find all errors in the following code

```
abstract public class Abstract {
    public int z = 0;

    abstract public void mistery(int x, int y);
    public int dark(int x, int y);
}

class Subclass extends Abstract {
    final public int dark(int x, int y) {
        z = z + x * y;
    }
    abstract public void strange(int x, int y);
}

final class LeafClass extends Subclass {
    abstract public void mistery(int x, int y) {
        z = x + y;
    }
    public int dark(int x, int y) {
        z =  z * x * y;
    }
    protected void strange(int x, int y) {
        z = z + x + y;
    }
    protected void strange(int x) {
        z = z + x;
    }
}
```

```
class Class extends LeafClass {
      public void strange(int x) { z = z + x * x; }
}
```

30) Analyze the following program and explain all the errors that the compiler can detect. Eliminate all lines containing an error and explain the program output.

```
class Surprise {
      private final void freak() { System.out.println("Unknown"); }

      public Surprise(String s) { freak(); }
      public Surprise() { }
}

public class Mistery extends Surprise {
      public void freak() { System.out.println("Mistery"); }

      public static void main(String [] args) {
            Mistery x = new Mistery();
            x.freak();
            Surprise y = new Surprise("Hello!");
            y.freak();
      }
}
```

31) Analyze the following program and explain all the errors that the compiler can detect. Eliminate all lines containing an error and explain the program output.

```
class Alpha {
      static String s = ".";
      protected Alpha() { s += "alpha "; }
}

class SubAlpha extends Alpha {
      private SubAlpha() { s += "sub "; }
}

public class AlphaTest extends Alpha {
      private AlphaTest() { s += "subsub "; }

      public static void main(String[] args) {
            new SubAlpha();
            System.out.println(s);
            new AlphaTest();
            System.out.println(s);
            new Alpha();
            System.out.println(s);
      }
}
```

32) Analyze the following program and justify its output (if it is correct), or explain the errors it contains, proposing a reasonable way to correct each one of them, and explain the output produced after the corrections.

```
class Mammal {
	String name = " furry ";
	String makeNoise() { return " growl "; }
	String getName() { return name; }
}

class Elephant extends Mammal {
	String name = " trunk ";
	String makeNoise() { return " trumpet "; }
}

public class ZooTest {

	public static void main(String[] args) {
		new ZooTest().go();
	}

	void go() {
		Mammal m = new Elephant();
		System.out.println(m.name + m.makeNoise() + m.getName());
		Mammal n = new Mammal();
		System.out.println(n.name + n.makeNoise() + n.getName());
		Elephant p = new Elephant();
		System.out.println(p.name + p.makeNoise() + p.getName());

	}
}
```

33) Analyze the following program and justify its output (if it is correct), or explain the errors it contains, proposing a reasonable way to correct each one of them, and explain the output produced after the corrections.

```
class Top {
	public Top(String s) {
		System.out.println("TOP: " + s);
    }
}

public class Test extends Top {
  public Test(String s) {
	System.out.println("Test: " + s);
  }

  public static void main(String[] args) {
    new Test("MAIN");
    System.out.println("END");
  }
}
```

34) Analyze the following program, and explain all errors that the compiler can detect. Delete all lines from the main method that contain some error, and explain the output produced when executing the program after eliminating the errors.

```
package packet;

public class Simple {
       int a = 5;
       protected int b = 6;

       public Simple() { System.out.println("New Simple"); }
}


package picket;
import packet.*;
public class Magic extends Simple {

   public Magic() { System.out.println("New Magic"); }

   public static void main(String[] args) {
     Simple f = new Simple();
     Magic  m = new Magic();
     Simple s = new Magic();
     Magic  r = new Simple();
     System.out.println("1: " + f.a);
     System.out.println("2: " + f.b);
     System.out.println("3: " + m.a);
     System.out.println("4: " + m.b);
     System.out.println("5: " + s.a);
     System.out.println("6: " + s.b);
     System.out.println("7: " + r.a);
     System.out.println("8: " + r.b);
   }
}
```