Java Language

Java is a high-level, interpreted, object-oriented programming language originally developed by Sun Microsystems and released in 1995. Java runs on a variety of platforms, such as Windows, Mac OS, and UNIX. With the advancement of Java and its widespread popularity, multiple configurations were built to suit various types of platforms. For example: J2EE for Enterprise Applications, J2ME for Mobile Applications.

1) Preliminaries

In this Review, we will use the Eclipse Framework. Normally the machines in the university are already equipped with Eclipse. If you want to install in your personal machine, we'll need :

– to download Eclipse in https://www.eclipse.org/
– if haven't done yet, you'll need to download a JDK (Java SE Development Kit). Just search “Java SE Development Kit Download” in your favorite research engine on the internet, download it from the oracle website according to your operating system, and install it.

If you use your personal machine, and you don't want to install Eclipse, you can try the exercises using the on-line compiler available at :
- **CodingGround** (https://www.tutorialspoint.com/codingground.htm)
- Scroll down until « Online IDEs »
- Choose Java among the options
- You can then write your Java code, compile and execute it. The result is displayed in the terminal zone, at the bottom of the page.
- If you use CodingGround, please save your exercices in .c files so that the professor can validate your solutions

Once in Eclipse, create a workspace only for these Review classes (in the first pop-up Eclipse opens when executed, to select the workspace).

2) Java – Basics

When we consider a Java program, it can be defined as a collection of objects that communicate via invoking each other's methods. Let us now briefly look into the main concepts.

- **Object** − Objects have states and behaviors. Example: A dog has states - color, name, breed as well as behavior such as wagging their tail, barking, eating. An object is an instance of a class.
- **Class** − A class can be defined as a template/blueprint that describes the behavior/state that the object of its type supports.
- **Methods** − A method is basically a behavior. A class can contain many methods. It is in methods where the logics are written, data is manipulated and all the actions are executed.
- **Instance Variables** − Each object has its unique set of instance variables. An object's state is created by the values assigned to these instance variables.

About Java programs, it is very important to keep in mind the following points.

- **Case Sensitivity** − Java is case sensitive, which means identifier **Hello** and **hello** would have different meaning in Java.

- **Class Names** − For all class names the first letter should be in Upper Case. If several words are used to form a name of the class, each inner word's first letter should be in Upper Case.

  Example**:** *class MyFirstJavaClass*

- **Method Names** − All method names should start with a Lower Case letter. If several words are used to form the name of the method, then each inner word's first letter should be in Upper Case.

  Example**:** *public void myMethodName()*

- **Program File Name** − Name of the program file should exactly match the class name.

  When saving the file, you should save it using the class name (Remember Java is case sensitive) and append '.java' to the end of the name (if the file name and the class name do not match, your program will not compile).

  Example**:** Assume 'MyFirstJavaProgram' is the class name. Then the file should be saved as *'MyFirstJavaProgram.java'*

- **public static void main(String args[])** − Java program processing starts from the main() method which is a mandatory part of every Java program.

3) Writing Good Programs

The only way to learn programming is by DOING IT. It is easy to write programs that work. It is much harder to write programs that not only work but also easy to maintain and understood by others, i.e., *good programs*. Writing program is not meaningful. You have to write good programs, so that others can understand and maintain your programs.
Pay particular attention to:

- Coding Style:
  - Follow the **Java Naming Conventions** for variables, methods, and classes STRICTLY. Use camel-case for names. Use nouns for variables (e.g., radius) and class names (e.g., Circle). Use verbs for methods (e.g., getArea(), isEmpty()).
  - Use **Meaningful Names**: Do not use names like a, b, c, d, x, x1, x2, and x1688. Avoid single-alphabet names like i, j, k. They are easy to type, but usually meaningless. Use single-alphabet names only when their meaning is clear, e.g., x, y, z for co-ordinates and i for array index. Use meaningful names like row and col (instead of x and y, i and j, x1 and x2), numStudents, maxGrade, size, and upperbound. Differentiate between singular and plural nouns (e.g., use books for an array of books, and book for each item).
  - Use consistent **indentation** and coding style. Many IDEs (such as Eclipse/NetBeans) can re-format your source codes with a click.
- Program Documentation: Comment! Comment! and more Comment!

(source : https://www3.ntu.edu.sg/home/ehchua/programming/java/J2a_BasicsExercises.html)

4) Hello Word in Java

Open the Java Perspective in Eclipse: If you're not already in the Java perspective, in the main menu select Window > Perspective > Open Perspective > Other > Java.

Create a Java Project: File > New > Java Project. Enter “ReviewJavaExos” for the Project Name and Finish.

Create your HelloWorld class: The next step is to create a new class. In the main toolbar again, click on the New button > Class. If not already specified, select ReviewJavaExos/src as the source folder. Enter HelloWorld for the class name, select the checkbox to create the main() method, then click Finish.

Add a print statement: Now that you have your HelloWorld class, in the main() method, add the following statement:

    System.out.println("Hello world!");

Then save your changes; the class will automatically compile upon saving. You should have the following code at this point :

```java
public class HelloWorld {
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello world!");
    }
}
```

Run your Java application : To run your application, right-click on your class in the Package Explorer and select Run As > Java Application. The Console view should appear at the bottom and display the "Hello, world!" output.

For the following exercises, you can refer to the **tutorial** available in https://www.tutorialspoint.com/java/index.htm , or any other tutorial available on-line.

For the following exercises, create the classes in the “ReviewJavaExos/src” source folder.

5) Exercises on Keyboard Input

5.1. Write a Java program to prompt a user to input his/her name and then the output will be shown as : Hello Dara!

5.2. Write a Java program to get the character at the given index within a given word.

6) Exercises on Flow Controls

6.1. Exercises on Conditional (Decision)

6.1.1. Write a Java program to allow the user to input his/her age. Then the program will show if the person is eligible to vote. A person who is eligible to vote must be older than or equal to 18 years old.

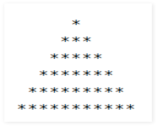
6.1.2. Write a Java program to determine whether an input number is an even number.

6.2. Exercises on Loop (Iteration)

6.2.1. Write a program to print the odd numbers from 1 to 99. Prints one number per line.

6.2.2. Find the Transpose of given 3x3 Matrix in Java. The transpose of a matrix is a new matrix whose rows are the columns of the original. (This makes the columns of the new matrix the rows of the original).

6.2.3. Write a Java program to print a triangle made of '*', of which number of lines n is given. For instance, for n = 6, the output is :



```
     *
    ***
   *****
  *******
 *********
***********
```

7) Exercises on String and char Operations

7.1. Write a Java program that remove vowels from a string.

You may need to use the class BufferedReader to read a string and the class Character.

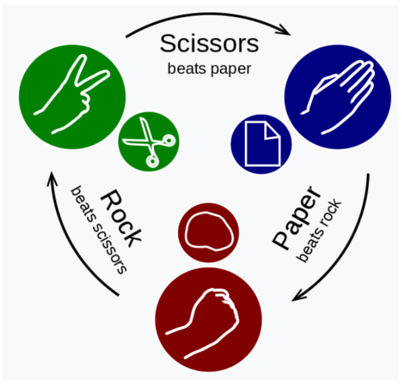
8) Advanced Exercises

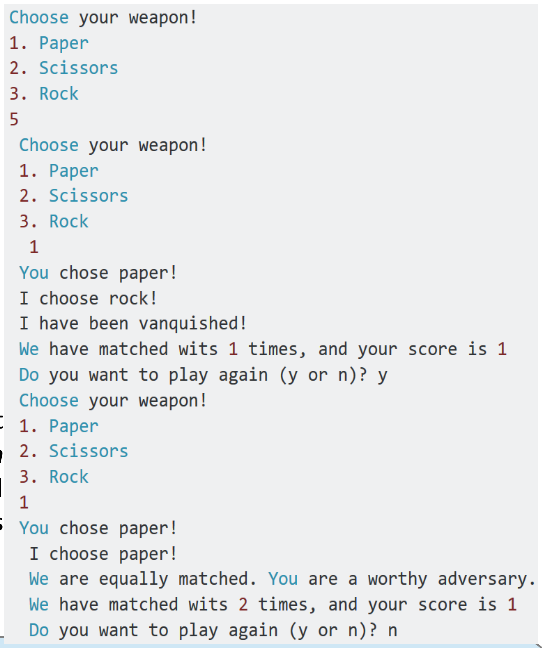
8.1. **Number Guessing Game** : Write a Java program that will generate a random number between 1 and 100, and make the user guess at most 5 times. At each attempt, the program should say the user if the input value is too low, too high, or invalid (not in the range).

8.2. **Hangman :** Randomly select a word from an array of pre-defined words, have the user guess characters in the word. For each character they guess that is not in the word, have it draw another part of a man hanging in a noose. If the picture is completed before they guess all the characters, they lose.

8.3. **Package Rice Bags:** Provided that you have a given number of small rice bags (1 kilo each) and big rice bags (5 kilo each), write a method that returns the minimum number of small bags necessary to package goal kilos of rice, knowing that the big bags are filled before the small bags. Return -1 if it is not possible to package the required rice amount with the bags provided.

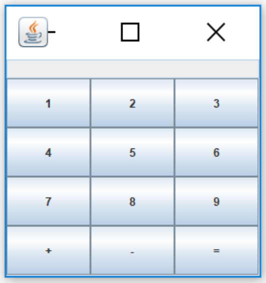
8.4. **Rock-paper-scissors** is a hand game usually played between two people, in which each player simultaneously forms one of three shapes with an outstretched hand. Implement this game in Java. Bellow a possible output for your program.





```
Choose your weapon!
1. Paper
2. Scissors
3. Rock
5
 Choose your weapon!
 1. Paper
 2. Scissors
 3. Rock
  1
 You chose paper!
 I choose rock!
 I have been vanquished!
 We have matched wits 1 times, and your score is 1
 Do you want to play again (y or n)? y
 Choose your weapon!
 1. Paper
 2. Scissors
 3. Rock
 1
 You chose paper!
  I choose paper!
  We are equally matched. You are a worthy adversary.
  We have matched wits 2 times, and your score is 1
  Do you want to play again (y or n)? n
```

8.5. **Calculator**: Create a program that implements the *Addition* and *Subtraction* basic operations in Math. Your program will provide a graphical user interface (GUI) as displayed bellow.



You may use the Java AWT (Abstract Window Toolkit) API,

to develop GUI or window-based applications in java

Follow the tutorial <https://www.javatpoint.com/java-awt> to discover this API

Extend your program to calculate also *Multiplication* and *Division*.

8.6.