Questionnaire No. 01: BASIC CONCEPTS OF THE OBJECT-ORIENTED PROGRAMMING PARADIGM AND DESIGN PRINCIPLES

1. Write statements that can be used in a Java program to read two integers and display the number of integers that lie between them, including integers. For example, four whole numbers are between 3 and 6: 3, 4, 5, and 6.

```
package ec.edu.espe.numbersimulator.view;
import java.util.Scanner;
public class DeclarationsNumber {

    public static void main(String[] args){

        int number1;
        int number2;
        int count = 0;

        System.out.println("enter number 1");
        Scanner num1 = new Scanner(System.in);
        number1 = num1.nextInt();
        System.out.println("*******************************");
        System.out.println("enter number 2");
        Scanner num2 = new Scanner(System.in);
        number2 = num2.nextInt();

        if(number1 <= number2){
            System.out.println("*******************************");
            System.out.println("those numbers are: \n");
            for(int i = number1 ; i <= number2 ; i++ ){

                count++;
                System.out.println(i);
            }

            System.out.println("*******************************");
            System.out.println(count++ +" whole numbers are between "+number1+" and
"+number2);

        }else {
            System.out.println("*******************************");
            System.out.println("those numbers are: \n");
            for (int i = number2 ; i <= number1 ; i++){

                count++;
                System.out.println(i);
            }
            System.out.println("*******************************");
```

```
        System.out.println(count++ +" whole numbers are between "+number2+" and
"+number1);
      }
   }
}
```

```
public class DeclarationsNumber {


    public static void main(String[] args){

        int number1;
        int number2;
        int count = 0;

        System.out.println("enter number 1");
        Scanner num1 = new Scanner(System.in);
        number1 = num1.nextInt();
        System.out.println("******************************");
        System.out.println("enter number 2");
        Scanner num2 = new Scanner(System.in);
        number2 = num2.nextInt();

        if(number1 <= number2){
            System.out.println("******************************");
            System.out.println("those numbers are: \n");
            for(int i = number1 ; i <= number2 ; i++ ){

                count++;
                System.out.println(i);
            }

            System.out.println("******************************");
            System.out.println(count++ +" whole numbers are between "+number1+" and "+number2);

        }else {
            System.out.println("******************************");
            System.out.println("those numbers are: \n");
            for (int i = number2 ; i <= number1 ; i++){

                count++;
                System.out.println(i);
            }
            System.out.println("******************************");
            System.out.println(count++ +" whole numbers are between "+number2+" and "+number1);
        }
    }
}
```

```
compile:
run:
enter number 1
8
*******************************
enter number 2
1
*******************************
those numbers are:

1
2
3
4
5
6
7
8
*******************************
8 whole numbers are between 1 and 8
BUILD SUCCESSFUL (total time: 4 seconds)
```

2. Some attributes of a song object. What attributes would you like to have for an object that represents a playlist that contains many songs?

myPlaylist
-playlistname : String -creationdate : int -playlistowner : String -songnumber : int -song : Song -yourfavorite : bool

3. What behaviors can a song have? What behaviors can a playlist have? Compare the difference in behavior between the two types of objects.

myPlaylist
-playlistname : String -creationdate : int -playlistowner : String -songnumber : int -song : Song -yourfavorite : bool
+removesong( ) +lookfor() +sharelist (copyright : Copyright) : String +modifylist( ) +playlisttime( ) : float

Song
-name : String -gender : Gender[ ] -age : int -artist : String -album : Album[ ] -concerts : String -lyrics : String -time : float -likes : bool
+play( ) +pause( ) +repeat( ) +add( ) +lookfor( ) +sortout( ) +sharesong (copyright : Copyright) : String

4. What attributes and behaviors would an object representing a credit card account have?

CardAccount

-name: String -date : int -imail : String -firm : String -password : int -direction : String -id : int
+transfer ( ) +takemoney ( ) +login ( ) +modifydata( ) +savemoney( ) +support( ) +registry( )

5. Write a program that reads a string from the keyboard and tests for a valid date. Displays the date and a message indicating whether it is valid. If it is invalid, it also displays a message explaining why it is invalid.

```
package ec.edu.espe.datesimulator.view;
import java.util.Scanner;
import ec.edu.espe.datesimulator.model.Date;

public class DateSimilator {

    public static void main(String[] args){

        Date date;
        date = new Date();

        System.out.println("enter current date");
        System.out.println("******************************");
        System.out.println("day: \n");
        Scanner day = new Scanner(System.in);
        date.day = day.nextInt();
        System.out.println("******************************");
        System.out.println("month: \n");
        Scanner month = new Scanner(System.in);
        date.month = month.nextLine();
        System.out.println("******************************");
        System.out.println("month: \n");
        Scanner year = new Scanner(System.in);
        date.year = year.nextInt();
        System.out.println("******************************");
        System.out.println("The date entered is ---> "+date);

        Date date2;
```

```
date2 = new Date();


if (date.getDay() == date2.getDay()&& date.getMonth()==date2.getMonth()&&
date.getYear()==date2.getYear()){

    System.out.println("the date is valid" );

}else{
    System.out.println("the date is not valid" );
    System.out.println("The current date is ----> "+date2)
}
```

```
Date date;
date = new Date();

System.out.println("enter current date");
System.out.println("*******************************");
System.out.println("day: \n");
Scanner day = new Scanner(System.in);
date.day = day.nextInt();
System.out.println("*******************************");
System.out.println("month: \n");
Scanner month = new Scanner(System.in);
date.month = month.nextLine();
System.out.println("*******************************");
System.out.println("month: \n");
Scanner year = new Scanner(System.in);
date.year = year.nextInt();
System.out.println("*******************************");
System.out.println("The date entered is ---> "+date);

Date date2;
date2 = new Date();

if (date.getDay() == date2.getDay()&& date.getMonth()==date2.getMonth()&& date.getYear()==date2.getYear()){

    System.out.println("the date is valid" );
}else{
    System.out.println("the date is not valid" );
    System.out.println("The current date is ----> "+date2);
```

```java
package ec.edu.espe.datesimulator.model;

/**
 *
 * @author luist
 */
public class Date {

    public int day;
    public String month;
    public int year;

    public Date(){
        this.day = 10;
        this.month = "january";
        this.year = 2021;
    }

    @Override
    public String toString() {
        return "Date{" + "day=" + day + ", month=" + month + ", year=" + year + '}';
    }
```

```
enter current date
*******************************
day:

10
*******************************
month:

january
*******************************
month:

2020
*******************************
The date entered is ---> Date{day=10, month=january, year=2020}
the date is not valid
The current date is ----> Date{day=10, month=january, year=2021}
BUILD SUCCESSFUL (total time: 13 seconds)
```

6. When does Java automatically provide a default constructor and when does it not?

Java automatically provides a constructor when the object properties are initialized by individual methods(getters) after instantiation.
Java does not automatically provide a constructor when a constructor has been created in the class with the default methods.

7. What is the difference between a static variable and an instance variable?
An instance variable is replicated in each class instance and maintains its independent value in each object. A static is global for all instances and its modification affects them all. Installation variable is equal to a local scope. Static variable is equal to global scope. A static variable is accessible without the need to create an instance.

8. Can a class contain both instance variables and static methods?
Yes, each object in the class has its own instance variables. A class has only one of each static variable and all objects share the static variable.

9. Can you refer to a static variable by name within a static method definition without using a class name and a period?
No, you cannot refer to an instance variable within the definition of a static method, because a static method can be invoked without any objects, since there are no instance variables.

10. Can a class contain static and non-static methods?
Yes, since a static method belongs to the class and not to the object, and can only access static data, It can contain non-static methods but not call them

11. Can you invoke a static method inside a non-static method?

No, because the instance methods operate the instance variables of the objects and are in the static method, although sometimes it is not necessary to create the object

12. Can a package have any name you want, or are there restrictions on what you can use for a package name? Explain the restrictions.

- All package names are lowercase
- The proper name of a package is the domain of your company in reverse form
- Packages published by different authors must have different names

13. What does it mean when we say that an event is "sent" to a listener object?

Event, they are actions that a user can perform and when the event is performed, actions occur, the listeners control the events, wait for the event to occur

14. Provide the definition of a static method named showArray that has an array of base type char as a single parameter and that writes a line of text to the screen consisting of the characters in the array argument written in order

the method called Show Array has a char datatype for the array, but you can't input, you can only input data that is static

A static method called show array that has an array of type base char as a single parameter, you cannot enter the range of your array as static type data is the only one that can be entered because each value is respectively already declared and its size will not change since it is a static method.

15. Provide the definition of a static method called getArrayValues that has an array of type base double as its only parameter and that returns another array whose base type and length are the same as the parameter, but whose elements have each been divided by 2

get Array Values array has a double array cannot be resized because it has a get, but it can be changed in another variable with the same data type

16. What is the difference between overriding a method and overloading the name of a method?

The override, the subclass method overrides the method in the main class and the overload creates several methods with the same name but with a different list of parameter types

17. Can a derived class directly access a private instance variable of the base class by name?

You can't with private, but you can when we put extends in the method (extends: creates derived classes and not a subclass)

18. Can a derived class directly invoke a private method of the base class?

It can be done, but we can't access it without first implementing a get

19. Can an object be referenced by variables of several different data types? Explain.

Object reference arrays can contain objects of different classes and invoking a method in a reference will produce the appropriate behavior depending on the actual type of the referenced object. When this happens at runtime, it is called dynamic or late assignment.

20. Describe two uses of the keyword super.

The most common use of the keyword **super** is to eliminate confusion between superclasses and subclasses that have methods with the same name.
It can be used to access the parent class constructor. **Super** can call constructors both with and without parameters depending on the situation.

21. What is the difference between this and super when these words are used within a constructor definition as the names of the methods that are called?

When accessing instance variables of a class, the keyword **this** refers to the members of the class itself in the current object, in other words, **this** refers to the current object on which a given method is acting and is used whenever you want to refer to the current object of the class.
To call the parent method within a class that has replaced that method, the parent method is referenced with the **super** keyword. **Super** is used to refer to a local variable in a method and a variable in the superclass that has the same name, **super** is also used to invoke the superclass constructor from the subclass constructor.

22. Can you derive an exception class from the predefined class IOException, or should it derive from the Exception class?

Exceptions are not a class, but there are several types of exceptions that are derived from the **IOException** class, such as **RuntimeException**, which represents exceptions defined automatically by programs, such as: division by 0, invalid matrix index, among others.

23. What happens if you invoke a method and throw a checked exception that it doesn't catch?
If no particular action is planned for the treatment of a certain exception, it is possible to propagate the exception without capturing it, leaving it to other parts of the program to define the actions for its treatment (**throws**).
The exception is not treated by the main() method and the program aborts its execution.

24. Consider an invocation of method A. Suppose method A calls method B and method B calls method C. If method C throws an exception that it does not catch on its own, where could the exception be caught? In B? In a? Outside of A?

Assuming that method C launches an exception that it does not handle, it exits the call stack to method B, while it does not handle such exception, it exits and goes to method A. Finally, if method A does not control the exception, the higher level method main() ends the program.
In conclusion, the exception is trapped outside the A method, as is the case with main.

25. Can you have a try block and corresponding catch blocks inside a bigger try block?
Yes, when a try-catch block is present inside another try block, it is called a nested try catch block.

26. What is an iterator for a collection of elements, such as an array or a linked list?

An iterator refers to the object that allows the programmer to loop through a container (a collection of items) particularly list

a call to the collection method of the iterator allows you to get a two-way iterator that can traverse a list object backwards or forwards

27. Write code that will create a stream called outStream that is an object of the PrintWriter class and that connects this stream to a text file called sam.txt so that your program can send the output to the file. If the sam.txt file does not exist, create a new, empty file. However, if a file named sam.txt already exists, delete its old content so that the program can start with an empty file of that name.

```
File file = new File("sam.txt");
    String fileName;
    fileName = "sam.txt";
    PrintWriter outStream;
    outStream = null;

        if (!file.exists()) {
            try {
                file.createNewFile();
                outStream = new PrintWriter(new FileOutputStream(fileName, true));{
                catch (IOException ex){
            }
        }else {
            try{
                outStream = new PrintWriter(new FileOutputStream(fileName, true));
                }catch (IOException ex){
                    }
    }
```

28. What kind of exception could the following statement raise and what would be indicated if this exception were thrown? PrintWriter outputStream = new PrintWriter ("out.txt");

File Not Found Exception

29. Write code that creates a stream called textStream that is an object of the PrintWriter class and that connects the stream to a text file called dobedo so that your program can send the output to this file.

```java

package textstream;

import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 *
 * @author ERIKA
 */
public class TextStream {

    public static void main(String[] args) {

            File dobedo = new File("dobedo.txt");
        // TODO code application logic here
        try{
            PrintWriter textStream = new PrintWriter(new FileWriter(dobedo,true));
        }catch(IOException e ){

        }

    }
}

```

30. What does it mean to serialize? How can a class be serialized?

What does it mean to serialize: It is the process where an object can be represented as a sequence of bytes, which includes the data of the object
such as the target type information and the types of data stored in the object.
How can a class be serialized?: During serialization, the attributes, class name, and its assembly are converted to a sequence of bytes,
In order to serialize an object, its class must be declared as [Serializable].