**BIRLA INSTITIUTE OF TECHNOLOGY AND SCIENCE, PILANI**
**Second Semester 2019-20**
**Object-Oriented Programming [CS F213]**

**LAB-1 [Introduction to Java Basics]**

1. **OOPS introduction using Java**
2. **Hello, world! in Java**
3. **Command line arguments as input**
4. **Exercises (Write Your own Java code)**

**1.** In structured languages such as C, we create data structures and use functions to manipulate the data structures. In OOP languages such as Java, we create objects that contain instance fields (state) and behaviour (methods). So, each object in Java is described in terms of its state and behaviour. A class is a blueprint or template for an object. From the class, we create objects (called instances). These objects have the state and behaviour as defined in the blueprint (class). A class is usually a named entity, such as: Person, Employee, Account, etc.
Think of some instance fields and methods of the above named entities.

**2.** Hello, world! program
Save the following code in a file **Hello.java**

```
public class Hello
{
    public static void main (String[] args) {
        System.out.println("Hello, world!");
     }
}
```

Notice how the name of the public class Hello is same as the file name. It is a requirement - remember this.

Compile the program using:
**`javac Hello.java`**

The above command compiles the java program into byte code and saves it as Hello.class.

Run the byte code (saved in Hello.class) using:
**`java Hello`**

The output `Hello, world!` will be displayed on the screen.

You might have noticed a few keywords in the above program such as *public* and *static*.
The *public* keyword is called an access specifier. In the above code, it means that the class *Hello* and method *main()* can be accessed by any other class. The *static* keyword before *main()* above means that the method can be called without creating an instance (object) of the class. We will learn more about these keywords later.

## 3. Command line arguments

The arguments to `main()` method are given as an array of strings when the command is given to run the byte code. Hence they are called command line arguments. (If you remember C's `argv` to `main()`, these are similar). An example of command line argument is:

```
java Hello 10 20
```

The command line argument above is an array of two strings: ["10", "20"]. Java arrays also start with index 0. In this example, args[0] is "10" and args[1] is "20".

Note that if you want to use the command line arguments as integers, you need to convert them from strings to ints. A way of doing this is:

`Integer.parseInt(args[0])`. (Again, it is analogous to C's `atoi()`)

The length of args array can be computed using `args.length`.

## 4. Exercises

Exercise-1: Write a program that takes as input a positive integer N as command line argument. It prints all numbers from 1 to N. It also prints all odd, even and prime numbers between 1 and N.

Exercise-2: Write a program that takes an array of integers as input and displays their sum.

Exercise-3: Write a program called **Fibonacci.java** to display the first N Fibonacci numbers.

F(n), where F(n)=F(n–1)+F(n–2) and F(1)=F(2)=1.

Also compute their average. [ **Note:** the value N is an integer and should be passed as a command line argument]

A sample run is:

```
javac Fibonacci.java
java Fibonacci
The first 20 Fibonacci numbers are:
1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181
6765
The average is 885.5
```

Exercise-4: Write a Java program called **SumDigits.java** to sum up the individual digits of a positive integer, given in the command line.

A sample run is:

```
javac SumDigits.java
java SumDigits 12345
The sum of digits = 1 + 2 + 3 + 4 + 5 = 15
```