EXPERIMENT 2

// Introduction to Programming in Java

PROGRAM 1

// This is an introductory program in Java.

// Source Code (Intro.java) -

```
class Intro
{
	public static void main(String args[ ])
	{
		System.out.println("This is an introductory program in Java.");
		System.out.print("So, ");
		System.out.print("let's begin!");
	}
}
```

Output -

This is an introductory program in Java.
So, let's begin!

Learnings -

- ★ Every source code in Java is written in the form of a class. This ensures that every Java program has object-oriented characteristics.
  - A **class** is a user-defined data type that contains the specifications of the **data members** (**attributes**) and the **member functions** (**methods**) of an object.
    An **object** is an instance of a class i.e., it is the basic run-time entity that works as per the specifications made for it in its class.
    - The data members specified for an object can be accessed only by its member functions.
    - However, different objects can communicate with each other by sending messages to each other using their functions.

    A class is just a blueprint for an object. So it does not get stored in the memory. But an object is its run-time instance, so as soon as it is declared with a class, memory gets allocated for it as per the specifications (data members & member functions) made in the class.

★ By convention, the name of a source file of Java should be the same as the name of the main class in which the entire source code is written.
    - This is so because, when a Java source code is compiled, a bytecode file is generated with the same name as that of the main class of the code, with the '**.class**' extension. This bytecode file is then picked up by the JVM for execution in the system. So keeping the names of source file and main class the same prevents any confusion regarding what is actually being implemented by the JVM.

★ Inside the main class, a **main()** function has to be created, and all the instructions to be executed by the system are written in the body of this function.
    - This is because execution of a Java program begins with a call to the main() by JVM. Thus, main() is the first function to be executed in a Java program and a call to this function indicates that instructions written in it are to be executed by the system.
        - main() in Java has the following prototype:

          public static void main(String args[ ])

          Here,
            - ➔ **void**, a return type, indicates that the main() does not return any value.
            - ➔ **static**, a keyword, enables the main() to be called without creating an object of the main class by JVM. This is necessary since main() is called by the JVM before creating any objects of the main class.
            - ➔ **public**, an access specifier, indicates that the main() can be called by the code written outside the main class, such as by JVM.
            - ➔ **String args[ ]** is an array declaration that indicates that an array of objects of type String is passed as an argument to the main(). This is done to accept any command-line arguments generated during execution of the program.

★ All Java programs automatically import the **java.lang** package which defines a class called **System**. This class contains several aspects of the run-time environment. For example, it helps in performing input operations, output operations, error handling operations, etc.
    - From System class, we get a variable '**out**' which is associated with the output stream and helps in displaying an output on the console.
      For displaying output on the console, this variable calls it's functions like -

        - **println() -** This function prints an output on the console and also inserts a new line after it.
        - **print() -** This function can also display an output on the console but it does not insert a new line after it.
    - The **dot operator / period (.)** is used to make calls to the out variable and it's functions from the System class.