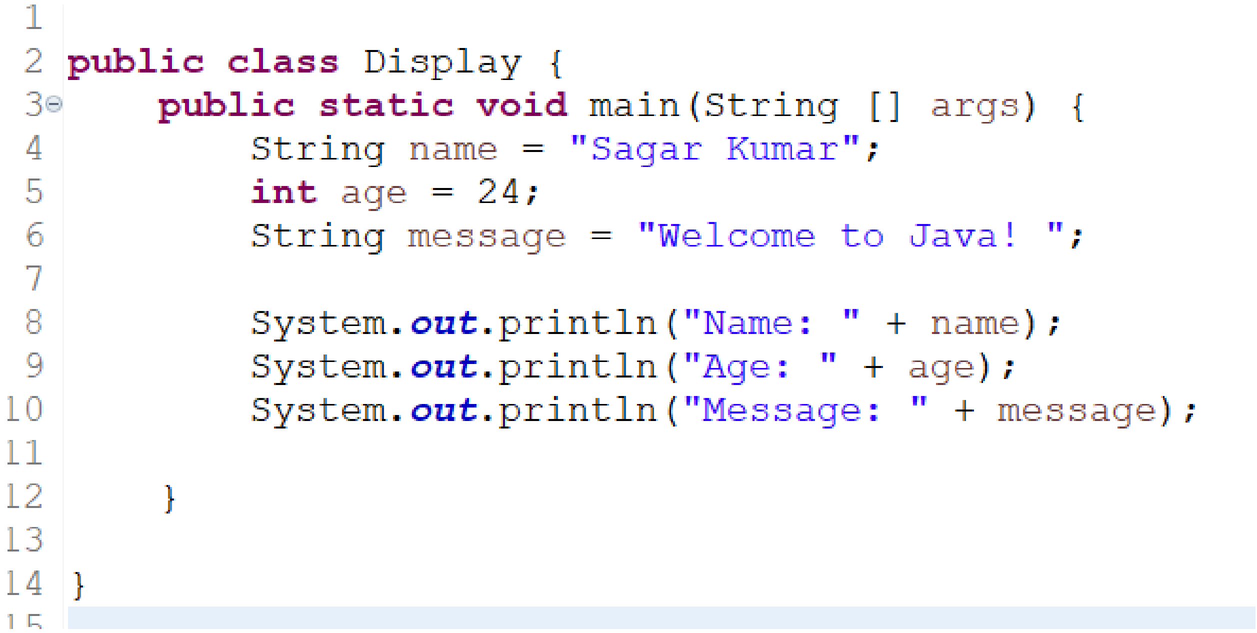
ASSIGNMENT-1

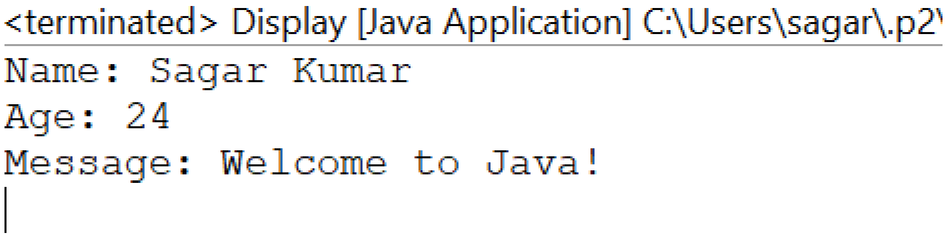
Q1) Write a program to display your name, age, and a short message using System.out.println.

ANS -



```java

public class Display {
    public static void main(String [] args) {
        String name = "Sagar Kumar";
        int age = 24;
        String message = "Welcome to Java! ";

        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
        System.out.println("Message: " + message);

    }

}
```

Output:



```
<terminated> Display [Java Application] C:\Users\sagar\.p2
Name: Sagar Kumar
Age: 24
Message: Welcome to Java!
```

Q2) Explain the steps of Java compilation and execution in your own words.

Ans - Steps -

1) I write my Java program in a text editor or an IDE (like Eclipse or VS Code) and save it with a `.java` extension.

Example:

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

     }
```

Compilation

2) The `.java` file (source code) is converted into a `.class` file (bytecode) using the **Java Compiler (javac)**.
3) Bytecode is a platform-independent code that can run on any system with a Java Virtual Machine (JVM).
4) Open a terminal or command prompt.
5) Navigate to the folder where my `.java` file is saved.
6) Type:
   `javac HelloWorld.java`
7) This creates a `HelloWorld.class` file in the same folder.

Execution

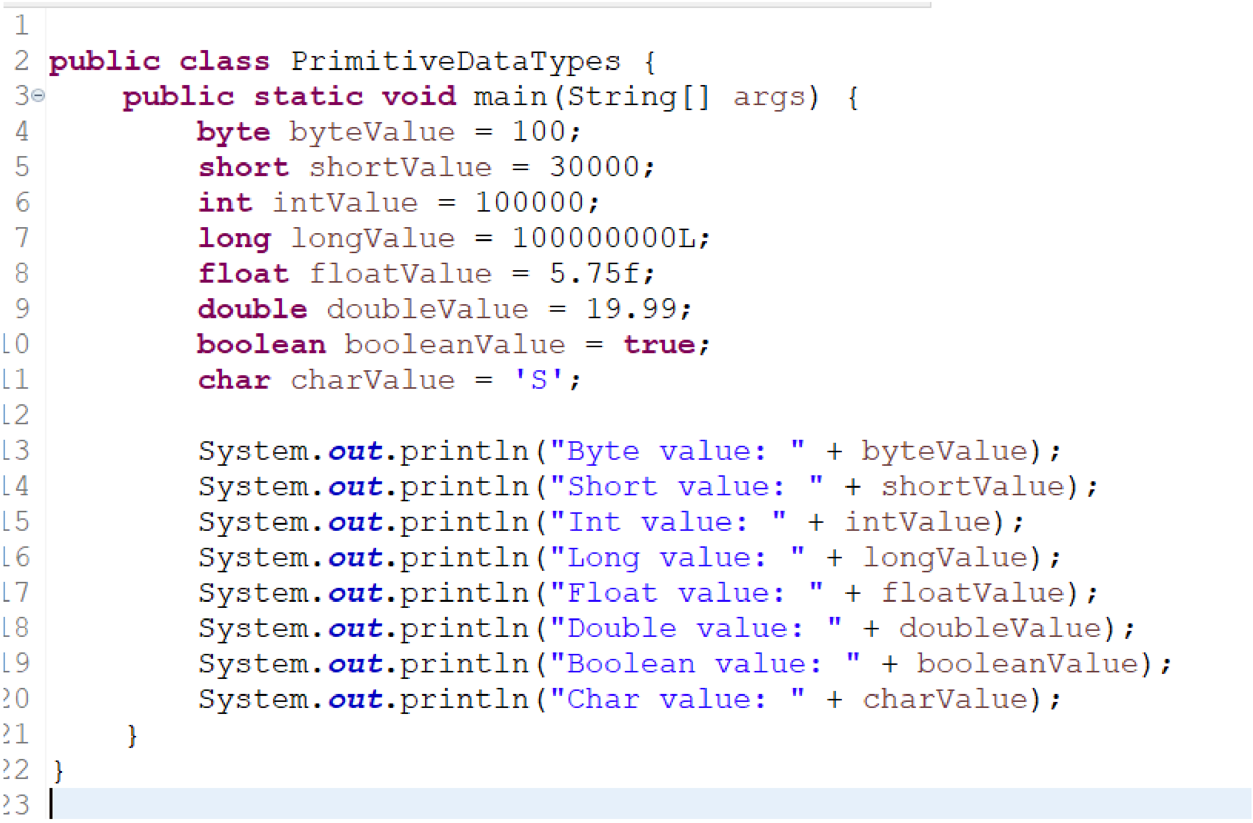
8) The JVM (Java Virtual Machine) reads the bytecode in the `.class` file and translates it into machine code (specific to my computer) so the program can run.
9) In the terminal or command prompt, type:
   java HelloWorld

10) The program runs, and I’ll see:

```
Hello, World!
```

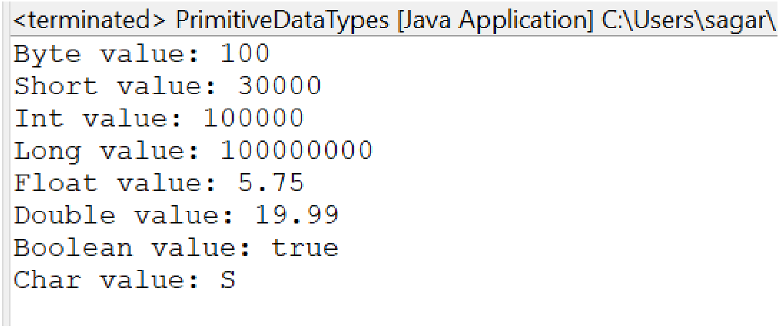
Q3) Create a program that demonstrates all the primitive data types in Java, assigning values and printing them.

ANS -



```java

public class PrimitiveDataTypes {
    public static void main(String[] args) {
        byte byteValue = 100;
        short shortValue = 30000;
        int intValue = 100000;
        long longValue = 100000000L;
        float floatValue = 5.75f;
        double doubleValue = 19.99;
        boolean booleanValue = true;
        char charValue = 'S';

        System.out.println("Byte value: " + byteValue);
        System.out.println("Short value: " + shortValue);
        System.out.println("Int value: " + intValue);
        System.out.println("Long value: " + longValue);
        System.out.println("Float value: " + floatValue);
        System.out.println("Double value: " + doubleValue);
        System.out.println("Boolean value: " + booleanValue);
        System.out.println("Char value: " + charValue);
    }
}

```

Output:



```
<terminated> PrimitiveDataTypes [Java Application] C:\Users\sagar\
Byte value: 100
Short value: 30000
Int value: 100000
Long value: 100000000
Float value: 5.75
Double value: 19.99
Boolean value: true
Char value: S
```

Q4) Draw a diagram explaining how Java achieves platform independence.

Ans-

```
     Java Code (.java)
     |
     V
JAVAC compiler
     |
     V
Byte Code (.class)
     |
    /|\
   / | \
  V  V  V
 JVM  JVM  JVM
  |   |   |
  V   V   V
Windows Linux Mac
```

1.Write Once:

- Developers write Java source code (.java files)
- This code is human-readable
- Same source code works across all platforms

2.Compile Once:

- Java compiler (javac) converts source code into bytecode
- Creates .class files
- Bytecode is platform-independent
- Can be moved to any platform

3.Run Anywhere:

- JVM installed on different platforms (Windows, Linux, Mac)

- JVM reads bytecode
- Converts bytecode to platform-specific machine code
- Executes the program

This "Write Once, Run Anywhere" principle is what makes Java platform-independent. The JVM acts as an abstraction layer between the Java bytecode and the underlying operating system, handling all platform-specific operations.

Q5) Research and explain the difference between float and double in Java.

Ans -

In Java, `float` and `double` are used to store numbers with decimal points. But there are some differences between them:

Precision:

- **`float`**: Can handle about 7 digits (e.g., 3.141592). If you try to store a longer number, it might round off the extra digits.
- **`double`**: Can handle about 15–16 digits (e.g., 3.141592653589793). It's more accurate for bigger numbers or numbers with lots of decimal places.

Size:

- **`float`:** Takes up 4 bytes (smaller).
- **`double`:** Takes up 8 bytes (bigger)**.**

Range:

- **`float`:** Can store very small numbers (like 0.000000034) or very big numbers (like 34,000,000,000).
- **`double`:** Can store even smaller or bigger numbers, making it more flexible**.**

Performance:

- **`float`:** A bit faster and uses less memory, so it’s good for things like games or graphics where exact precision isn’t super important.
- **`double`:** A bit slower and uses more memory, but it’s more accurate, so it’s better for scientific calculations or anything requiring high precision.

`float` is smaller and faster, `double` is bigger and more accurate.