



Experiment No 2

Bhuvi Ghosh
60009210191

Aim: - Implement Classes, Object and Packages in Java

Theory: -

1. Class and Object in Java

I. Java Class

A class is a blueprint for the object. Before we create an object, we first need to define the class. We can think of the class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object. Since many houses can be made from the same description, we can create many objects from a class

Create a class in Java

We can create a class in Java using the class keyword. For example,

```
class ClassName {  
    // fields  
    // methods  
}
```

Here, **fields (variables)** and **methods** represent the **state** and **behavior** of the object respectively.

- **fields** are used to **store data**
- **methods** are used to **perform some operations**

For our **bicycle** object, we can create the class as

```
class Bicycle {  
    // state or field  
    private int gear = 5;  
    // behavior or method  
    public void braking () {  
        System.out.println("Working of Braking");  
    }  
}
```

In the above example, we have created a class named **Bicycle**. It contains a field named **gear** and a method named **braking ()**.

Here, **Bicycle** is a prototype. Now, we can create any number of bicycles using the prototype.

And, all the bicycles will share the fields and methods of the prototype.



Object

An entity that has state and behavior is known as an object e.g., chair, bike, marker, pen, table, car, etc. It can be physical or logical (tangible and intangible). The example of an intangible object is the banking system.

An object has three characteristics:

- **State:** represents the data (value) of an object.
- **Behavior:** represents the behavior (functionality) of an object such as deposit, withdraw, etc.
- **Identity:** An object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. However, it is used internally by the JVM to identify each object uniquely.

For Example, Pen is an object. Its name is Reynolds; color is white, known as its state. It is used to write, so writing is its behavior.

An object is an instance of a class. A class is a template or blueprint from which objects are created. So, an object is the instance(result) of a class.

Object Definitions:

- An object is a real-world entity.
- An object is a runtime entity.
- The object is an entity which has state and behavior.
- The object is an instance of a class.

Creating an Object in Java

Here is how we can create an object of a class.

Syntax:

```
className object = new className ();
```

//Example for Bicycle class

```
Bicycle sportsBicycle = new Bicycle();
```

```
Bicycle touringBicycle = new Bicycle();
```

We have used the new keyword along with the constructor of the class to create an object. Constructors are similar to methods and have the same name as the class. For example, Bicycle () is the constructor of the Bicycle class.

Here, sportsBicycle and touringBicycle are the names of objects. We can use them to access fields and methods of the class.

Access Members of a Class



We can use the name of objects along with the • (dot) operator to access members of a class.

For example,

```
class Bicycle {
```

```
    // field of class
```

```
    int gear = 5;
```

```
    // method of class
```

```
    void braking () {
```

```
        ...
```

```
    }
```

```
}
```

```
// create object
```

```
Bicycle sportsBicycle = new Bicycle ();
```

```
// access field and method
```

```
sportsBicycle.gear;
```

```
sportsBicycle.braking();
```

In the above example, we have created a class named Bicycle. It includes a field named gear and a method named braking (). Notice the statement,

```
Bicycle sportsBicycle = new Bicycle ();
```

Here, we have created an object of Bicycle named sportsBicycle. We then use the object to access the field and method of the class.

sportsBicycle.gear - access the field gear

sportsBicycle.braking() - access the method braking ()

2. Java Methods

A method is a block of code that performs a specific task. Suppose you need to create a program to create a circle and color it. You can create two methods to solve this problem:

a method to draw the circle

a method to color the circle

Dividing a complex problem into smaller chunks makes your program easy to understand and reusable.

In Java, there are two types of methods:

User-defined Methods: We can create our own method based on our requirements.

Standard Library Methods: These are built-in methods in Java that are available to use.



Declaring a Java Method

The syntax to declare a method is:

```
returnType methodName () {  
    // method body  
}
```

Here,

returnType - It specifies what type of value a method returns for example if a method has an int return type then it returns an integer value.

If the method does not return a value, its return type is void.

methodName - It is an identifier that is used to refer to the particular method in a program.

method body - It includes the programming statements that are used to perform some tasks. The method body is enclosed inside the curly braces { }.

For example,

```
int addNumbers () {  
    // code  
}
```

In the above example, the name of the method is addNumbers (). And, the return type is int.

3. Java Constructors

A constructor in Java is similar to a method that is invoked when an object of the class is created. Unlike Java methods, a constructor has the same name as that of the class and does not have any return type. For example,

```
class Test {  
    Test () {  
        // constructor body  
    }  
}
```

Here, Test () is a constructor. It has the same name as that of the class and doesn't have a return type.

Types of Constructor

In Java, constructors can be divided into 3 types:

- No-Arg Constructor
- Parameterized Constructor
- Default Constructor

Java No-Arg Constructors

Similar to methods, a Java constructor may or may not have any parameters (arguments).

If a constructor does not accept any parameters, it is known as a no-argument constructor. For example,

```
private Constructor () {  
    // body of the constructor  
}
```



Example

```
class Main {
    int i;
    // constructor with no parameter
    private Main () {
        i = 5;
        System.out.println("Constructor is called");
    }

    public static void main (String [] args) {

        // calling the constructor without any parameter
        Main obj = new Main ();
        System.out.println("Value of i: " + obj.i);
    }
}
```

Note:

Once a constructor is declared private, it cannot be accessed from outside the class. So, creating objects from outside the class is prohibited using the private constructor.

Java Parameterized Constructor

A Java constructor can also accept one or more parameters. Such constructors are known as parameterized constructors (constructor with parameters).

Example

```
class Main {

    String languages;

    // constructor accepting single value
    Main (String lang) {
        languages = lang;
        System.out.println(languages + " Programming Language");
    }

    public static void main (String [] args) {

        // call constructor by passing a single value
        Main obj1 = new Main("Java");
        Main obj2 = new Main("Python");
        Main obj3 = new Main("C");
    }
}
```

Java Default Constructor

If we do not create any constructor, the Java compiler automatically create a no-arg constructor during the execution of the program. This constructor is called default constructor.

Example

```
class Main {
    int a;
    boolean b;
```



```
public static void main (String [] args) {
```

```
    // A default constructor is called  
    Main obj = new Main ();  
    System.out.println("Default Value:");  
    System.out.println("a = " + obj.a);  
    System.out.println("b = " + obj.b);  
}  
}
```

4. Java Package

A package is simply a container that groups related types (Java classes, interfaces, enumerations, and annotations). For example, in core Java, the ResultSet interface belongs to the java.sql package. The package contains all the related types that are needed for the SQL query and database connection.

Based on whether the package is defined by the user or not, packages are divided into two categories:

- Built-in Package
- User-defined Package

Built-in Package

Built-in packages are existing java packages that come along with the JDK. For example, java.lang, java.util, java.io, etc. For example:

```
import java.util.ArrayList;  
class ArrayListUtilization {  
    public static void main(String[] args) {  
        ArrayList<Integer> myList = new ArrayList<>(3);  
        myList.add(3);  
        myList.add(2);  
        myList.add(1);  
  
        System.out.println(myList);  
    }  
}
```

Output:

```
myList = [3, 2, 1]
```

The ArrayList class belongs to java.util package. To use it, we have to import the package first using the import statement.

```
import java.util.ArrayList;
```

User-defined Package

Java also allows you to create packages as per your need. These packages are called user-defined packages.

How to define a Java package?

To define a package in Java, you use the keyword package.

```
package packageName;
```



Java uses file system directories to store packages. Let's create a Java file inside another directory.

For example:

```
└── com
    └── test
        └── Test.java
```

Now, edit Test.java file, and at the beginning of the file, write the package statement as:

```
package com.test;
```

Here, any class that is declared within the test directory belongs to the com.test package.

Here's the code:

```
package com.test;
class Test {
    public static void main(String [] args){
        System.out.println("Hello World!");
    }
}
```

Output:

Hello World!

Here, the Test class now belongs to the com.test package.

Package Naming convention

The package name must be unique (like a domain name). Hence, there's a convention to create a package as a domain name, but in reverse order. For example, com.company.name

Here, each level of the package is a directory in your file system. Like this:

```
└── com
    └── company
        └── name
```

And, there is no limitation on how many subdirectories (package hierarchy) you can create.

5. Lab Assignments to complete in this session

- I. Write a Java program to create a user-defined package and function to print a message for the users and import the same package in another program.
- II. Write a java program to create a user-defined package letmecalculate having class calculator and functions addition, subtraction, multiplication, division. Import this package in another program to use the class calculator.
- III. Write a constructor in the Car class given below that initializes the brand class field with the string "Ford". Call the getBrand () method in the main method of the Sample class and store the value of the brand in a variable, and print the value.
- IV. Write a program to print the names of students by creating a Student class. If no name is passed while creating an object of Student class, then the name should be "Unknown",



otherwise the name should be equal to the String value passed while creating object of Student class.

- V. Write a Java class Complex for dealing with complex number. Your class must have the following features:

Instance variables:

- **realPart** for the real part of type double
- **imaginaryPart** for imaginary part of type double.

Constructor:

- **public Complex ()**: A default constructor, it should initialize the number to 0, 0)
- **public Complex (double realPart, double imaginaryPart)**: A constructor with parameters, it creates the complex object by setting the two fields to the passed values.

Instance methods:

- **public void setRealPart (double realPart)**: Used to set the real part of this complex number.
- **public void setImaginaryPart (double realPart)**: Used to set the imaginary part of this complex number.
- **public double getRealPart ()**: This method returns the real part of the complex number
- **public double getImaginaryPart ()**: This method returns the imaginary part of the complex number

Write a separate class **ComplexDemo** with a main () method and test the Complex class methods.

- VI. Create a class named 'Student' with String variable 'name' and integer variable 'roll_no'. Assign the value of roll_no as '2' and that of name as "John" by creating an object of the class Student.

Q1)

The screenshot shows the IntelliJ IDEA interface. On the left, the Project tool window displays the file structure: `~/IdeaProjects/j` with subfolders `.idea`, `out`, and `src`. The `src` folder contains files `Java5`, `Main`, `Main2`, `Main3`, `Main4`, `.gitignore`, and `j.iml`. The top editor pane shows the `Main4.java` file with the following code:

```
1  import example.Java5;
2  class Main4{
3      public static void main(String[] args)
4      {
5          Java5 j= new Java5();
6          j.show();
7      }
8  }
```

The bottom editor pane shows the `Java5.java` file with the following code:

```
1  package example;
2  public class Java5 {
3      no usages
4      private void show1() { System.out.println("This is a string"); }
5      2 usages
6      public void show() { System.out.println("This is a string"); }
7      public static void main(String[] args)
8      {
9          Java5 obj=new Java5();
10         obj.show();
11     }
12 }
```

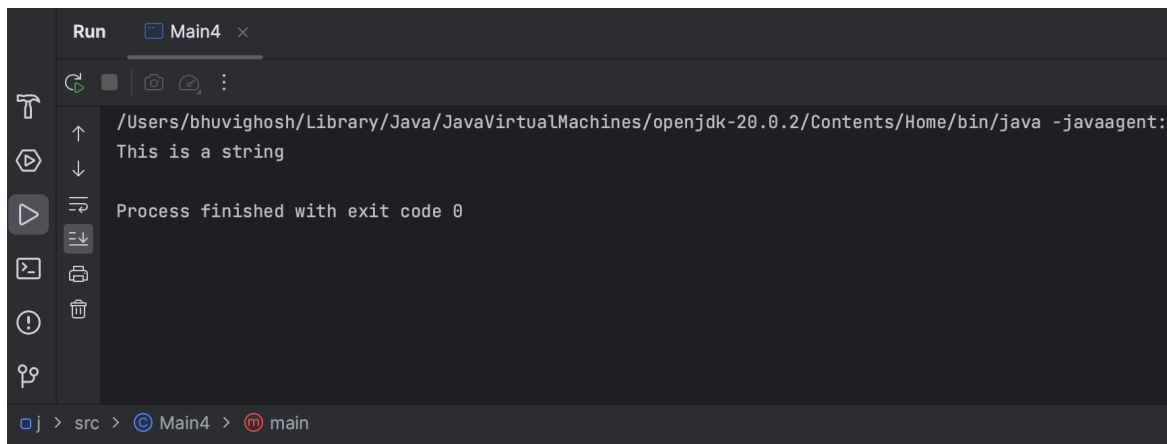
The bottom status bar shows the command prompt output:

```
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.app/Contents/Li
This is a string

Process finished with exit code 0
```

The screenshot shows the IntelliJ IDEA interface with the `Main4.java` file open in the editor. The code is as follows:

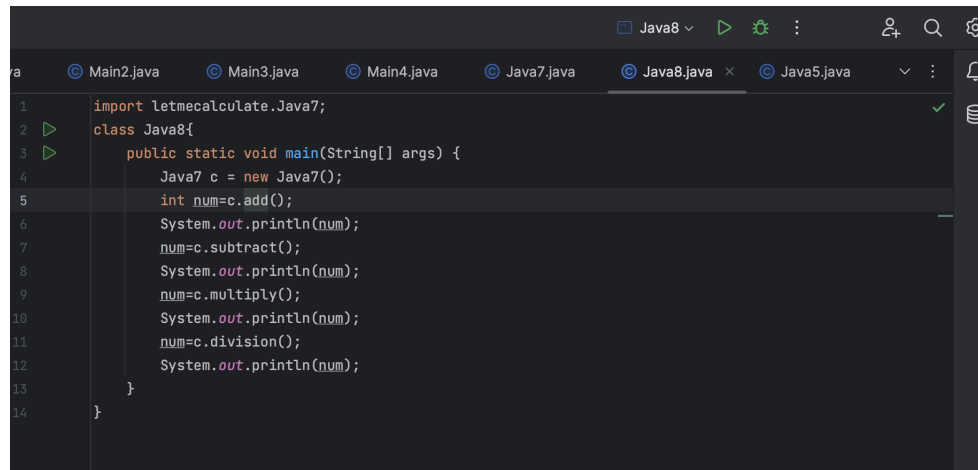
```
1  import example.Java5;
2  class Main4{
3      public static void main(String[] args)
4      {
5          Java5 j= new Java5();
6          j.show();
7      }
8  }
```



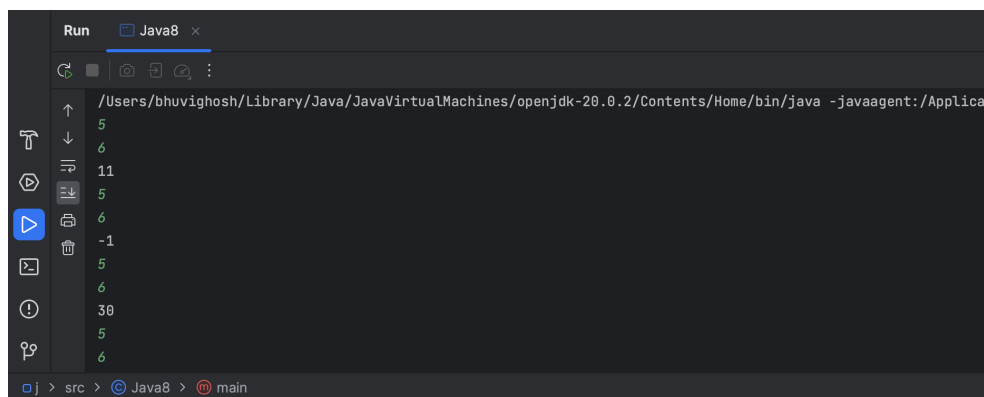
```
Run Main4 x
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:
This is a string
Process finished with exit code 0
```

j > src > Main4 > main

Q2)



```
Java8.java
import letmecalculate.Java7;
class Java8{
    public static void main(String[] args) {
        Java7 c = new Java7();
        int num=c.add();
        System.out.println(num);
        num=c.subtract();
        System.out.println(num);
        num=c.multiply();
        System.out.println(num);
        num=c.division();
        System.out.println(num);
    }
}
```



```
Run Java8 x
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:/Applica
5
6
11
5
6
-1
5
6
30
5
6
Process finished with exit code 0
```

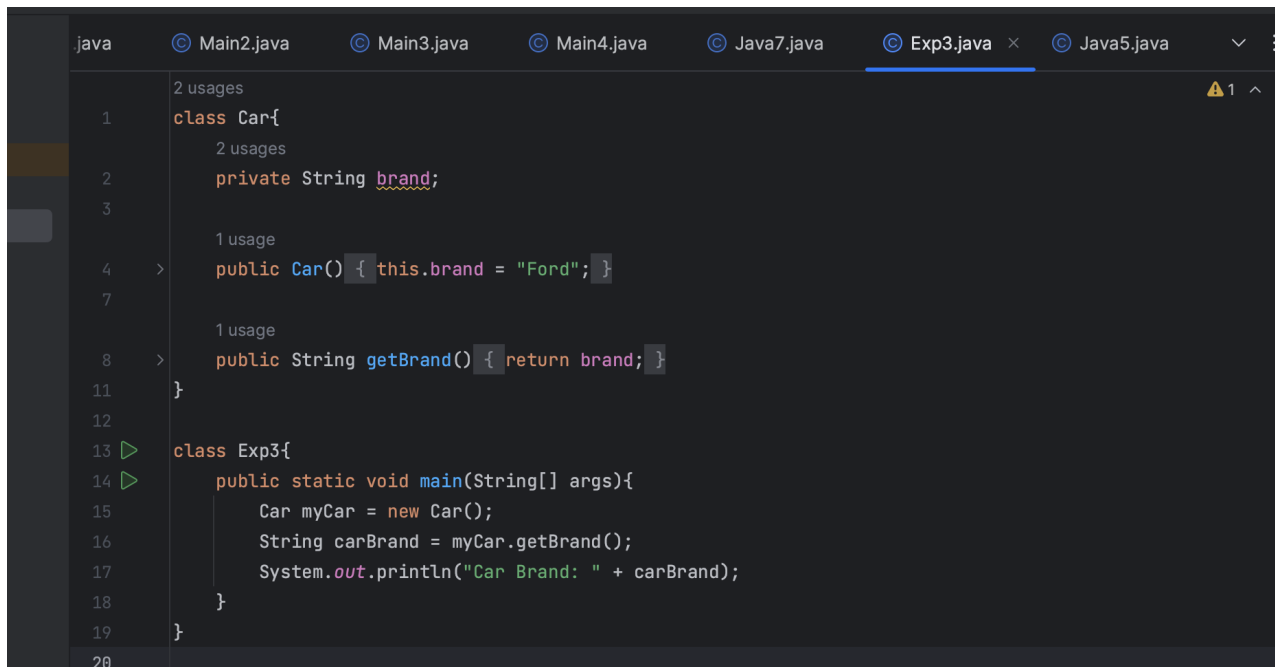
j > src > Java8 > main



```
Run Java8 x
0
6
11
5
6
-1
5
6
30
5
6
Process finished with exit code 0
```

j > src > Java8 > main

Q3)



```

1  class Car{
2      private String brand;
3
4      public Car() { this.brand = "Ford"; }
5
6
7
8      public String getBrand() { return brand; }
9
10 }
11
12
13 class Exp3{
14     public static void main(String[] args){
15         Car myCar = new Car();
16         String carBrand = myCar.getBrand();
17         System.out.println("Car Brand: " + carBrand);
18     }
19 }
20

```

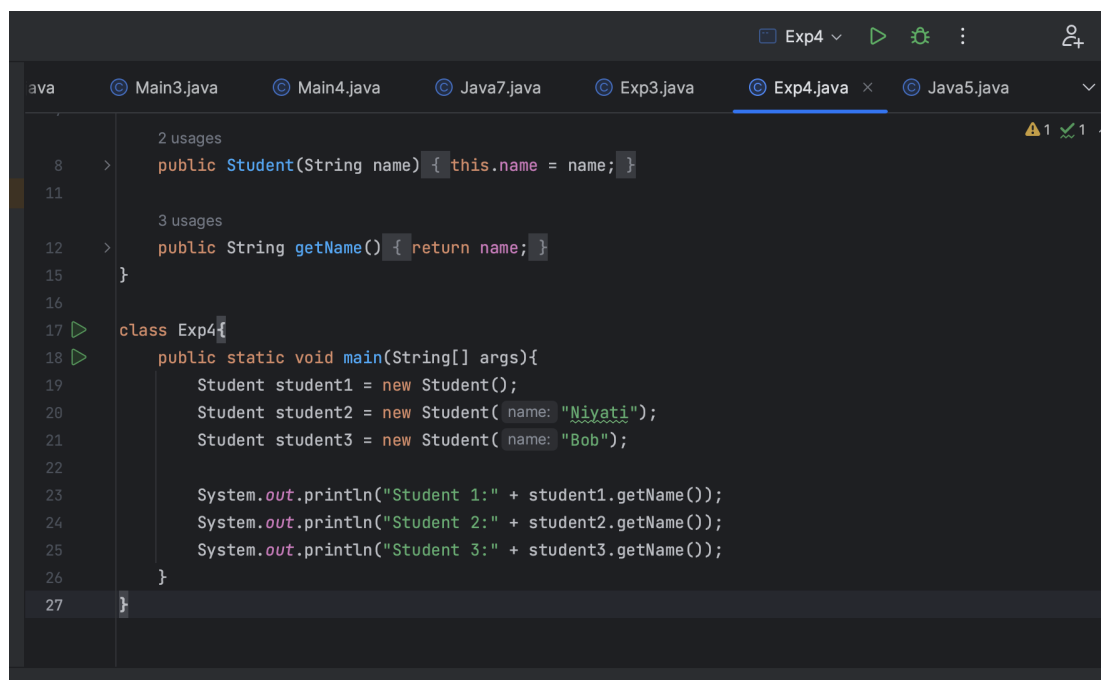


```

/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:/Applications/IntelliJ IDEA.
Car Brand: Ford
Process finished with exit code 0

```

Q4)



```

8  public Student(String name) { this.name = name; }
9
10
11
12 public String getName() { return name; }
13
14 }
15
16
17 class Exp4{
18     public static void main(String[] args){
19         Student student1 = new Student();
20         Student student2 = new Student( name: "Niyati");
21         Student student3 = new Student( name: "Bob");
22
23         System.out.println("Student 1:" + student1.getName());
24         System.out.println("Student 2:" + student2.getName());
25         System.out.println("Student 3:" + student3.getName());
26     }
27 }

```

```
Run Exp4 x
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:/Applications/I
Student 1:unknown
Student 2:Niyati
Student 3:Bob
Process finished with exit code 0
j > src > Exp4.java
```

Q5)

```
Exp5
Main3.java Main4.java Java7.java Exp3.java Exp4.java Exp5.java x
class Complex {
    private double realPart;
    private double imaginaryPart;

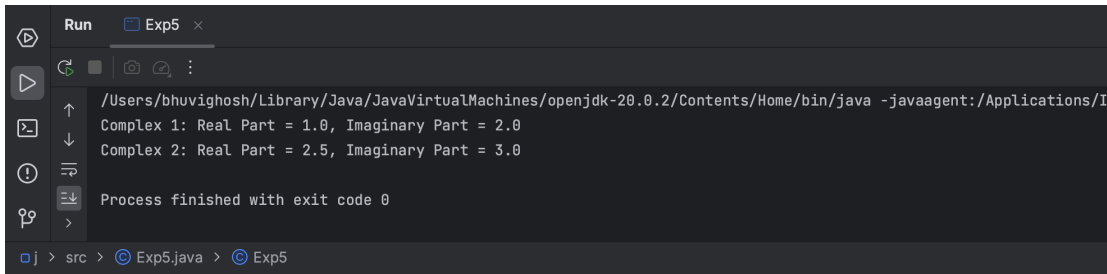
    public Complex(){
        this.realPart = 0.0;
        this.imaginaryPart = 0.0;
    }

    public Complex(double realPart, double imaginaryPart){
        this.realPart = realPart;
        this.imaginaryPart = imaginaryPart;
    }

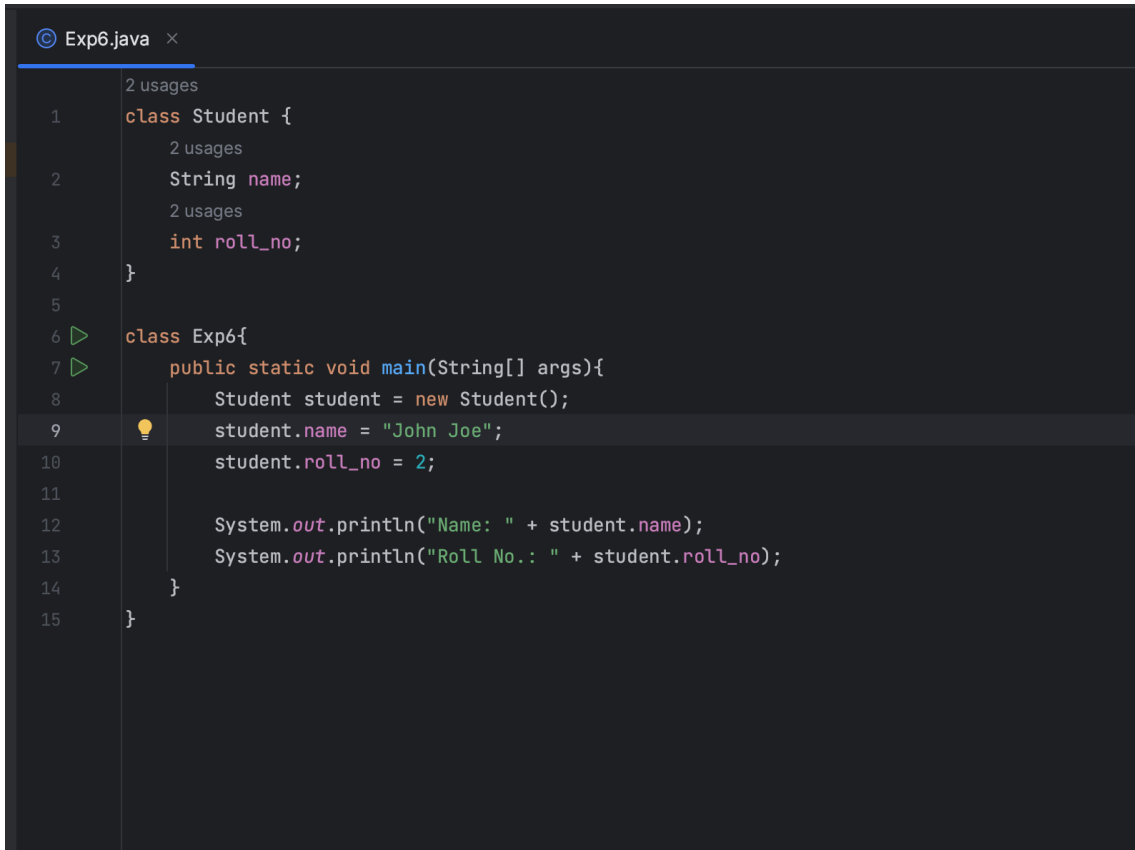
    public void setRealPart(double realPart) { this.realPart = realPart; }
```

```
15 > public void setRealPart(double realPart) { this.realPart = realPart; }
18
19 > public void setImaginaryPart(double imaginaryPart) { this.imaginaryPart = imaginaryPart; }
22
23 > public double getRealPart() { return realPart; }
26
27 > public double getImaginaryPart() { return imaginaryPart; }
30
31
33 > class Exp5{
34 >     public static void main(String[] args){
35 >         Complex complex1 = new Complex();
36 >         Complex complex2 = new Complex( realPart: 2.5, imaginaryPart: 3.0);
37 >
38 >         complex1.setRealPart(1.0);
39 >         complex1.setImaginaryPart(2.0);
40 >
41 >         System.out.println("Complex 1: Real Part = " + complex1.getRealPart() + ", Imaginary Part = " + c
42 >         System.out.println("Complex 2: Real Part = " + complex2.getRealPart() + ", Imaginary Part = " + c
43 >     }
```

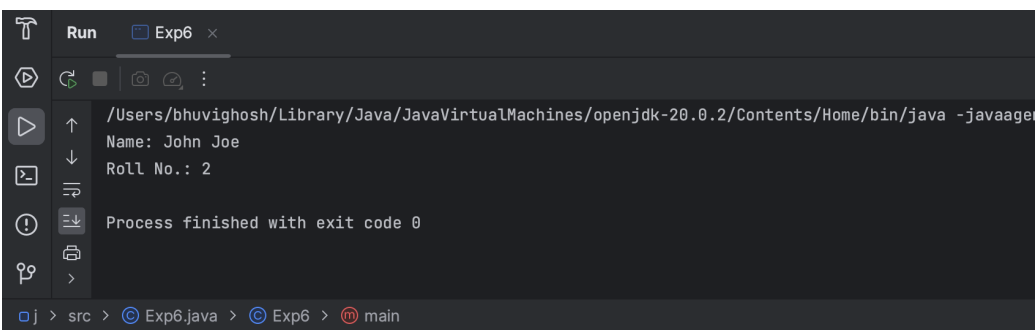
Q6)



```
Run Exp5 x
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaagent:/Applications/I
Complex 1: Real Part = 1.0, Imaginary Part = 2.0
Complex 2: Real Part = 2.5, Imaginary Part = 3.0
Process finished with exit code 0
src > Exp5.java > Exp5
```



```
Exp6.java x
2 usages
1 class Student {
2     2 usages
3     String name;
4     2 usages
5     int roll_no;
6 }
7
8 class Exp6{
9     public static void main(String[] args){
10         Student student = new Student();
11         student.name = "John Joe";
12         student.roll_no = 2;
13
14         System.out.println("Name: " + student.name);
15         System.out.println("Roll No.: " + student.roll_no);
16     }
17 }
```



```
Run Exp6 x
/Users/bhuvighosh/Library/Java/JavaVirtualMachines/openjdk-20.0.2/Contents/Home/bin/java -javaage
Name: John Joe
Roll No.: 2
Process finished with exit code 0
src > Exp6.java > Exp6 > main
```