Object Oriented Programming with Java (Subject Code: BCS-403)

Unit 1

Lecture 4

Lecture 4

- Encapsulation
- Defining Classes in Java
- Control Flow

Encapsulation

- **Encapsulation** is defined as the wrapping up of data under a single unit.
- It is the mechanism that binds together code and the data it manipulates.
- It is a protective shield that prevents the data from being accessed by the code outside this shield.
- Technically in encapsulation, the variables or data of a class is hidden from any other class and can be accessed only through any member function of its own class in which it is declared.

- Encapsulation can be achieved by Declaring all the variables in the class as private and writing public methods in the class to set and get the values of variables.
- It is more defined with the setter and getter method.

Advantages of Encapsulation

- **Data Hiding:** it is a way of restricting the access of our data members by hiding the implementation details. Encapsulation also provides a way for data hiding.
- **Reusability:** Encapsulation also improves the re-usability and is easy to change with new requirements.
- **Testing code is easy:** Encapsulated code is easy to test for unit testing.
- **Freedom to programmer in implementing the details of the system.**

Disadvantages of Encapsulation in Java

- Can lead to increased complexity, especially if not used properly.
- Can make it more difficult to understand how the system works.
- May limit the flexibility of the implementation.

Data Encapsulation in Java

```
class Area {
    int length;
    int breadth;
     // constructor to initialize values
    Area(int length, int breadth)
    {
        this.length = length;
        this.breadth = breadth;
    }
     // method to calculate area
    public void getArea()
    {
        int area = length * breadth;
        System.out.println("Area: " + area);
    }
}
```

```
class Main {
    public static void main(String[] args)
    {
        Area rectangle = new Area(2, 16);
        rectangle.getArea();
    }
}
```

Class Definition in Java

- In object-oriented programming, a class is a basic building block.
- It can be defined as template that describes the data and behavior associated with the class instantiation.
- Instantiating is a class is to create an object (variable) of that class that can be used to access the member variables and methods of the class.
- A class can also be called a logical template to create the objects that share common properties and methods.

- Java provides a reserved keyword class to define a class.
- The keyword must be followed by the class name.
- Inside the class, we declare methods and variables.

class declaration includes the following in the order as it appears:

- Modifiers: A class can be public or has default access.
- class keyword: The class keyword is used to create a class.
- Class name: The name must begin with an initial letter (capitalized by convention).
- Superclass (if any): The name of the class's parent (superclass), if any, preceded by the keyword extends. A class can only extend (subclass) one parent.
- Interfaces (if any): A comma-separated list of interfaces implemented by the class, if any, preceded by the keyword implements. A class can implement more than one interface.
- Body: The class body surrounded by braces, { }.

```
// class definition
public class Calculate {
   // instance variables
   int a;
   int b;
   // constructor to instantiate
   Calculate (int x, int y) {
      this.a = x;
      this.b = y;
   }
    // method to add numbers
   public int add () {
      int res = a + b;
      return res;
   }
}
```

```
class MyMain
{
public static void main(String[] args)
{
    // creating object of Class
    Calculate c1 = new Calculate(45, 4);
    // calling the methods of Calculate class
    System.out.println("Addition is :" + c1.add());
   }
}
```

Control Flow in Java

Java compiler executes the code from top to bottom. The statements in the code are executed according to the order in which they appear.

However, Java provides statements that can be used to control the flow of Java code.

Such statements are called control flow statements. It is one of the fundamental features of Java, which provides a smooth flow of program.

Java provides three types of control flow statements.

1. Decision Making statements
    1. if statements
    2. switch statement
2. Loop statements
    1. do while loop
    2. while loop
    3. for loop
    4. for-each loop
3. Jump statements
    1. break statement
    2. continue statement

Java for loop

```
for(initialization, condition, increment/decrement) {
//block of statements
}
int sum = 0;
for(int j = 1; j<=10; j++)
{
sum = sum + j;
}
System.out.println("The sum of first 10 natural numbers is " +
sum);
```

Java while loop

```
while(condition){
//looping statements
}
```

```
while(i<=10)
{
System.out.println(i);
i = i + 2;
}
```

Java for-each loop

Java provides an enhanced for loop to traverse the data structures like array or collection. In the for-each loop, we don't need to update the loop variable.

The syntax to use the for-each loop in java is given below.

```
for(data_type var : array_name/collection_name){
//statements
}
```

```
public class Calculation {
public static void main(String[] args)
{
String[] names =
{"Java","C","C++","Python","JavaScript"};
System.out.println("Printing the content of the array
names:\n");
for(String name:names) {
System.out.println(name);
} }
}
```