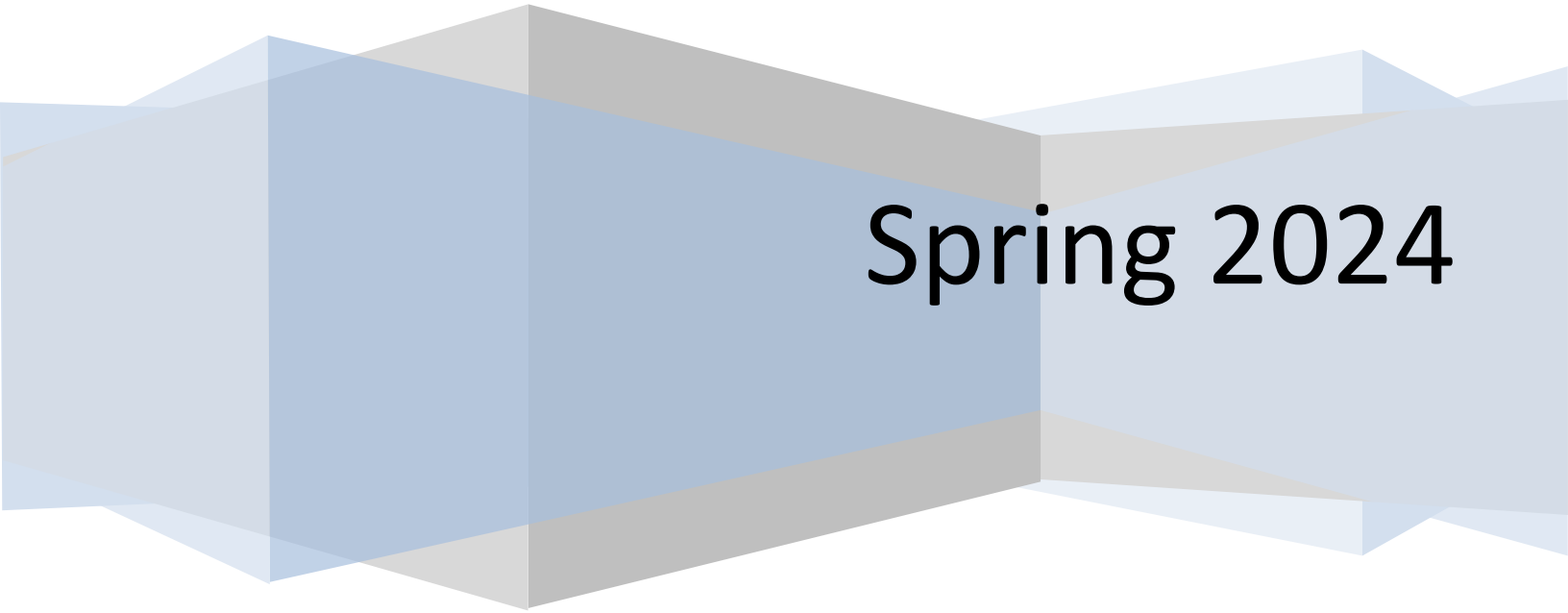
Northern University, Nowshera

Classes and Objects

Week # 05 - Lecture 09 - 10



Spring 2024

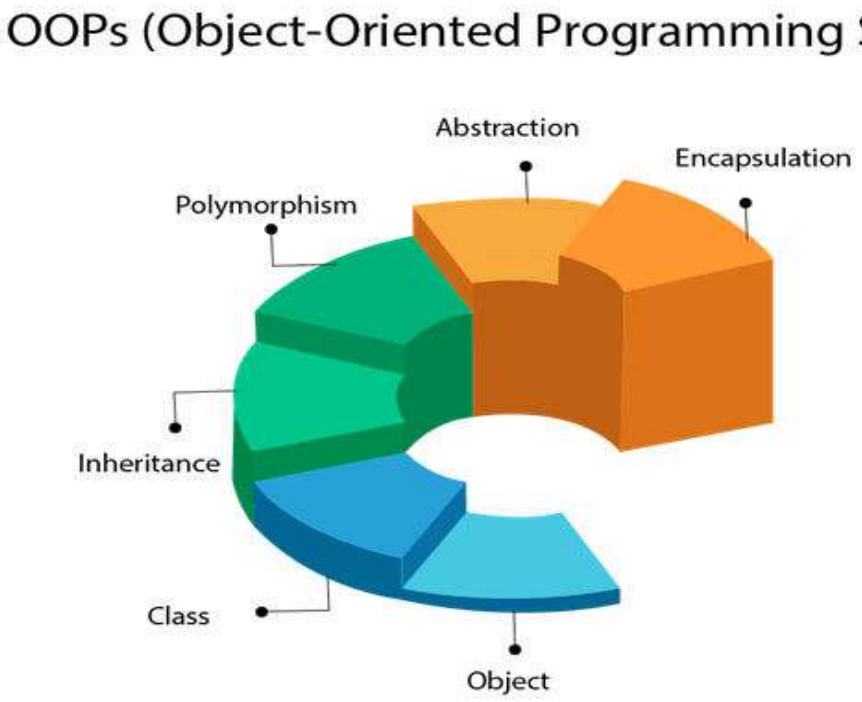
Learning Objectives:

1. **Objects Oriented Programming**
2. **Core Concepts of Object Oriented Programming**
3. **Class and Object**
4. **Access Specifiers**

1. Object Oriented Programming (OOPS)?

Object Oriented Programming is a programming concept that works on the principle that objects are the most important part of your program. It allows users create the objects that they want and then create methods to handle those objects. Manipulating these objects to get results is the goal of Object Oriented Programming.

Object Oriented Programming popularly known as OOP, is used in a modern programming language like Java.



2. Core concepts of OOP:

1. **Class**

   The class is a group of similar entities. It is only a logical component and not the physical entity. For example, if you had a class called “Expensive Cars” it could have objects like Mercedes, BMW, Toyota, etc. Its properties (data) can be price or speed of these cars. While the methods may be performed with these cars are driving, reverse, braking etc.

2. **Object**

   An object can be defined as an instance of a class, and there can be multiple instances of a class in a program. An Object contains both the data and the function, which operates on the data. For example - chair, bike, marker, pen, table, car, etc.

3. **Inheritance**

   Inheritance is an OOPS concept in which one object acquires the properties and behaviors of the parent object. It's creating a parent-child relationship between two classes. It offers robust and natural mechanism for organizing and structure of any software.

4. **Polymorphism**

   Polymorphism refers to the ability of a variable, object or function to take on multiple forms. For example, in English, the verb *run* has a different meaning if you use it with *a laptop*, *a foot race*, and *business*. Here, we understand the meaning of *run* based on the other words used along with it. The same also applied to Polymorphism.

5. **Abstraction**

   An abstraction is an act of representing essential features without including background details. It is a technique of creating a new data type that is suited for a specific application. For example, while driving a car, you do not have to be concerned with its internal working. Here you just need to concern about parts like steering wheel, Gears, accelerator, etc.

6. **Encapsulation**

   Encapsulation is an OOP technique of wrapping the data and code. In this OOPS concept, the variables of a class are always hidden from other classes. It can only be accessed using the methods of their current class. For example - in school, a student cannot exist without a class.

7. **Association**

   Association is a relationship between two objects. It defines the diversity between objects. In this OOP concept, all objects have their separate lifecycle, and there is no owner. For example, many students can associate with one teacher while one student can also associate with multiple teachers.

8. **Aggregation**

   In this technique, all objects have their separate lifecycle. However, there is ownership such that child object can't belong to another parent object. For example consider class/objects department and teacher. Here, a single teacher can't belong to multiple departments, but even if we delete the department, the teacher object will never be destroyed.

9. **Composition**

   A composition is a specialized form of Aggregation. It is also called "death" relationship. Child objects do not have their lifecycle so when parent object deletes all child object will also delete automatically. For that, let's take an example of House and rooms. Any house can have several rooms. One room can't become part of two different houses. So, if you delete the house room will also be deleted.

3. Advantages of OOPS:

- OOP offers easy to understand and a clear modular structure for programs.
- Objects created for Object-Oriented Programs can be reused in other programs. Thus it saves significant development cost.
- Large programs are difficult to write, but if the development and designing team follow OOPS concept then they can better design with minimum flaws.
- It also enhances program modularity because every object exists independently.

4. Class and Objects

Java is an object-oriented programming (OOP) language. In this lecture, you'll be introduced to OOP and how you can create custom class and objects in your Java program.

Java is an object-oriented programming language. It allows you to divide complex problems into smaller sets by creating objects.

These objects share two characteristics:

1. State
2. Behavior

Let's take few examples:

1. `Lamp` is an object
   - It can be in `on` or `off` state.
   - You can `turn on` and `turn off` lamp (behavior).
2. `Bicycle` is an object
   - It has `current gear`, `two wheels`, `number of gear` etc. states.
   - It has `braking`, `accelerating`, `changing gears` etc. behavior.

4.1 Java Class

Before you create objects in Java, you need to define a class. A class is a blueprint for the object. We can think of 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.

4.1.1 How to define a class in Java?

Here's how a class is defined in Java:

```
class ClassName {
  // variables
  // methods
}
```

Here's an example:

```
class Lamp {

  // instance variable
  private boolean isOn;

  // method
  public void turnOn() {
    isOn = true;
  }

  // method
  public void turnOff() {
    isOn = false;
  }
}
```

Here, we defined a class named `Lamp`. The class has one instance variable (variable defined inside class) `isOn` and two methods `turnOn()` and `turnOff()`. These variables and methods defined within a class are called **members** of the class.

Notice two keywords, `private` and `public` in the above program. These are access modifiers which will be discussed in detail later. For now, just remember:

- The `private` keyword makes instance variables and methods private which can be accessed only from inside the same class.
- The `public` keyword makes instance variables and methods public which can be accessed from outside of the class.

  In the above program, `isOn` variable is private whereas `turnOn()` and `turnOff()` methods are public.

  If you try to access `private` members from outside of the class, compiler throws error.

4.2. Java Objects

When class is defined, only the specification for the object is defined; no memory or storage is allocated.

To access members defined within the class, you need to create objects. Let's create objects of `Lamp` class.

```
1.  class Lamp {
2.   boolean isOn;
3.
4.   void turnOn() {
5.    isOn = true;
6.   }
7.
8.   void turnOff() {
9.    isOn = false;
10.  }
11. }
```

```
12.
13. class Main {
14. public static void main(String[] args) {
15.   Lamp l1 = new Lamp(); // create l1 object of Lamp class
16.   Lamp l2 = new Lamp(); // create l2 object of Lamp class
17.  }
18. }
```

This program creates two objects `l1` and `l2` of class `Lamp`.

---

5. How to access members?

You can access members (call methods and access instance variables) by using `.` operator.

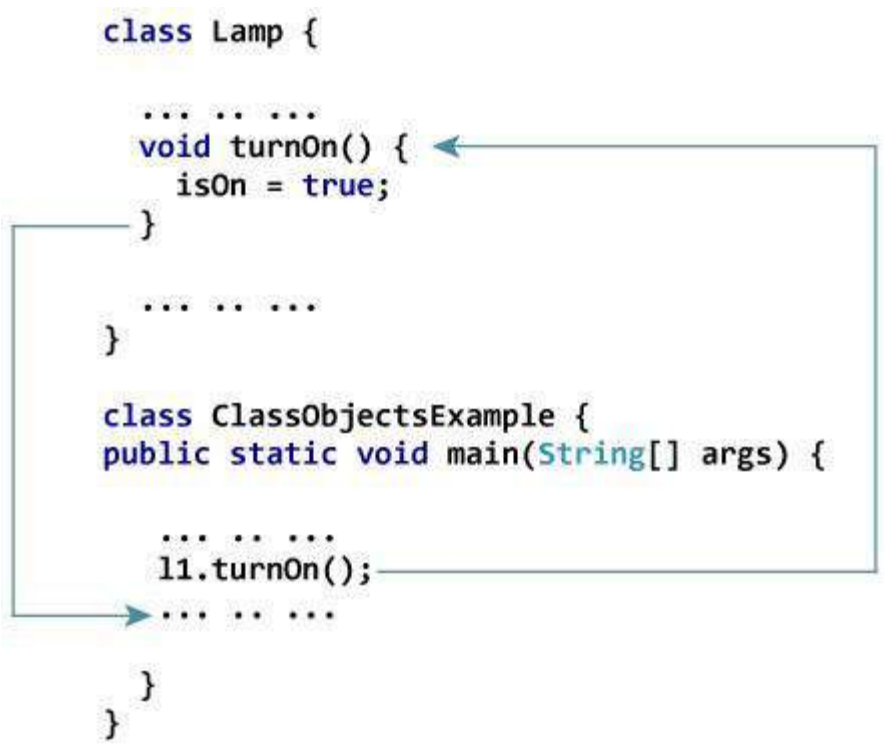
For example,

```
l1.turnOn();
```

This statement calls `turnOn()` method inside `Lamp` class for `l1` object.

We have mentioned word **method** quite a few times. You will learn about *Java methods* in detail in the next chapter. Here's what you need to know for now:

When you call the method using the above statement, all statements within the body of `turnOn()` method are executed. Then, the control of program jumps back to the statement following `l1.turnOn();`



Similarly, the instance variable can be accessed as:

```
l2.isOn = false;
```

It is important to note that, the `private` members can be accessed only from inside the class. If the code `l2.isOn = false;` lies within the `main()` method (outside of the `Lamp` class), compiler will show error.

Example: Java Class and Objects

```
1. class Lamp {
2.  boolean isOn;
3.
4.  void turnOn() {
```

```
    isOn = true;
  }

  void turnOff() {
    isOn = false;
  }

  void displayLightStatus() {

     System.out.println("Light on? " + isOn);
  }
}


class ClassObjectsExample {
public static void main(String[] args) {

   Lamp l1 = new Lamp(), l2 = new Lamp();

   l1.turnOn();
   l2.turnOff();

   l1.displayLightStatus();
   l2.displayLightStatus();
  }
}
```

When you run the program, the output will be:

```
Light on? true
Light on? false
```

In the above program

- `Lamp` class is created.
- The class has an instance variable `isOn` and three methods `turnOn()`, `turnOff()` and `displayLightStatus()`.
- Two objects `l1` and `l2` of `Lamp` class are created in the `main()` function.
- Here, `turnOn()` method is called using `l1` object: `l1.turnOn();`
- This method sets `isOn` instance variable of `l1` object to `true`.
- And, `turnOff()` method is called using `l2` object: `l2.turnOff();`
- This method sets `isOff` instance variable of `l2` object to `false`.
- Finally, `l1.displayLightStatus();` statement displays Light on? true because `isOn` variable holds `true` for `l1` object.
- And, `l2.displayLightStatus();` statement displays Light on? false because `isOn` variable holds `false` for `l2` object
- Note, variables defined within a class are called **instance variable** for a reason.
- When an object is initialized, it's called an instance. Each instance contains its own copy of these variables. For example, `isOn` variable for objects `l1` and `l2` are different.

Example #1: Write a class named "**Student**" with attributes name registration number, and marks and following functions:

3. setStudent() : to get values from user
4. getStudent() : print attributes – name, Registration number and marks

Also write main function to call these functions.

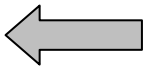
Solution:

```
import java.util.Scanner;
public class Student {
    private String name;
    private String regNo;
    private float marks;

    public void setStudent() {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter name: ");
        name=s.nextLine();
        System.out.println("Enter Registration No: ");
        regNo=s.nextLine();
        System.out.println("Enter marks: ");
        marks=s.nextFloat();
    }

    public void getStudent() {
        System.out.println(name+"\t\t"+regNo+"\t\t"+marks);
    }
}
public class Main {
    public static void main(String[] rgs)
    {
        Student s1=new Student();
        System.out.println("Enter Details of Student #1");
        s1.setStudent();

        Student s2=new Student();
        System.out.println("Enter Details of Student #2");
        s2.setStudent();

        Student s3=new Student();
        System.out.println("Enter Details of Student #3");
        s3.setStudent();

        System.out.println("NAME\tREG#\tMARKS\n_____________________________________");
        s1.getStudent();
        s2.getStudent();
        s3.getStudent();
    }
}
```

Example #2: Accessing member function outside the class:

In this example we will discuss how to access private data members outside the class? Let's suppose we want to print details of a student with maximum marks. Here we need marks property in main program, which is private and cannot be accessed with dot operator. So we need a member function that will return private value to calling object. See the following example code:

```
import java.util.Scanner;
public class Student {
    private String name;
    private String regNo;
    private float marks;

    public void setStudent() {
        Scanner s=new Scanner(System.in);
        System.out.println("Enter name: ");
        name=s.nextLine();
        System.out.println("Enter Registration No: ");
        regNo=s.nextLine();
        System.out.println("Enter marks: ");
        marks=s.nextFloat();
    }

    public void getStudent() {
        System.out.println(name+"\t\t"+regNo+"\t\t"+marks);
    }
   public float retMarks()
   {
       return marks;
   }

}
public class Main {
    public static void main(String[] rgs)
    {
        Student s1=new Student();
        System.out.println("Enter Details of Student #1");
        s1.setStudent();

        Student s2=new Student();
        System.out.println("Enter Details of Student #2");
        s2.setStudent();

        Student s3=new Student();
        System.out.println("Enter Details of Student #3");
        s3.setStudent();

        System.out.println("NAME\tREG#\tMARKS\n_____________________________________");
        s1.getStudent();
        s2.getStudent();
        s3.getStudent();
        System.out.println("TOPER IS: \n ");
        if(s1.retMarks()>s2.retMarks() && s1.retMarks()> s3.retMarks())
```



Function to return private values

```
                s1.getStudent();
            else
                if (s2.retMarks()>s1.retMarks()&& s2.retMarks()>s3.retMarks())
                    s2.getStudent();
                else s3.getStudent();

        }
    }
```

In the above program

- Student class is created.
- In main three objects/ instances of student class has been created by using new operator
- setStudent() method is called for each instance variable that will allow the user to input name, registration and marks of student
- We have created retMarks() method to get private property, As we cannot access private data of instance variables in main() or outside the class
- retMarks() is called in main to get and compare marks of three instance variables.

Assignment # 2

Q#1: Update student class (see example #2) by adding following data members and methods

- RollNo (int)
- Name (String)
- Subject name (String)
- Grade (char)
- obtained marks (float)
- input ( ): will input Rollno, Name, Subject Name, Obtained marks.
- Calculate Grade ( ): Will calculate grade of student based on credit hours, total marks and obtained marks. Here this is assumed that grading criteria is known for you as followed in NU
- printStudents (char grade) : Will print all those students whose grade matches the given grade as argument

The Program should print the following information:

1) Create at least 5 objects in main and get their values by calling required functions.
2) calculate their grades based on their input values
3) Display the information of student with highest marks
4) Display all those students whose grade is A
5) Print failure students (below d)

Q#2: Write a program to maintain the travelling of different travelers. The traveler class contains kilometers and hours travelled as data members and required function to get input and display travelling information.

You are required to get travelling information of at least 5 travelers and display them in tabular view.

Also display information of traveler who travelled maximum distance in minimum time.

Q#3: You have a class "Distance" with feet and inches as data members and member functions as follows.

1. getdistance() to get distance values
2. showdistance() to print feet and inches

In main function you have to keep input from user and saved them in five variables/objects of type Distance. After completion of input, your program should display following information.

- Shortest Distance
- Longest Distance
- TotalDistance