

Introduction To OOP

Session 6



NIIT
Institute

Objectives

- Creating Classes and Objects
 - State the the syntax for declaring classes
 - Describe constructor and states its syntax
 - State the syntax of creating objects
- Instance Variables
 - State the purpose of instance variables
 - State the syntax of declaring instance variables
- Methods
 - Explain the purpose of methods and its naming conventions.
 - Explain instance methods
 - Describe variable-argument method



NIIT
Institute

- **Object-oriented programming (OOP)**

- Is a [programming paradigm](#) based on the concept of "[objects](#)", which may contain [data](#), in the form of [fields](#), often known as *attributes*; and code, in the form of procedures, often known as [methods](#).
- OOP allows decomposition of a problem into a number of entities called objects and then builds data and functions around these objects.
 - The software is divided into a number of small units called objects. The data and functions are built around these objects.
 - The data of the objects can be accessed only by the functions associated with that object.
 - The functions of one object can access the functions of another object.



Concept an Object

- An object is the software representation of a real-world entity.
- Example of real-world entities exist around everyone, such as a car, bike, flower or person.
- Every entity has some characteristics and is capable of performing certain actions
 - Characteristics are attributes or features describing the entity
 - Actions are activities or operations involving the object



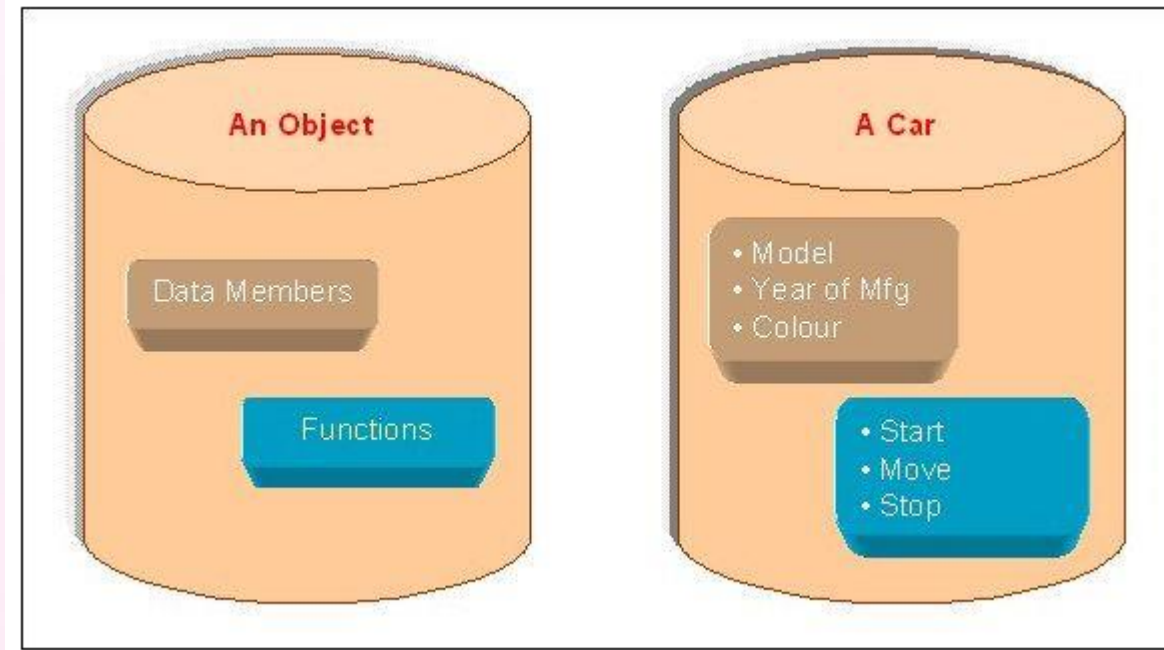
Software Object

- The concept objects in the real-world can be extended to the programming world where software “objects” can be defined.
- Like its real-world counterpart, a software object has state and behavior.
 - The “state” of the software object refers to its characteristics or attributes
 - The “behavior” of the software object comprises its actions.
- The software object stores its state in filed(also called variables in programming languages) and exposes its behavior through method (called functions in programming languages)



Defining a Class

- A class defines an entity in terms of common characteristics and actions
- A class is a template or blueprint which defines the state and behavior for all objects belonging to that class.
- In an object-oriented language, a class comprises fields and methods, collectively called as members, which depict the state and behavior respectively



JMULBE
Inure

Defining class

Defining a Class

- A class defines:
 - The characteristics and behavior of an object.
 - The member variables and methods of objects that share common characteristics.
- The following code snippet shows how to declare a class:

```
class <ClassName>
{
//Declaration of member variables
//Declaration of methods
}
```



JMULBE
Inure

Class

- A class defines a new data type.
- In OOP languages it is mandatory to create a class for representing data.
- A class is a blueprint of an object that contains variables for storing data and functions to perform operations on the data.
- A class will not occupy any memory space and hence it is only a logical representation of data.
- Every time an instance of a class is created, an object is created.
- The object contains its own copy of each instance variable defined by the class.
- A dot operator (.) is used to access these variables.
- The dot operator links the name of the object with the name of an instance variable.

Declaring Classes

- Syntax

```
class class_name
```

```
{
```

```
    //instance variables
```

```
    //instance methods
```

```
}
```

- ❖ Class name should be a noun.
- ❖ Class name can be mixed case, with the first letter of each internal word capitalized
- ❖ Class name should be simple, descriptive and meaningful.
- ❖ Class names cannot Java keyword
- ❖ Class name can begin with a dollar(\$) symbol or an underscore(_) character

Declaring Classes

- Example

```
class Employee
{
    private int id;
    private String name;
    int getId()      {
        return id;
    }
    String getName() {
        return name;
    }
    void setId(int _id) {
        id = _id;
    }
    void setName(String _name) {
        name = _name;
    }
}
```

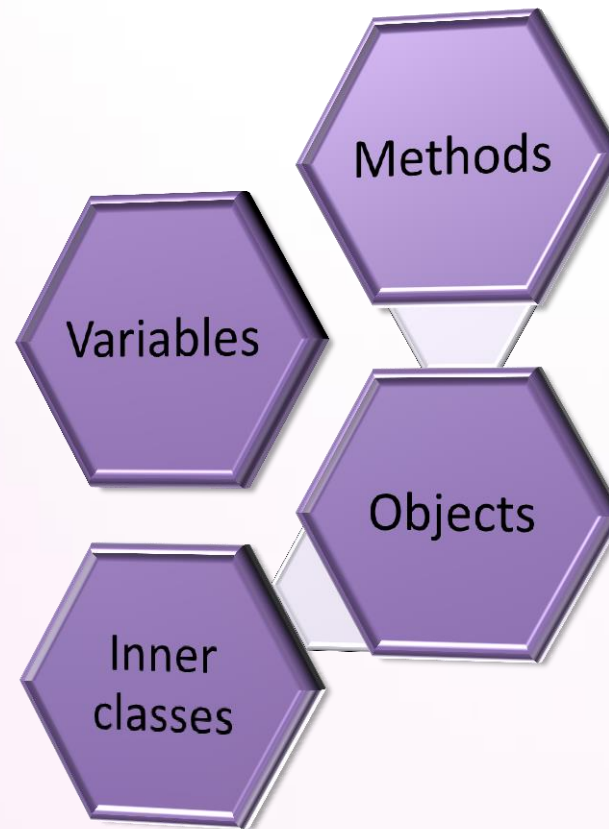


JMULBE
Inure

Class Members

Identifying Class Members

- In Java, a class can contain the following members:



Constructor

- Constructors are special methods in a class.
- Constructors are used to initialize member variables of the class.
- It has the same name as the class name and does not have a return type.
- The constructor is invoked automatically when an object of class is created.
- Two types of constructors:
 - Parameterized constructors
 - Default constructors (Implicit constructors)



Constructor

- Example: declaring constructor for Book class

```
class Book
{
    String title;
    int price;

    Book ()
    {
        this.title = "";
        this.price = 0;
    }

    Book(String _title, int _price)
    {
        this.title = _title;
        this.price = _price;
    }
}
```



NIIT
JALGAON

Creating Objects

- An object is created using **new** keyword
- On encountering the new keyword, the JVM allocates memory for the object, invokes the constructor of the class, and return a reference of that allocated memory. The reference is assigned to the object.
- Syntax

<class_name> <object_name> = new <constructor_name()>;

An object can be declared without using the new operator:

<class_name> <object_name>;

- Example

```
Point p = new Point();  
Point p1= new Point(100,200);  
Point p2;
```


Concept of Instance Variables

- Instance variables are used to store information about an entity.
- Each object of class will contains a copy of instance variables.

```
class Car
{
    float price;
    String color;
    . . .
}
```

Concept of Instance Variables


- Declare instance variables

[access_modifier] data_type instanceVariableName

- Access instance variables

objectName.instanceVariableName

- Example



```
Car objCar = new Car();  
objCar.price=500000;
```

Concept of Methods

- A method is defined as the actual implementation of an operation on an object.
- Following conventions have to be followed while naming a method.
 - Cannot be Java keyword
 - Cannot contain spaces
 - Cannot begin with a digit
 - Can begin with a letter, underscore or a “\$” symbol
 - Should be a verb in lowercase
 - Should be descriptive and meaningful
 - Should be a multi-word name that begins with a verb in lowercase, followed by adjectives, nouns,...



NIIT
Application

Declaring Instance Method

- Syntax

```
access_specifier modifier datatype method_name ([parameter_list])  
{  
    //body of the method  
}
```

- Example

```
//Class Declaration  
class Student  
{  
    int age;                                //Instance variable  
    //instance method  
    int getAge() {  
        return age;                        //Accessing Instance variable  
    }  
    void setAge(int i) {  
        age=i;                             //Accessing Instance variable  
    }  
}
```

Declaring Instance Method

- Methods that have a return type other than `void`, return a value to the calling routine using the `return` statement.
- Many methods need parameters.
- Parameters allow a method to be generalized.



NIIT
Institute
of Information
Technology

Declaring Instance Method

- Passing Arguments by Value
 - When the value from the calling method is passed as an argument to the called method, any changes made to that passed value in the called method will not modify the value in the calling method.
 - Variables of primitive data types, such as `int` and `float` are passed by value
- Passing Arguments by Reference
 - Call by reference enables the called method to change the value of the parameters passed to it from the calling method
 - When the method returns, the passed-in reference still references the same object as before.
 - Values of the instance variables can be changed in the method

Variable Argument Methods

```
//Class Declaration
• class Welcome
{
    public static void showArray(int...num)      {
        for (int I : num)
            System.out.println("Value is "+ i + ". ");
        }
    public static void main(String[] args){
        showArray(201, 301);
    }
}
```

OUTPUT:

Value is: 201

Value is: 301

Demo

- Teacher demo process defining class and member class to student.



Summary

- Creating Classes and Objects
- Instance Variables
- Methods

