iGenuine

Java Programming

Lesson 6 - Classes and Objects

WHAT WILL YOU LEARN IN THIS SESSION ?

- Define what a class is and how it relates to real-world objects.
- Explain the concept of an object and its role in Java programming.
- Create a basic Java class with attributes and methods.
- Use access modifiers appropriately to control the visibility and accessibility of class members
- Writing Java programs that include examples of instance variables, class variables, final variables, and local variables.

iGenuine

01

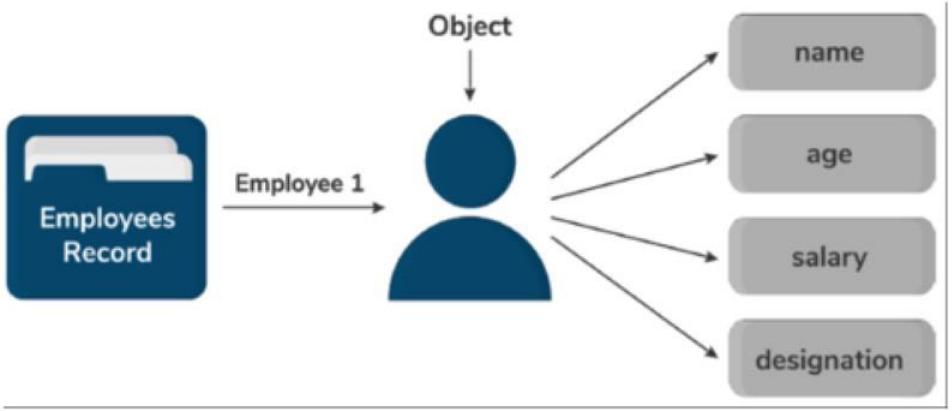
Introduction to Classes and Objects

iGenuine

Introduction

We use programming to solve real-world problems, and it won't make much sense if one can't model real-world scenarios using programming languages. This is where object-oriented programming comes into play.

Object-oriented programming ,also called OOP, is a programming model that is dependent on the **concept of Objects and classes.**



Java objects

- Java is an object-oriented programming language.
- Everything in Java is associated with classes and objects.
- An object is any entity that has a state and behavior.
- For example: in real life, a car is an object.
- The car has attributes, such as weight and color, and methods, such as drive and brake.

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.

iGenuine

Java Class Syntax

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

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

iGenuine

Java Class

A class in Java can contain:

- Fields
- Methods
- Constructors
- Blocks
- Nested class and interface

Java Class

- For bicycle object, we can create the class as:

```
class Bicycle {

  // state or field
    int gear = 5;

  // behavior or method
  public void braking() {
    System.out.println("Working of Braking");
  }
}
```

Java Objects

- An object is called an instance of a class.
- For example, suppose Bicycle is a class then MountainBicycle, SportsBicycle, TouringBicycle, etc can be considered as objects of the class.

iGenuine

Creating an Object in Java

- Java Object Syntax: `className variable_name = new className();`

// for Bicycle class

- `Bicycle sportsBicycle = new Bicycle();`

- `Bicycle touringBicycle = new Bicycle();`

iGenuine

Creating an Object in Java

- className is the name of class that can be anything like: Bicycle that we declared in the above example.

- variable_name is name of reference variable that is used to hold the reference of created object.

- The new is a keyword which is used to allocate memory for the object.

Example: Creating a Class and its object

```
public class Student{
        String name;
        int rollno;
        int age;

       void info(){
        System.out.println("Name: "+name);
        System.out.println("Roll Number: "+rollno);
        System.out.println("Age: "+age);
       }
```

Example: Creating a Class and its object

```
public static void main(String[] args) {
        Student student = new Student();
        // Accessing and property value
        student.name = "Ramesh";
        student.rollno = 253;
        student.age = 25;
        // Calling method
        student.info();
    }
}
```

```
    // Calling method
    student.info();
}
}
```

Output:

Name: Ramesh

Roll Number: 253

Age: 25

02

Three ways to initialize object

iGenuine

There are 3 ways to initialize object in Java.

- By reference variable
- By method
- By constructor

1) Object and Class Example: Initialization through reference

iGenuine

- Initializing an object means storing data into the object.

```
class Student{
 int id;
 String name;
}

class TestStudent2{

 public static void main(String args[]){
  Student s1=new Student();
  s1.id=101;
  s1.name="Sonoo";
  System.out.println(s1.id+" "+s1.name);
 }
}
```

We can also create multiple objects and store information in it through reference variable.

iGenuine

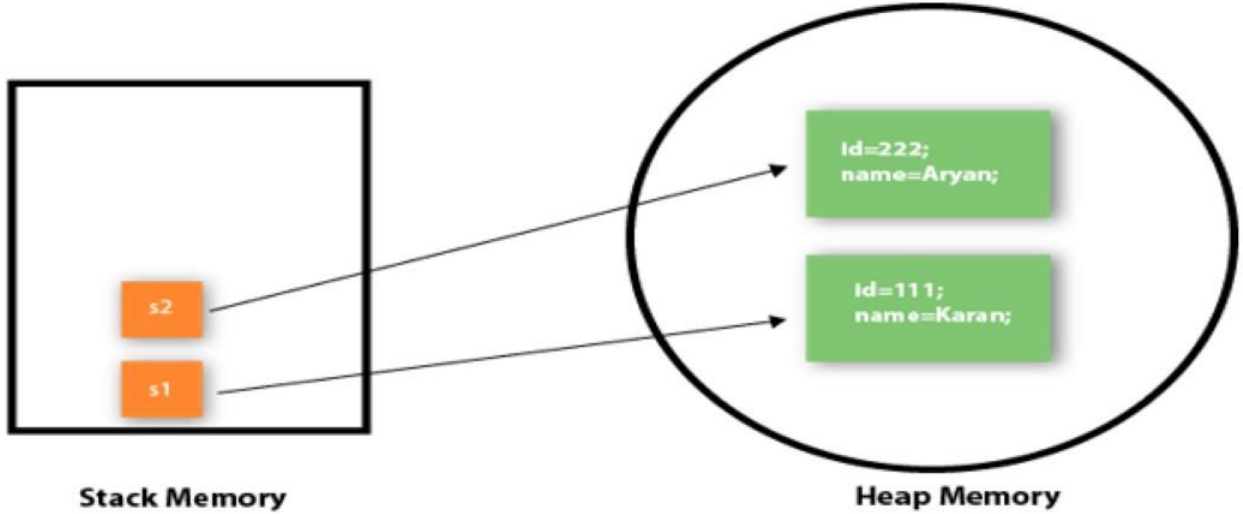
```
class Student{
 int id;
 String name;
}
class TestStudent3{
 public static void main(String args[]){
  //Creating objects
  Student s1=new Student();
  Student s2=new Student();
  //Initializing objects
  s1.id=101;
  s1.name="Sonoo";
  s2.id=102;
  s2.name="Amit";
  System.out.println(s1.id+" "+s1.name);
  System.out.println(s2.id+" "+s2.name);
 }
}
```

2) Object and Class Example: Initialization through method

- In this example, we are creating the two objects of Student class and initializing the value to these objects by invoking the insertRecord method.
- Here, we are displaying the state (data) of the objects by invoking the displayInformation() method.

```
class Student{
 int rollno;
 String name;
 void insertRecord(int r, String n){
  rollno=r;
  name=n;
 }
 void
displayInformation(){System.out.println(rollno+"
"+name);
}
}
```

```
class TestStudent4{
 public static void main(String args[]){
  Student s1=new Student();
  Student s2=new Student();
  s1.insertRecord(111,"Karan");
  s2.insertRecord(222,"Aryan");
  s1.displayInformation();
  s2.displayInformation();
 }
}
```



- Object gets the memory in heap memory area.
- The reference variable refers to the object allocated in the heap memory area.
- Here, s1 and s2 both are reference variables that refer to the objects allocated in memory.

3) Object and Class Example: Initialization through a constructor

03

Constructors in Java

iGenuine

What is a Constructor?

- A constructor in Java is a **special method** that is called when an object is instantiated.
- Its primary purpose is to **initialize** the newly created object.
- Constructors have the **same name as the class** and do not have a return type.

iGenuine

Types of Constructors

- Default Constructor
- Parameterized Constructor
- Copy Constructor

iGenuine

1. Default Constructor

- A default constructor is a **no-argument constructor** that the Java compiler automatically provides if no other constructors are defined in the class.

- It initializes the object with **default values**.

Example:

```
public static void main(String[] args) {
        Dog dog = new Dog(); // Default constructor is called
        System.out.println(dog.getName());
          System.out.println(dog.getAge());
 }
```

iGenuine

2. Parameterized Constructor

A parameterized constructor allows you to initialize an object with specific values provided as **arguments.**

Example:

```
public Dog(String name, int age) {
        name = name;
        age = age;
    }
public static void main(String[] args) {
          Dog dog = new Dog("Buddy", 3);
             System.out.println(dog.getName());
}
```

iGenuine

3. Copy Constructor

- A copy constructor creates a new object by **copying** the attributes of an existing object.
- Java does **not provide a default copy constructor**, so you have to define it yourself.

Example:

```
public Dog(Dog dog) {
        name = name;
        age = age;
    }
public static void main(String[] args) {
        Dog dog1 = new Dog("Buddy", 3);
        Dog dog2 = new Dog(dog1);
        System.out.println(dog2.getName());
        System.out.println(dog2.getAge());
    }
```

Scenario: Social Media Platform (e.g., Instagram)

iGenuine

Example Classes:

- **User Class:** Represents a user on the platform.
- **Post Class:** Represents a post made by a user.
- **Comment Class:** Represents a comment made on a post.
- **Message Class:** Represents a direct message between users.

iGenuine

1.User Class:

```
public User(String userIdParam, String usernameParam, String emailParam,
String profilePictureUrlParam) {
        userId = userIdParam;
        username = usernameParam;
        email = emailParam;
        profilePictureUrl = profilePictureUrlParam;
    }
```

```
User newUser = new User("U001", "john_doe", "john.doe@example.com", "profilePicUrl");
```

2.Post Class:

```
public Post(String postIdParam, User userParam, String contentParam, String
imageUrlParam, Date timestampParam) {
        postId = postIdParam;
        user = userParam;
        content = contentParam;
        imageUrl = imageUrlParam;
        timestamp = timestampParam;
        comments = new ArrayList<>();
    }
```

```
Post newPost = new Post("P001", newUser, "Had a great day!", "imageUrl",
new Date());
```

3.Comment Class:

```
public Comment(String commentIdParam, User userParam, String textParam,
Date timestampParam) {
        commentId = commentIdParam;
        user = userParam;
        text = textParam;
        timestamp = timestampParam;
    }
```

Comment newComment = new Comment("C001", newUser, "Nice picture!", new Date());

4.Message Class

```
public Message(String messageIdParam, User senderParam, User receiverParam,
String textParam, Date timestampParam) {
        messageId = messageIdParam;
        sender = senderParam;
        receiver = receiverParam;
        text = textParam;
        timestamp = timestampParam;
    }
```

Message newMessage = new Message("M001", newUser, receiverUser, "Hello!", new Date());

Summary

1. In these popular online platforms, constructors play a **vital role** in initializing objects with the necessary data, ensuring that each entity in the system is properly set up.

1. This approach helps maintain a consistent and reliable state across the application's various components, making it easier to manage interactions between users, posts, comments, and messages.

THANK YOU