Class, Object
Method & Constructors

Shristi Technology Labs

Contents

- What is a class?
- Creating Objects
- Adding methods in a class
- Use of constructor
- Adding Parameterized constructor
- Use of this keyword

What is a class?

- Is a building block of oops
- Is a template or blueprint of an object
- Has state and behaviour of an entity

```
class classname{
        //instance variables(state)
        // methods(behaviour)
    }
```

- Each instance (object) of the class has its own copy of instance variables

Example for a class

```
class Employee{
 // String name;
 // int id, salary;
 }
```

- ***name, id and salary*** are called ***instance variables***
- The instance variables take the default value of the data type being used

(ie***) name is null .***

salary and id will be 0

Objects

- Object is an instance of a class
- Has state and behaviour
- State depicts the data(variables)
- Behaviour depicts the operations(methods) on that instance
- To create an object use ***new*** keyword

classname variable-name; // only a reference is created

variable-name = new classname(); // memory gets allocated to the object

eg.

```
Employee emp;
emp  = new Employee();      // memory is allocated to the object
```

Instance Variables

- Instance variables for an object can be initialised using dot operator

eg.

```
Employee emp = new Employee();
emp.name  = “Ram”;
emp.salary = 1000;
```

Creating the second object which takes different values

```
Employee emp1 = new Employee();
emp1.name = “John”;
emp1.salary = 1300;
```

Example

```
class Employee{
String name; int salary;
```

```
public static void main(String args[]){
    Employee e = new Employee();
    e.name="Ram";
    e.salary =1000;
    System.out.print(e.name+" "+e.salary);
    Employee e1 = new Employee();
    e1.name="John";
    e1.salary =2000;
    System.out.print(e1.name+" "+e1.salary);
}
```

Example for two class scenario

```
class Employee{
String name;
 int salary;
}
```

```
class  Test{
public static void main(String args[]){
    Employee e = new Employee();
    e.name="Ram";
    e.salary =1000;
    System.out.print(e.name+" "+e.salary);
    Employee e1 = new Employee();
    e1.name="John";
    e1.salary =2000;
    System.out.print(e1.name+" "+e1.salary);
 }
```

Methods

- **Are operations done on that object**
- **Can be with or without return type and with/without parameter**

access-specifier return-type method-name(parameter-list)

eg

```
 void getDetails(){
// set of stmts
}
String greet(String message){
    return  “Welcome “ +message;
    }
```

Example for adding methods

```
class Employee{
   String name;
    int salary;
 void getDetails(){
System.out.print("name "+name);
System.out.print("salary"+salary);
}
String greet(String msg){
 return "welcome"+msg;
 }

     }
```

```
class Test{
public static void main(String args[]){
     Employee e = new Employee();
     e.name="Ram";
     e.salary =1000;
     e.getDetails();
     String v = e.greet("Java");
     System.out.print(v);
     Employee e1 = new Employee();
     e1.name="John";
     e1.salary =2000;
     e1.getDetails();
      System.out.print(e1.greet("back"));
     }
```

Constructors

- Are used to initialize instance variables
- Must have the same name as that of the class name
- A constructor does not have a return type
- If you don't create a constructor, the java compiler creates a default constructor.
- A default constructor is one without parameters
- A class can have more than one constructor

```
classname(){
 }
```

eg:

Employee(){ }

Example for constructor

```
class Employee{
      String name;
     int salary;

Employee(){
   name =“Ram”;
   salary = 1000;
}
 void getDetails(){
System.out.print(“name “+name);
System.out.print(“salary”+salary);
}
String greet(String msg){
 return “welcome”+msg;
 }}
```

```
class Test{
public static void main(String args[]){
      Employee  e = new Employee();
      e.name=“Ram”;
      e.salary =1000;
      e.getDetails();
      String v = e.greet(“java”);
      System.out.print(v);

      Employee  e1 = new Employee();
      e1.name=“John”;
      e1.salary =2000;
      e1.getDetails();
       System.out.print(e1.greet(“back”));
      }}
```

o/p same for both objects

Constructor with parameters

- Use constructor with parameters for different objects to initialize different values

eg.

```
Employee(String n,int s ){
        name = n;
        salary =s;
    }
```

Pass the parameters while creating the object

- ***Employee e = new Employee("Ram",1000);***
- ***Employee e1 = new Employee("Tom",2000);***

Example for para constructor

```
class Employee{
      String name;
    int salary;

Employee(String n, int s){
  name = n;
  salary = s;
}
 void getDetails(){
System.out.print("name "+name);
System.out.print("salary"+salary);
}
String greet(String msg){
 return "welcome"+msg;
 }
```

```
class Test{

public static void main(String args[]){

     Employee e = new   Employee ("Ram",1000);
     e.getDetails();
     String v = e.greet("back");
     System.out.print(v);

     Employee e1 = new Employee("Tom",2000);

     e1.getDetails();
      System.out.print(e1.greet("home"));
     }
}
```

What if instance variable name and local variable name are same?

(ie)

```
String name; int salary;
Employee(String name, int salary){
   name = name;
   salary = salary;
 }
```

- **The local variable hides the instance variable**
- **The instance variables will be initialised to their default values only.**

Example for Local variable hiding instance variable

```
class Employee{
      String name;
     int salary;

Employee(String name, int salary){
   name = name;
   salary = salary;
}
 void getDetails(){
System.out.print("name "+name);
System.out.print("salary"+salary);
}
String greet(String msg){
 return "welcome"+msg;
 }}
```

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

      Employee e = new   Employee ("Ram",1000);
      e.getDetails();
      String v = e.greet("back");
      System.out.print(v);

      Employee e1 = new Employee("Tom",2000);

      e1.getDetails();
       System.out.print(e1.greet("home"));
      }
}
```

o/p name and salary will be null and 0

Use this keyword

- **'this' refers to the current object.**

(ie)

```
String name; int salary;
Employee(String name, int salary){
   this.name = name;
   this.salary = salary;
 }
```

Example

```
class Employee{
      String name;
     int salary;

Employee(String name, int salary){
   this.name = name;
   this.salary = salary;
}
 void getDetails(){
System.out.print("name "+name);
System.out.print("salary"+salary);
}
String greet(String msg){
 return "welcome"+msg;
 }}
```

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

     Employee e = new   Employee
     ("Ram",1000);
     e.getDetails();
     String v = e.greet("back");
     System.out.print(v);
}
}
```

o/p name and salary will be Ram and 1000

Summary

- What is a Class?
- Creating objects
- Adding Methods to a class
- Adding Constructor to a class
- Use of parameterised constructor
- Use of this keyword

Thank You