UNITED
GROUP OF INSTITUTIONS

Web Technology (KCS-602) Unit 1

Prepared By

Abhishek Kesharwani

Assistant Professor,UCER Naini,Allahabad

Index

Object in Java

- **Object is an instance of a class.** Class is a template or blueprint from which objects are created. So object is the instance(result) of a class.

An object has three characteristics:

- **state:** represents data (value) of an object.
- **behavior:** represents the behavior (functionality) of an object such as deposit, withdraw etc.
- **identity:** Object identity is typically implemented via a unique ID. The value of the ID is not visible to the external user. But,it is used internally by the JVM to identify each object uniquely.

Class in Java

- Class is a template or blueprint from which objects are created.

A class in java can contain:

- **data member**
- **method**
- **constructor**
- **block**
- **class and interface**

Syntax to declare a class:

```
class <class_name>
{
    data member;
    method;
}
```

Constructor in Java

- **Constructor in java** is a *special type of method* that is used to initialize the object.
- Java constructor is *invoked at the time of object creation*.
- It constructs the values i.e. provides data for the object that is why it is known as constructor.

Rules for creating java constructor

There are basically two rules defined for the constructor.

- Constructor name must be same as its class name
- Constructor must have no explicit return type

Types of java constructors

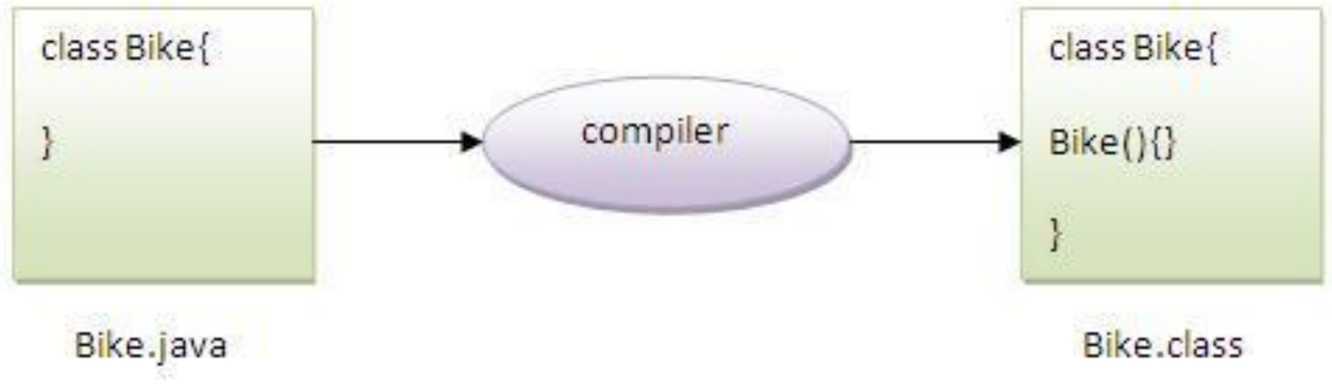
There are two types of constructors:

- Default constructor (no-arg constructor)
- Parameterized constructor

Example of default constructor

- In this example, we are creating the no-arg constructor in the Bike class. It will be invoked at the time of object creation.

```
class Bike1{
Bike1(){System.out.println("Bike is created");}
public static void main(String args[]){
Bike1 b=new Bike1();
}}
```



```
class Bike{

}
```

```
class Bike{

Bike(){}

}
```

Rule: If there is no constructor in a class, compiler automatically creates a default constructor.

Example of parameterized constructor

```
class Student4{
    int id;
    String name;

    Student4(int i,String n){
    id = i;
    name = n;
    }
    void display(){System.out.println(id+" "+name);}

    public static void main(String args[]){
    Student4 s1 = new Student4(111,"Karan");
    Student4 s2 = new Student4(222,"Aryan");
    s1.display();
    s2.display();
   }
}
```

Constructor Overloading in Java

- Constructor overloading is a technique in Java in which a class can have any number of constructors that differ in parameter lists.
- The compiler differentiates these constructors by taking into account the number of parameters in the list and their type.

```
class Student5{
    int id;
    String name;
    int age;
    Student5(int i,String n){
    id = i;
    name = n;
    }
    Student5(int i,String n,int a){
    id = i;
    name = n;
    age=a;
    }
    void display(){System.out.println(id+" "+name+" "+age);}

    public static void main(String args[]){
    Student5 s1 = new Student5(111,"Karan");
    Student5 s2 = new Student5(222,"Aryan",25);
    s1.display();
    s2.display();
   }
}
```

Interview Questions Asked in different Company

- How many constructors do we create in java in a single class?
- What is constructor chaining?

Java Copy Constructor

- There is no copy constructor in java. But, we can copy the values of one object to another like copy constructor in C++.
- There are many ways to copy the values of one object into another in java. They are:
- By constructor
- By assigning the values of one object into another

```
class Student6{
   int id;
   String name;
   Student6(int i,String n){
   id = i;
   name = n;
   }

   Student6(Student6 s){
   id = s.id;
   name =s.name;
   }
   void display(){System.out.println(id+" "+name);}

   public static void main(String args[]){
   Student6 s1 = new Student6(111,"Karan");
   Student6 s2 = new Student6(s1);
   s1.display();
   s2.display();
  }
}
```

Java static method

If you apply static keyword with any method, it is known as static method.

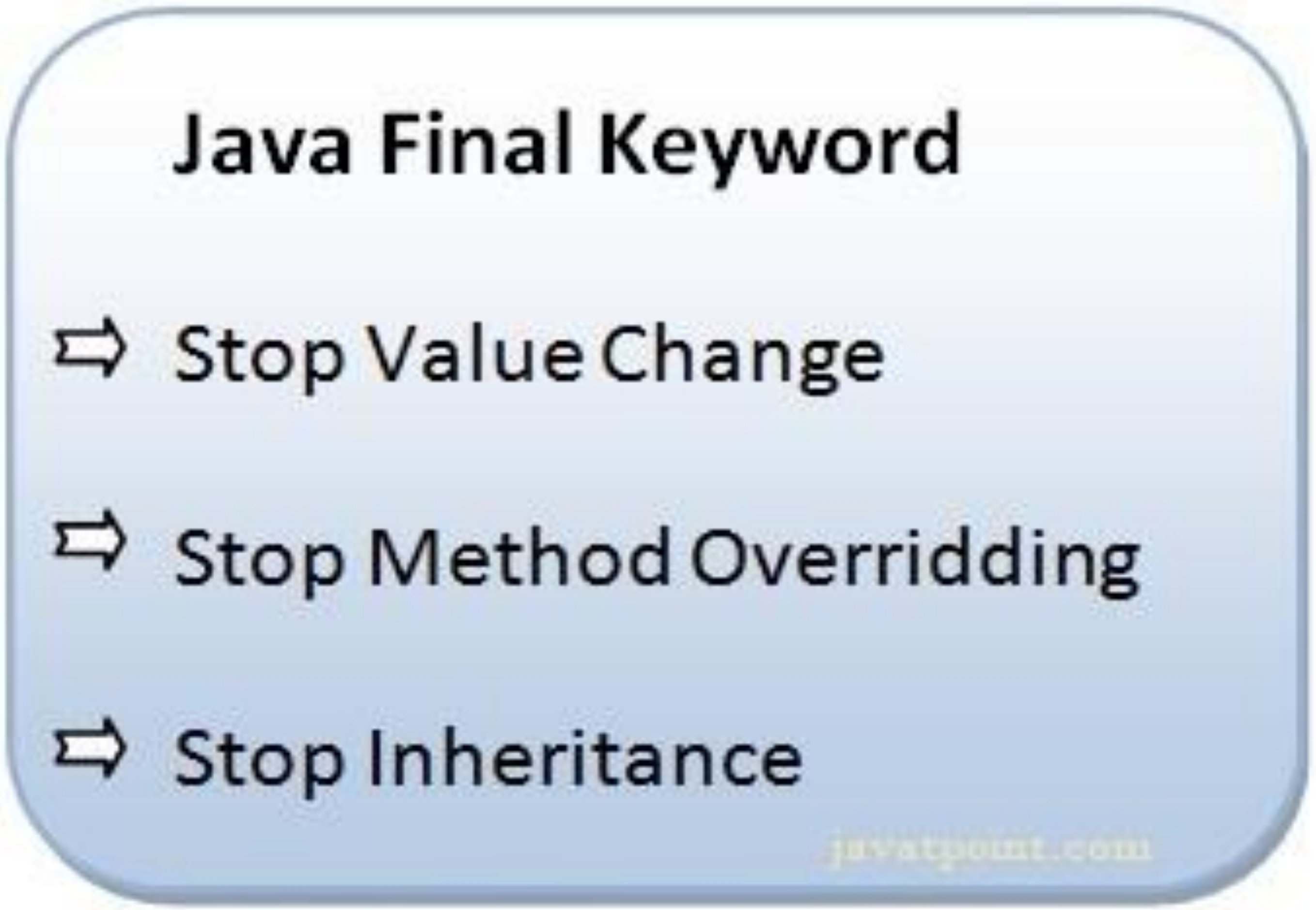
- A static method belongs to the class rather than object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- static method can access static data member and can change the value of it.

Final Keyword In Java

The **final keyword** in java is used to restrict the user. The java final keyword can be used in many context. Final can be:

- variable
- method
- class

- The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable.
- It can be initialized in the constructor only.
- The blank final variable can be static also which will be initialized in the static block only.



Java Final Keyword

⇨ Stop Value Change

⇨ Stop Method Overridding

⇨ Stop Inheritance

If you make any variable as final, you cannot change the value of final variable(It will be constant).

```
class Bike9{
 final int speedlimit=90;//final variable
 void run(){
  speedlimit=400;
 }
 public static void main(String args[]){
 Bike9 obj=new  Bike9();
 obj.run();
 }
}//end of class
```

Output:Compile Time Error

Java final method

If you make any method as final, you cannot override it.

```
class Bike{
 final void run(){System.out.println("running");}
}

class Honda extends Bike{
  void run(){System.out.println("running safely with 100kmph
   ");}

  public static void main(String args[]){
  Honda honda= new Honda();
  honda.run();
  }
}
```

Output:Compile Time Error

Java final class

If you make any class as final, you cannot extend it.

```
final class Bike{}

class Honda1 extends Bike{
 void run(){System.out.println("running safely with 100k
   mph");}

 public static void main(String args[]){
 Honda1 honda= new Honda();
 honda.run();
 }
}
```

Output:Compile Time Error

- A final variable that is not initialized at the time of declaration is known as blank final variable.
- If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful.

Can we initialize blank final variable?

Yes, but only in constructor. For example:

```
class Bike10{
  final int speedlimit;//blank final variable

  Bike10(){
  speedlimit=70;
  System.out.println(speedlimit);
  }

  public static void main(String args[]){
    new Bike10();
 }
}
```