BCA – 401: Java Programming

Rahul Kumar Singh

In today's Class we have discussed on final variables, final methods and final classes in Java and abstract method & abstract class in java.

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 and class. The final keyword is used to indicate that a variable, method, or class cannot be modified or extended.

Once any entity (variable, method or class) is declared final, it can be assigned only once. That is,

- the final variable cannot be re-initialized with another value.
- the final method cannot be overridden.
- the final class cannot be extended.

In Java we use final variable to create constant variable, final method to prevent method overriding and final class to prevent inheritance.

Java final variable:-

When a variable is declared as final, its value cannot be changed once it has been initialized. This is useful for declaring constants or other values that should not be modified.

Example:-

```
 class Bike
{
  final int speedlimit=90;           //final variable
  void run()
{
   speedlimit=110;
   System.out.println("Speed Limit: " + speedlimit);
  }
  public static void main(String args[])
{
  Bike obj=new  Bike();
  obj.run();
  }
 }
```

Output:-

```
Compile Time Error
error: cannot assign a value to final variable speedlimit
 speedlimit=400;
 ^1 error
```

}

Java final method:-

When a method is declared as final, it cannot be overridden by a subclass. If we make any method as final, we cannot override it. This is useful for methods that are part of a class's public API and should not be modified by subclasses.

Example:-

```
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
error: run() in Honda cannot override run() in Bike
ly with 100kmph");}
 ^ 1 error
 overridden method is final
```

Java final class:-

When a class is declared as final, it cannot be extended by a subclass. This is useful for classes that are intended to be used as is and should not be modified or extended.

If you make any class as final, you cannot extend it. The final class cannot be inherited by another class.

Example:-

```
final class Bike
{
 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:-

```
error: cannot inherit from final Bike
 class Honda extends Bike{
 ^1 error
```

Abstraction in Java:-

Abstraction is a process of hiding the implementation details and showing only functionality to the user.

Another way, it shows only essential things to the user and hides the internal details, for example, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.

Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction:-

There are two ways to achieve abstraction in java.

- Abstract class (0 to 100%)
- Interface (100%)

Abstract class in Java:-

A class which is declared as abstract is known as an abstract class. It can have abstract and non-abstract methods. It needs to be extended and its method implemented. It cannot be instantiated.

Rules for Java abstract class:-

- An abstract class must be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- It cannot be instantiated.

- It can have constructors and static methods also.
- It can have final methods which will force the subclass not to change the body of the method.
- If there is an abstract method in a class, that class must be abstract.
- If you are extending an abstract class that has an abstract method, you must either provide the implementation of the method or make this class abstract.

Example of abstract class:-

```
abstract class A

{

}
```

Abstract Method in Java:-

A method which is declared as abstract and does not have implementation is known as an abstract method.

Example of abstract method:-

```
abstract void printStatus();   //no method body and abstract
```

Example of Abstract class that has an abstract method:-

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

```
abstract class Bike
{
 abstract void run();
}
class Honda extends Bike
{
void run()
{
	System.out.println("running safely");
}
public static void main(String args[])
{
	Bike obj = new Honda();
	obj.run();
}
}
```

Output:-

```
running safely
```

Understanding the real scenario of Abstract class:-

In this example, Shape is the abstract class, and its implementation is provided by the Rectangle and Circle classes.

Mostly, we don't know about the implementation class (which is hidden to the end user), and an object of the implementation class is provided by the factory method.

A factory method is a method that returns the instance of the class.

In this example, if we create the instance of Rectangle class, draw() method of Rectangle class will be invoked.

```
abstract class Shape

{

abstract void draw();

}

class Rectangle extends Shape

{

void draw()

{

System.out.println("drawing rectangle");

}

}
```

```
class Circle extends Shape
{
void draw()
{
System.out.println("drawing circle");
}
}
class TestAbstraction
{
public static void main(String args[])
{
Shape s=new Circle();
s.draw();
}
}
```

Output:-

```
drawing circle
```

Another example of Abstract class in java:-

```
abstract class Bank
```

```
{
abstract int getRateOfInterest();
}
class SBI extends Bank
{
int getRateOfInterest()
{
return 7;
}
}
class PNB extends Bank
{
int getRateOfInterest()
{
return 8;
}
}
class TestBank
{
public static void main(String args[])
{
Bank b;
b=new SBI();
System.out.println("Rate of Interest is: " + b.getRateOfInterest()+" %");
b=new PNB();
System.out.println("Rate of Interest is: " + b.getRateOfInterest()+" %");
}
}
```

Output:-

```
Rate of Interest is: 7 %
Rate of Interest is: 8 %
```

Abstract class having constructor, data member and methods:-

An abstract class can have a data member, abstract method, method body (non-abstract method), constructor, and even main() method.

Example of an abstract class that has abstract and non-abstract methods:-

```
abstract class Bike
```

```
{
   Bike()
   {
        System.out.println("bike is created");
   }
   abstract void run();
   void changeGear()
   {
        System.out.println("gear changed");
    }
 }
```

//Creating a Child class which inherits Abstract class

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

//Creating a Test class which calls abstract and non-abstract methods

```
 class TestAbstraction2
{
 public static void main(String args[])
{
  Bike obj = new Honda();
  obj.run();
  obj.changeGear();
 }
}
```

Output:-

```
bike is created
running safely
gear changed
```