Final Keyword In Java – Final variable, Method and Class

BY CHAITANYA SINGH | FILED UNDER: OOPS CONCEPT

In this tutorial we will learn the usage of final keyword. final keyword can be used along with variables, methods and classes. We will cover following topics in detail.

1) final variable
2) final method
3) final class

1) final variable

final variables are nothing but constants. We cannot change the value of a final variable once it is initialized. Lets have a look at the below code:

```
class Demo{

   final int MAX_VALUE=99;
   void myMethod(){
      MAX_VALUE=101;
   }
   public static void main(String args[]){
      Demo obj=new  Demo();
      obj.myMethod();
   }
}
```

Output:

```
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
        The final field Demo.MAX_VALUE cannot be assigned

        at beginnersbook.com.Demo.myMethod(Details.java:6)
        at beginnersbook.com.Demo.main(Details.java:10)
```

We got a compilation error in the above program because we tried to change the value of a final variable “MAX_VALUE”.

Note: It is considered as a good practice to have constant names in UPPER CASE(CAPS).

Blank final variable

A final variable that is not initialized at the time of declaration is known as **blank final variable**. We **must initialize the blank final variable in**

constructor of the class otherwise it will throw a compilation error (Error: variable MAX_VALUE might not have been initialized).

This is how a blank final variable is used in a class:

```
class Demo{
   //Blank final variable
   final int MAX_VALUE;

   Demo(){
      //It must be initialized in constructor
      MAX_VALUE=100;
   }
   void myMethod(){
      System.out.println(MAX_VALUE);
   }
   public static void main(String args[]){
      Demo obj=new  Demo();
      obj.myMethod();
   }
}
```

Output:

```
100
```

Whats the use of blank final variable?

Lets say we have a Student class which is having a field called Roll No. Since Roll No should not be changed once the student is registered, we can declare it as a final variable in a class but we cannot initialize roll no in advance for all the students(otherwise all students would be having same roll no). In such case we can declare roll no variable as blank final and we initialize this value during object creation like this:

```
class StudentData{
   //Blank final variable
   final int ROLL_NO;

   StudentData(int rnum){
      //It must be initialized in constructor
      ROLL_NO=rnum;
   }
   void myMethod(){
      System.out.println("Roll no is:"+ROLL_NO);
   }
   public static void main(String args[]){
      StudentData obj=new  StudentData(1234);
      obj.myMethod();
   }
}
```

Output:

```
Roll no is:1234
```

More about blank final variable at StackOverflow and Wiki.

Uninitialized static final variable

A static final variable that is not initialized during declaration can only be initialized in static block. Example:

```
class Example{
   //static blank final variable
   static final int ROLL_NO;
   static{
      ROLL_NO=1230;
   }
   public static void main(String args[]){
      System.out.println(Example.ROLL_NO);
   }
}
```

Output:

```
1230
```

2) final method

A final method cannot be overridden. Which means even though a sub class can call the final method of parent class without any issues but it cannot override it.

Example:

```
class XYZ{
   final void demo(){
      System.out.println("XYZ Class Method");
   }
}

class ABC extends XYZ{
   void demo(){
      System.out.println("ABC Class Method");
   }

   public static void main(String args[]){
      ABC obj= new ABC();
      obj.demo();
   }
}
```

The above program would throw a compilation error, however we can use the parent class final method in sub class without any issues. Lets have a look at this code: This program would run fine as we are not overriding the final method. That shows that final methods are inherited but they are not eligible for overriding.

```
class XYZ{
   final void demo(){
      System.out.println("XYZ Class Method");
   }
}

class ABC extends XYZ{
   public static void main(String args[]){
      ABC obj= new ABC();
      obj.demo();
   }
}
```

Output:

```
XYZ Class Method
```

3) final class

We cannot extend a final class. Consider the below example:

```
final class XYZ{
}

class ABC extends XYZ{
   void demo(){
      System.out.println("My Method");
   }
   public static void main(String args[]){
      ABC obj= new ABC();
      obj.demo();
   }
}
```

Output:

```
The type ABC cannot subclass the final class XYZ
```

Points to Remember:

1) A constructor cannot be declared as final.
2) Local final variable must be initializing during declaration.
3) All variables declared in an interface are by default final.
4) We cannot change the value of a final variable.
5) A final method cannot be overridden.
6) A final class not be inherited.
7) If method parameters are declared final then the value of these parameters cannot be changed.
8) It is a good practice to name final variable in all CAPS.
9) final, finally and finalize are three different terms. finally is used in exception handling and finalize is a method that is called by JVM during garbage collection.