



Final Keyword

Agenda

1 final keyword

final Keyword



Keyword final

- The **final** keyword used in context of behavioral restriction on:
 - variables
 - methods
 - classes
- Using final on variables to make them behave as constants which we have seen in earlier module.
- When a variable is made final – it can be initialized only once either by
 - Declaration and initialization
final int x=10;
 - Using constructor
- System allows you to set the value only once; after which it can't be changed.

Quiz

What will be the output for the below code ?

```
public class Sample {  
    final double pi;  
    public Sample()  
{  
    pi = 3.14;  
}  
    public Sample(double pi)  
{  
    this.pi = pi;  
}  
  
}
```

```
public static void main() {  
    Sample ob = new  
        Sample(22/7)  
  
        System.out.println(ob.  
pi);  
}
```

The Role of the Keyword final in Inheritance

- The **final** keyword has two important uses in the context of a class hierarchy. These uses are highlighted as follows:
- Using final **to Prevent Overriding**
 - While method overriding is one of the most powerful feature of object oriented design, there may be times when you will want to **prevent certain critical methods in a superclass** from being overridden by its subclasses.
 - This can be achieved by declaring such critical methods as final.

Keyword final with methods- Example

```
/* Example for final methods*/
class GBase {
    public final void display(String s){
        System.out.println(s);
    }
}

class Sample extends GBase{
    public void display(String s) {
        System.out.println(s);
    }

    public static void main(String args[]) {
        Sample ob = new Sample();
        ob.display("TRY ME");
    }
}
```

Output:

Compile Time Error : **Cannot override the final method from GBase**

The Role of the Keyword final in Inheritance (Contd.).

Sometimes you will want **to prevent a class from being inherited.**

- This can be achieved by preceding the class declaration with final.
- Declaring a class as final implicitly declares all of its methods as final too.
- It is illegal to declare a class as both abstract and final since an abstract class.

final methods- Example

```
final class GBase {  
    public void display(String s){  
        System.out.println(s);  
    }  
}  
  
class Sample extends GBase{  
    public void display(String s)  
    {  
        System.out.println(s);  
    }  
    public static void main(String args[]) {  
        Sample ob = new Sample();  
        ob.display("TRY ME");  
    }  
}
```

Output: Explain Why this CTE occurs?

Compile Time Error :

The type Sample cannot subclass the final class GBase

Quiz

What will be the output for the below code ?

```
class abstract GBase{  
public final void testBase() {  
System.out.println("Hello World");  
}  
}
```





Thank You

