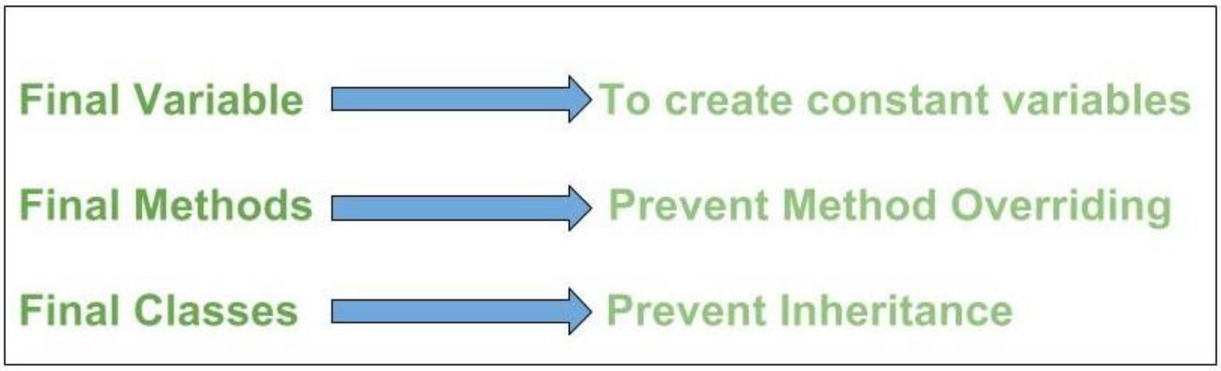
Final Keyword in Java

Java, *final* keyword is applied in various contexts. The *final* keyword is a modifier means the *final* class can't be extended, a variable can't be modified, and a method can't be override it means that an entity cannot later be changed.

It used for the following purposes:



1- *Final variables*

The field declared as final behaves like constant. Means once it is declared, it can't be changed. Before compiling, only once it can be set; after that you can't change its value. Attempt to change in its value lead to exception or compile time error. If the final variable dose not initialize the compiler will throw compile-time error. It could be declared as shown below:

Example1:

public final double radius = 126.45;
public final int PI = 3.145;

The fields which are declared as static, final and public are known as *named constants*. Given below an example of named constant:

Example2:

```
public class Maths{
public static final double x = 2.998E8; }
```

A **final variable** does not need not be initialized at the point of declaration: this is called a **"blank final"** variable.

A blank final instance variable of a class must be assigned at the end of every constructor of the class in which it is declared; similarly, a blank **final static variable must be assigned in a static initializer of the class in which it is declared**: otherwise, a compile-time error occurs in both cases.

Example3:

```
public class Sphere {

 public static final double PI = 3.141592653589793;
 or
public static final double PI = 3.141592653589793;

   public final double radius;
   public final double xpos;
   public final double ypos;
   public final double zpos;

   Sphere(double x, double y, double z, double r) {
       radius = r;
       xpos = x;
       ypos = y;
       zpos = z;
   }

   [...]
}
```

Any attempt to reassign radius, xpos, ypos, or zpos will meet with a compile error. <u>In fact, even if the constructor doesn't set a final variable, attempting to set it outside the constructor will result in a compilation error.</u>

```
public class Test {

   public static void main(String args[]) {

      final int i = 10;

      i = 30; // Error because i is final.

   }

}
```

2- Final methods

You can declare some or all of a class's methods final. A final method can't be overridden by subclasses but it is still could be overloaded. This is used to prevent unexpected behavior from a subclass altering a method that may be crucial to the function or consistency of the class.

A final method within a class could be declared in java as shown:

```
public class MyClass {
   public final void myFinalMethod() {...}
}
```

Example 4 (Trace):

```
public class FinalMethodExample {
  public final void display(){
    System.out.println("Hello welcome to Tutorialspoint");    }
  public static void main(String args[]){
    new FinalMethodExample().display();    }
  class Sample extends FinalMethodExample{
    public void display(){
      System.out.println("hi");
    }    }    }
```

Output

```
FinalMethodExample.java:12: error: display() in FinalMethodExample.Sample
cannot override display() in FinalMethodExample
public void display(){
          ^
overridden method is final
1 error
```

You might wish to make a method final if it has an implementation that should not be changed and it is critical to the consistent state of the object.

Methods called from constructors should generally be declared final. If a constructor calls a non-final method, a subclass may redefine that method with surprising or undesirable results.

Final method arguments

You can also declare method's argument as final. The final argument can't be modified by the method directly.

3- ***Final classes***

A **final** **class** cannot be extended. This is done for reasons of security (المنية والحماية) and efficiency (الكفاءة و الفعالية). Accordingly, many of the Java standard library classes are final, for example java.lang.System and java.lang.String. **All methods in a final class are implicitly final.**

The final class is declared in java as shown:

If we say:

```
public  final class A {}
```

this means that A cannot be further extended or subclassed. This feature has a big implication. It allows control over a class, so that no one can subclass the class and possibly introduce anomalous behavior (سلوك غير طبيعي). For example, java.lang.String is a final class. This means, for example, that I can't subclass String and provide my own length() method that does something very different from returning the string length.

H.W. Answer the following:

1- Constructor cant be final …why?
2- Could you define a final class without final methods?
3- Could you use a set method to initialize the final attributes within a class.
4- Does the final method process the final variables?
5- Explain the blank final.
6- Discuss that " The value of a final variable is not necessarily known at compile time"