The ’*final*’ keyword in Java

The '*final*' keyword is a non-access modifier used for classes, attributes or variables, and methods, which makes them non-changeable, meaning impossible to inherit or override.

Most common uses of '*final*' keyword are:

1. '*final*' variables
2. '*final*' methods
3. '*final*' classes
4. '*final*' parameters

*Main.java

```java
package finalKeyword;

import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {

        // constant
        final double PI = 3.14;
        PI = 4.0; // invalid, final variable PI cannot be reassigned

        // object with changeable state
        final ArrayList<String> planets = new ArrayList<>();
        planets.add("Mercury"); // valid
        planets.add("Venus"); // valid
        planets.add("Earth"); // valid
        planets.remove(0); // valid
        planets = new ArrayList<>(); // invalid, the planets variable
                                     // itself cannot be reassigned to
                                     // reference a different object.
    }

}
```

1. '*final*' variables:

A '*final*' variable is a variable that cannot be reassigned to reference a different object.

It can still be modified if it refers to an object whose state can be changed.

It is commonly used to declare constant values like 'Pi', whose value is always 3.14 in decimal.

```java
class User {
    private String password;
    private String userName;
    private String phoneNumber;

    // final method
    public final void setPassword(String password) {
        // code for setting password
    }

    // static method
    public static void setUsername(String userName) {
        // code for setting username
    }

    // static method
    public static void setPhone(String phoneNumber) {
        // code for setting phone number
    }

}

class AdminUser extends User {
    @Override
    public void setPassword(String password) { // invalid, cannot override final method from User
        // new code for setting password
    }

    public void setUsername(String userName) { // This instance method cannot override the
                                               // static method from User
        // new code for setting username
    }

    // static method
    public static void setPhone(String phoneNumber) { // valid, the static method is for AdminUser class
        // code for setting phone number
    }
}
```

2. '*final*' methods:

A '*final*' method is a method that cannot be overridden by subclasses.

This can be used to prevent subclasses from changing the behaviour of the method.

Methods that are declared static in superclass are also implicitly final for the instances of subclass, however the subclass can have its own static method with the same method signature as of superclass……

```java
public class Main {
    public static void main(String[] args) {

    }
}

class User {
    private String password;
    private String email;

    public final void login(String email, String password) {
        // code for logging in using email and password
    }

}

class AdminUser extends User {
    private String phoneNumber;

    public void login(String phoneNumber) {
        // code for logging in using phone number
    }
}
```

However, a subclass can override a final superclass method if its method has a different method signature.

For example, different parameters.

```java
public class Main {

    public static void main(String[] args) {

    }

}

final class SuperClass {

    public void doSuperClassStuff() {
        // code to do superclass stuffs
    }

    public final void doOtherSuperClassStuff() {
        // code to do superclass stuffs
    }
}

class SubClass extends SuperClass { // invalid, the type SubClass cannot
                                    // subclass the final class SuperClass
    @Override
    public void doSuperClassStuff() { // invalid, cannot override method
                                      // in final class

    }

    @Override
    public void doOtherSuperClassStuff() { // invalid, cannot override final  method

    }
}

```

3. '*final*' classes:

A '*final*' class is a class that cannot be subclassed.

This can be used to prevent other classes from extending the class and potentially changing its behaviour.

All methods inside a final class are implicitly final. This means that they cannot be overridden by subclasses of the class.

```java
package finalParameter;

public class Main {

    public static void main(String[] args) {
    }

    public static int incrementAge(final int age) {

        age = age + 1; // invalid, age is final

        return age;
    }

}

```

4. '*final*' parameters:

A '*final*' parameter is a method parameter that is marked as final.

This means that the value of the parameter cannot be changed within the method.

Below are all additional concepts about the '*final*' keyword in Java.

1. '*final*' variables must be initialized when they are declared, or in a static initializer block if they are static variables. They cannot be left uninitialized and assigned a value later.

2. '*final*' variables can be either instance variables (belonging to a specific object) or static variables (belonging to a class).

3. '*final*' variables are often used to create constants, which are values that do not change at runtime.

4. '*final*' methods can be overridden by anonymous classes, but they cannot be overridden by named subclasses.

5. '*final*' classes cannot have any subclasses at all. This means that they cannot have any abstract methods, since abstract methods must be implemented by subclasses.

6. '*final*' classes can still have instance variables and non-final methods, just like any other class.