

Final Keyword In Java

Java final variable

1. cannot change the value of final variable (It will be constant).
2. It can be initialized only in constructor.
3. A static final variable that is not initialized at the time of declaration is known as static blank final variable. It can be initialized only in static block.

Java final method

1. you cannot override it.
2. final method is inherited but you cannot override it.
3. If you declare any parameter as final, you cannot change the value of it.
4. constructor can't be made final because is never inherited.

Java final class

1. you cannot extend it.
-

Java static keyword

The **static keyword** in [Java](#) is used for memory management mainly. **Java static property is shared to all objects.**

Java static variable

1. The static variable gets memory only once in the class area at the time of class loading.

Java static method

- A static method belongs to the class rather than the object of a class.
- A static method can be invoked without the need for creating an instance of a class.
- A static method can access static data member and can change the value of it.
- The static method cannot use non static data member or call non-static method directly.
- this and super cannot be used in static context.

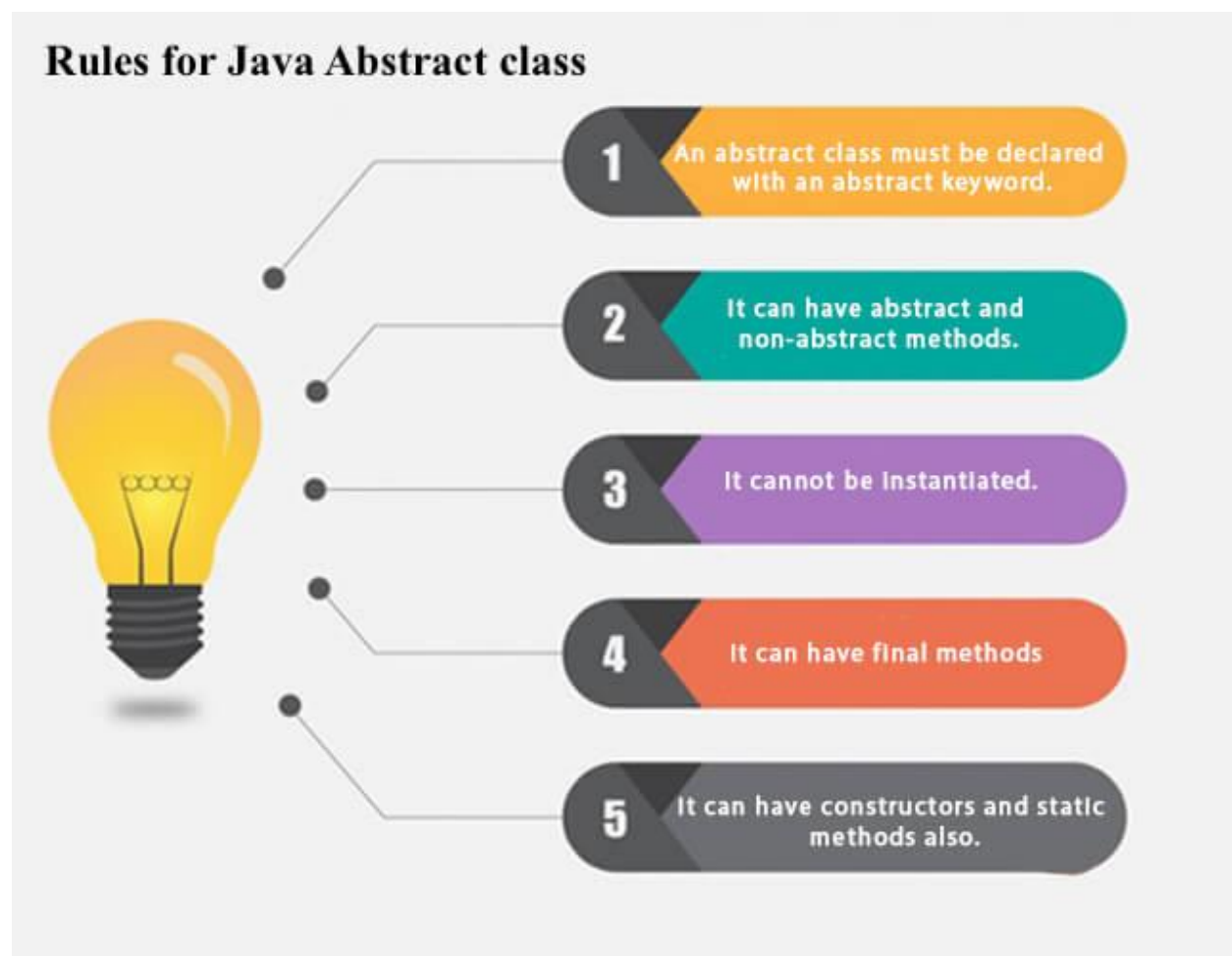
Why is the Java main method static?

It is because the object is not required to call a static method. If it were a non-static method, JVM creates an object first then call main() method that will lead the problem of extra memory allocation.

Java static block

- Is used to initialize the static data member.
- It is executed before the main method at the time of classloading.

Abstract class in Java : Factory Design Pattern



Interface in Java

1. Since [Java 8](#), interface can have default and static methods.
2. An interface which has no member is known as a marker or tagged interface, for example, [Serializable](#), Cloneable, Remote,

The Java compiler adds `public` and `abstract` keywords before the interface method. Moreover, it adds `public`, `static` and `final` keywords before data members

3. Since Java 8, we can have method body in interface. But we need to make it default method
4. we can have static method in interface

```
interface Drawable{  
    void draw();  
    default void msg(){System.out.println("default method");}  
    static int cube(int x){return x*x*x;}  
}
```

Nested Interface in Java

```
interface printable{  
    void print();  
    interface MessagePrintable{  
        void msg();  
    }  
}
```

Access Modifiers in Java

Access Modifier	within class	within package	outside package by subclass only	outside package
Private	Y	N	N	N
Default	Y	Y	N	N
Protected	Y	Y	Y	N
Public	Y	Y	Y	Y

Object Cloning in Java

The **java.lang.Cloneable interface** must be implemented by the class whose object clone we want to create. If we don't implement Cloneable interface, clone() method generates **CloneNotSupportedException**.

protected Object clone() throws CloneNotSupportedException

Object.clone() doesn't invoke any constructor so we don't have any control over object construction.

Object.clone() supports only **shallow copying** but we will need to override it if we need **deep cloning**.

shallow copying : only primitive data types get copied.

Java Strictfp Keyword

Java strictfp keyword ensures that you will get the same result on every platform if you perform operations in the floating-point variable. The precision may differ from platform to platform that is why java programming language have provided the strictfp keyword, so that you get same result on every platform. So, now you have better control over the floating-point arithmetic.

1. The strictfp keyword can be applied on methods, classes and interfaces.
 2. The strictfp keyword **cannot** be applied on abstract methods, variables or constructors.
-

Java String compare

1. By equals() method : **authentication**
2. By == operator : **sorting**
3. By compareTo() method : **reference matching**

```
class Teststringcomparison3{
    public static void main(String args[]){
        String s1="Sachin";
        String s2="Sachin";
        String s3=new String("Sachin");
        System.out.println(s1==s2);//true (because both refer to same instance)
        System.out.println(s1==s3);//false(because s3 refers to instance created in non-pool)
    }
}
```

S1 and s2 points to same string in Pool

S3 refers to new string from non-pool.

Java Inner Classes

Non-static nested classes are known as inner classes.

Type	Description
<u>Member Inner Class</u>	A class created within class and outside method.
<u>Anonymous Inner Class</u>	A class created for implementing interface or extending class. Its name is decided by the java compiler.
<u>Local Inner Class</u>	A class created within method.
<u>Static Nested Class</u>	A static class created within class.
<u>Nested Interface</u>	An interface created within class or interface.

Java Member inner class

A non-static class that is created inside a class but outside a method is called member inner class.

```
class TestMemberOuter1{
    private int data=30;
    class Inner{
        void msg(){System.out.println("data is "+data);}
    }
    public static void main(String args[]){
        TestMemberOuter1 obj=new TestMemberOuter1();
        TestMemberOuter1.Inner in=obj.new Inner();
        in.msg();
    }
}
```

Java Anonymous inner class

```
abstract class Person{
    abstract void eat();
}
class TestAnonymousInner{
    public static void main(String args[]){
        Person p=new Person(){
            void eat(){System.out.println("nice fruits");}
        };
        p.eat();
    }
}
```

Java Local inner class

1. A class i.e. created inside a method is called local inner class in java.

1. *Local inner class cannot be invoked from outside the method.*
2. *access the non-final local variable in local inner class.*

Java static nested class

- It can access static data members of outer class including private.
- Static nested class cannot access non-static (instance) data member or method.

```
class TestOuter1{  
    static int data=30;  
    static class Inner{  
        void msg(){System.out.println("data is "+data);}  
    }  
    public static void main(String args[]){  
        TestOuter1.Inner obj=new TestOuter1.Inner();  
        obj.msg();  
    }  
}
```

Java Nested Interface

- Nested interface must be public if it is declared inside the interface

```
interface interface_name{  
    ...  
    interface nested_interface_name{  
        ...  
    }  
}
```

- it can have any access modifier if declared within the class.

```
class class_name{  
    ...  
    interface nested_interface_name{  
        ...  
    }  
}
```

- Nested interfaces are declared static implicitly.

```

interface Showable{
    void show();
    interface Message{
        void msg();
    }
}

class TestNestedInterface1 implements Showable.Message{
    public void msg(){System.out.println("Hello nested interface");}

    public static void main(String args[]){
        Showable.Message message=new TestNestedInterface1();//upcasting here
        message.msg();
    }
}

```

Can we define a class inside the interface?

Yes, If we define a class inside the interface, java compiler creates a static nested class.

```

interface M{
    class A{}
}

```
