Problem Statement in a Static Keyword

Java is used for memory management mainly. We can apply static keyword with variables, methods, blocks and nested classes. The static keyword belongs to the class than an instance of the class.

The static can be:

Variable (also known as a class variable)

Method (also known as a class method)

Block

Nested class

The excessive use of the "static" keyword in a software project is causing maintainability, testing, and scalability issues.

Key Challenges:

Maintainability: Complex code due to static elements, making it hard to modify and understand.

Testing: Difficulty in testing with static elements, hindering unit and integration testing.

Scalability: Limited scalability due to static elements' rigidity.

Objectives:

Identify problematic use cases.

Improve code maintainability.

Enhance testing strategies.

Improve scalability.

Expected Outcomes:

Better code readability.

Improved testing procedures.

Enhanced scalability.

Reduced bugs.

Approach:

Code review, refactoring, and exploring alternatives to static elements.

Timeline:

Based on the project's complexity, prioritize and gradually refactor code.

Problem Statement in a final Keyword

The final keyword in java is used to restrict the user. The java final keyword can be used in many context.

Final can be:

variable

method

class

The final keyword can be applied with the variables, a final variable that have no value it is called blank final variable or uninitialized final variable. It can be initialized in the constructor only.

The blank final variable can be static also which will be initialized in the static block only.

We will have detailed learning of these. Let's first learn the basics of final keyword.

Problem Statement in a this Keyword

this keyword in Java

There can be a lot of usage of Java this keyword. In Java, this is a reference variable that refers to the

current object.

Usage of Java this keyword

Here is given the 6 usage of java this keyword.

this can be used to refer current class instance variable.

this can be used to invoke current class method (implicitly)

this() can be used to invoke current class constructor.

this can be passed as an argument in the method call.

this can be passed as argument in the constructor call.

this can be used to return the current class instance from the method.

Suggestion: If you are beginner to java, lookup only three usages of this keyword.

Problem Statement in a Super Keyword

Usage of Java this keyword

Here is given the 6 usage of java this keyword.

this can be used to refer current class instance variable.

this can be used to invoke current class method (implicitly)

this() can be used to invoke current class constructor.

this can be passed as an argument in the method call.

this can be passed as argument in the constructor call.

this can be used to return the current class instance from the method.

Suggestion: If you are beginner to java, lookup only three usages of this keyword.

1. Use of super Keyword in Java

It is majorly used in the following contexts as mentioned below:

Use of super with variables

Use of super with methods

Use of super with constructors

Use of super with Variables

This scenario occurs when a derived class and base class has the same data members.

In that case, there is a possibility of ambiguity for the JVM.

2. Use of super with Methods

This is used when we want to call the parent class method.

So whenever a parent and child class have the same-named methods then to resolve ambiguity we use the super keyword.

3. Use of super with constructors

The super keyword can also be used to access the parent class constructor. One more important thing is that ‘super’ can call both parametric as well as non-parametric constructors depending on the situation.

Important Points to Remember while using Super Keyword

Call to super() must be the first statement in the Derived(Student) Class constructor because if you think about it, it makes sense that the superclass has no knowledge of any subclass, so any initialization it needs to perform is separate from and possibly prerequisite to any initialization performed by the subclass. Therefore, it needs to complete its execution first.

If a constructor does not explicitly invoke a superclass constructor, the Java compiler automatically inserts a call to the no-argument constructor of the superclass. If the superclass does not have a no-argument constructor, you will get a compile-time error. The object does have such a constructor, so if the Object is the only superclass, there is no problem.

Problem Statement in a public Keyword

Java public keyword

A Java public keyword is an access modifier. It can be assigned to variables, methods, constructors, and classes. It is the most non-restricted type of access modifier.

Points to remember

The public access modifier is accessible everywhere. So, we can easily access the public inside and outside the class and package.

If you are overriding any method, overridden method (i.e., declared in the subclass) must not be more restrictive. So, if you assign public to any method or variable, that method or variable can be overridden to sub-class using public access modifier only.

If a program contains multiple classes, at most one class can be assigned as public.

If a class contain a public class, the name of the program must be similar to the public class name.

Problem Statement in a private Keyword

Java private keyword

A Java private keyword is an access modifier. It can be assigned to variables, methods, and inner classes. It is the most restricted type of access modifier.

Points to remember

The private access modifier is accessible only within the same class.

We can't assign private to outer class and interface.

The best use of private keyword is to create a fully encapsulated class in Java by making all the data members of that class private.

If we make any class constructor private, we cannot create the instance of that class from outside the class.

If we are overriding any method, overridden method (i.e., declared in the subclass) must not be more restrictive.

According to the previous point, if we assign a private modifier to any method or variable, that method or variable can be overridden to sub-class using all type of access modifiers. However, still, we can't invoke private method outside the class.

Problem Statement in a default Keyword

Java default keyword

A Java default keyword is an access modifier. If you didn't assign any access modifier to variables, methods, constructors and, classes, by default, it is considered as default access modifier.

Points to remember

The default access modifier is accessible within the package only.

Unlike private and protected, we can create a default outer class by not assigning any access modifier to it. In such a case, it not restricted to take class name similar to a program name.

If you are overriding any method, overridden method (i.e., declared in the subclass) must not be more restrictive. So, the default method or variable can't be allowed to use private access modifier.

Problem Statement in a Protected Keyword

Java protected keyword

A Java protected keyword is an access modifier. It can be assigned to variables, methods, constructors and inner classes.

Points to remember

The protected access modifier is accessible within the package. However, it can also accessible outside the package but through inheritance only.

We can't assign protected to outer class and interface.

If you make any constructor protected, you cannot create the instance of that class from outside the package.

If you are overriding any method, overridden method (i.e., declared in the subclass) must not be more restrictive.

According to the previous point, if you assign protected to any method or variable, that method or variable can be overridden to sub-class using public or protected access modifier only.

Problem Statement in a constructor

In Java, a constructor is a block of codes similar to the method. It is called when an instance of the class is created. At the time of calling constructor, memory for the object is allocated in the memory.

It is a special type of method which is used to initialize the object.

Every time an object is created using the new() keyword, at least one constructor is called.

It calls a default constructor if there is no constructor available in the class. In such case, Java compiler provides a default constructor by default.

There are two types of constructors in Java: no-arg constructor, and parameterized constructor.

Note: It is called constructor because it constructs the values at the time of object creation. It is not necessary to write a constructor for a class. It is because java compiler creates a default constructor if your class doesn't have any.

Rules for creating Java constructor

There are two rules defined for the constructor.

Constructor name must be the same as its class name

A Constructor must have no explicit return type

A Java constructor cannot be abstract, static, final, and synchronized

Note: We can use access modifiers while declaring a constructor. It controls the object creation. In other words, we can have private, protected, public or default constructor in Java.

Types of Java constructors

There are two types of constructors in Java:

Default constructor (no-arg constructor)

Parameterized constructor