Java Modifiers

Topics : JAVA
Written on April 08, 2023

In Java, modifiers are keywords that are used to define the scope and behavior of classes, methods, variables, and other program elements. There are several types of modifiers in Java, including:

1. **Access Modifiers:** These modifiers determine the accessibility of classes, variables, and methods. There are four access modifiers in Java:

- public: A public member can be accessed from anywhere, inside or outside the class.
- private: A private member can only be accessed within the same class.
- protected: A protected member can be accessed within the same class, its subclasses, and classes in the same package.
- default (no keyword): A default member can be accessed within the same package.

Example :

```
public class ExampleClass {
  public int publicVar; // accessible from anywhere
  private int privateVar; // accessible only within this class
  protected int protectedVar; // accessible within this class and its subclasses
  int defaultVar; // accessible within the same package
}

public class Subclass extends ExampleClass {
  public void exampleMethod() {
    publicVar = 1; // OK, accessible from anywhere
    privateVar = 2; // Compile-time error, privateVar is not accessible from this subclass
    protectedVar = 3; // OK, accessible because Subclass is a subclass of ExampleClass
    defaultVar = 4; // OK, accessible because Subclass is in the same package as ExampleClass
  }
}
```

2. **Non-Access Modifiers:** These modifiers do not affect the accessibility of classes, variables, and methods. They provide additional information about the behavior of program elements. There are several non-access modifiers in Java:

- final: A final variable cannot be reassigned, a final method cannot be overridden, and a final class cannot be subclassed.
- static: A static variable or method belongs to the class rather than to an instance of the class.
- abstract: An abstract class or method cannot be instantiated and must be subclassed.
- synchronized: A synchronized method can only be accessed by one thread at a time.
- volatile: A volatile variable is accessed from main memory rather than from a thread's local cache.

Example :

```
public class ExampleClass {
  static int staticVar; // belongs to the class and not to any instance
  final int finalVar = 1; // value cannot be changed once initialized
  abstract void abstractMethod(); // cannot be instantiated, only used as a base for subclasses
  synchronized void synchronizedMethod() { // only one thread can access this method at a time
    // method body here
  }
  transient int transientVar; // will not be serialized when written to a file or transferred over a
network
  volatile int volatileVar; // the value of this variable will never be cached
}

public class ExampleSubclass extends ExampleClass {
  @Override // indicates that this method is intended to override a method in the superclass
  void abstractMethod() {
    // method body here
  }

  @SuppressWarnings("unchecked") // suppresses unchecked warnings from this method
  void exampleMethod() {
    // method body here
  }
}
```

3. **Strictfp Modifier:** This modifier is used to ensure that floating-point calculations are consistent across different platforms.

Modifiers are an important aspect of Java programming because they allow developers to control access to program elements and specify how they behave in different contexts.

There are also annotations, which provide metadata about a class, method, or variable. Annotations do not directly affect the behavior of the program, but they can be used by tools or frameworks to generate code or enforce conventions. Some common annotations include:

- @Override: Indicates that a method is intended to override a method in a superclass or interface.
- @Deprecated: Indicates that a method or class should not be used anymore.
- @SuppressWarnings: Suppresses warnings that would normally be generated by the compiler.