

Launch Multi-File Source-Code Programs

Starting from Java 11, it's possible to run Java programs directly from a single file without the need for compilation. This feature is referred to as "single-file source-code programs" or simply "single-file programs" in Java.

The same feature is extended in the form of "multi-file source-code programs" which will allow developers to run a Java program that has logic spread across multiple source code files.

Before Java 11

Hello

```
public class Hello {  
  
    public static void main(String[] args) {  
        System.out.println("Hello World...");  
    }  
  
}
```

Compilation -> `javac Hello.java`
Execution -> `java Hello`

From Java 11

Hello

```
public class Hello {  
  
    public static void main(String[] args) {  
        System.out.println("Hello World...");  
    }  
  
}
```

Compilation & Execution -> `java Hello.java`

But this feature works only if you have all your code within a single source file

From Java 22

Hello

```
public class Hello {  
  
    public static void main(String[] args) {  
        Greetings.sayHello();  
    }  
  
}
```

Greetings

```
public class Greetings {  
  
    public static String sayHello() {  
        System.out.println("Hello World...");  
    }  
  
}
```

Compilation & Execution -> `java Hello.java`

Though Hello has dependency on Greetings, from Java 22, we can execute the code as it supports multi-file source-code programs

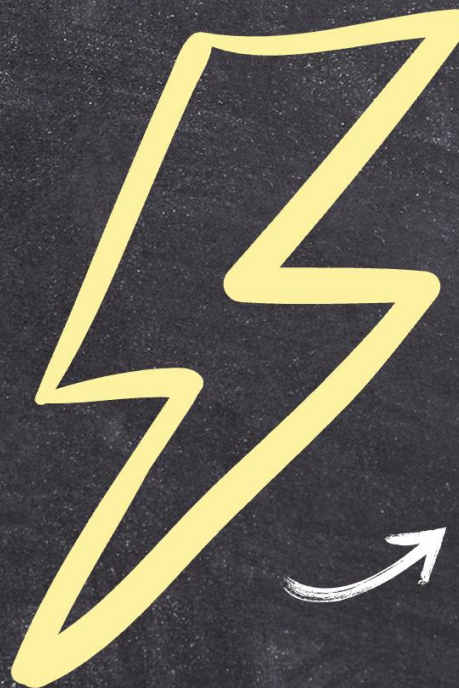
Unnamed Variables & Patterns

1

In certain code snippets, local variables may remain unused, yet deleting them might not be feasible. This could occur due to language constraints or reliance on the side effects of those variables.

2


In situations where deleting an unused variable is either impossible or ill-advised, replacing them with an unnamed variable, denoted by `_`, can serve as a clear and obvious solution.



3

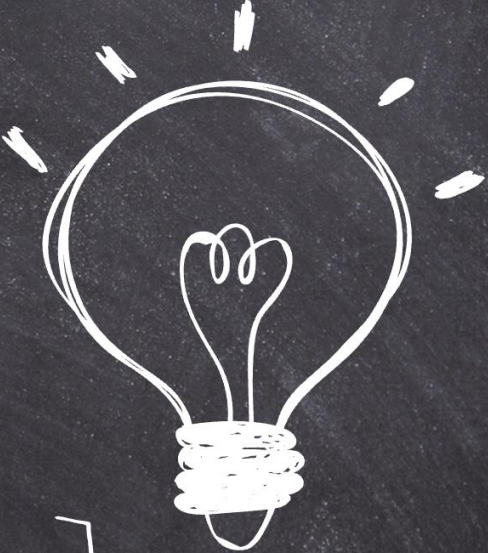
Unnamed variables & patterns would free maintainers from having to understand irrelevant names, and would avoid false positives on non-use from static analysis tools.

Unnamed Variables & Patterns



Starting from Java 22, you can mark unused local variables, patterns, and pattern variables to be ignored by replacing their names (or their types and names) with an underscore, `_`.

These variables and patterns, known as Unnamed variables and patterns, no longer have a name. This feature is beneficial as it reduces the time and effort required to understand a code snippet and could potentially prevent errors in the future.



It's important to note that this feature doesn't apply to instance or class variables. Unnamed variables cannot be passed to methods or assigned values

Unnamed Variables & Patterns

```
public static int countWords(Iterable<String> words) {  
    int totalWords = 0;  
    for (String word : words) {  
        totalWords++;  
    }  
    return totalWords;  
}
```

In the provided example, the unused variable “word” can’t be deleted. So it can be replaced with the unnamed variable syntax which is _

```
public static int countWords(Iterable<String> words) {  
    int totalWords = 0;  
    for (String _ : words) {  
        totalWords++;  
    }  
    return totalWords;  
}
```