

Java Scope

```
Scope.java x
1 package org.studyeasy;
2
3 public class Scope {
4
5     public static void main(String[] args) {
6         for(int i = 0; i < 10; i++) {
7             System.out.println(i);
8         }
9         System.out.println("****");
10
11         /*
12          Here the i ; we get error as soon as loop ends local variable
13          ends in loop.So (int i;)is declared at the beginning
14
15         System.out.println(i);
16         */
17
18         System.out.println(i);
19     }
20 }
21 }
22
```

Here I gives an error and explanation is given in comments.Sol is below.

```
Scope.java x
1 package org.studyeasy;
2
3 public class Scope {
4
5     public static void main(String[] args) {
6         int i;
7         for(i = 0; i < 10; i++) {
8             System.out.println(i);
9         }
10        System.out.println("****");
11
12        /*
13         Here the i ; we get error as soon as loop ends local variable
14         ends in loop.So (int i;)is declared at the beginning
15
16        System.out.println(i);
17        */
18
19        System.out.println(i);
20    }
21 }
22
```

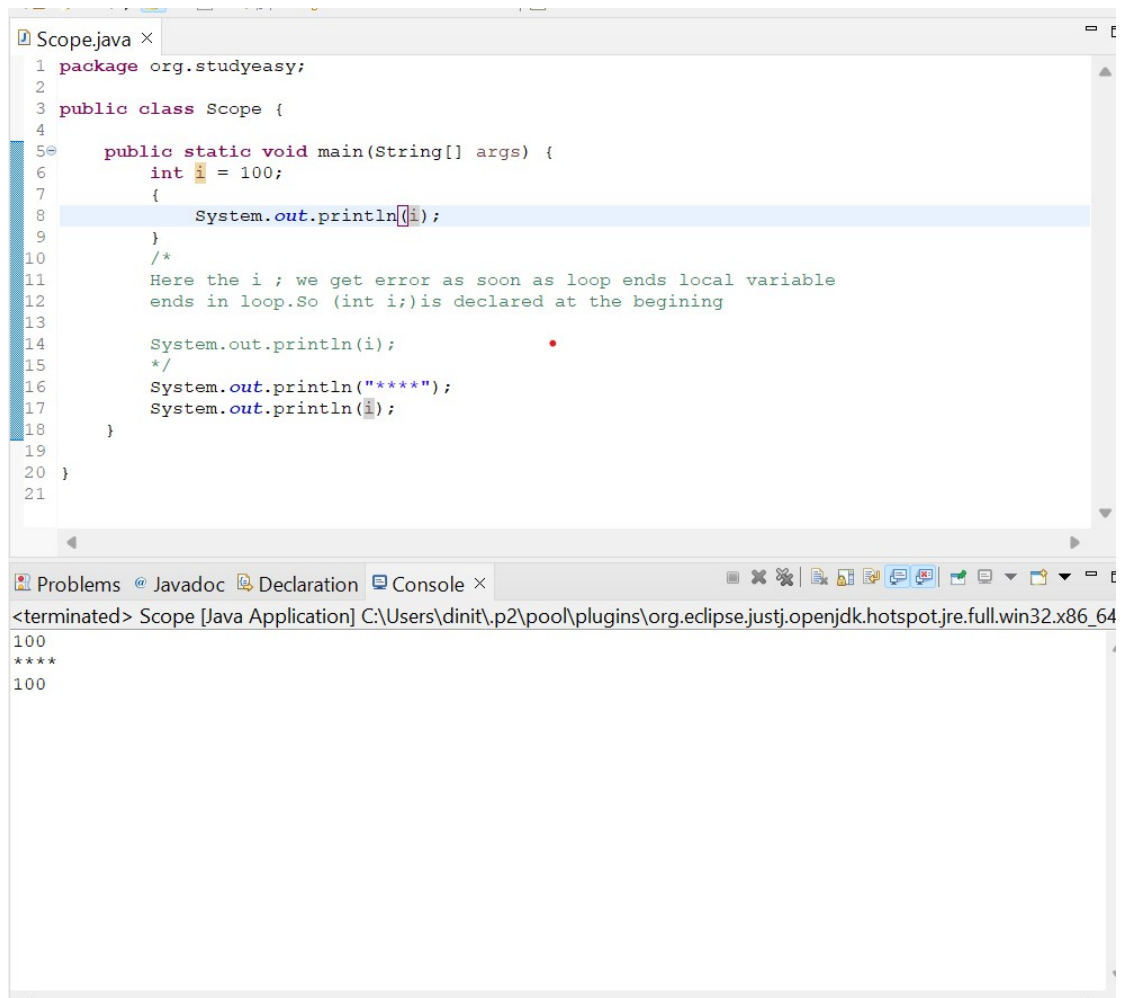
Problems @ Javadoc Declaration Console x

<terminated> Scope [Java Application] C:\Users\dinit\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

```
0
1
2
3
4
5
6
7
8
9
****
10
```

Writable Smart Insert 15 · 9 · 328

Now the outer variable will be accessed by the inner scopes, but the inner scope variable can't be accessed from the outside world.

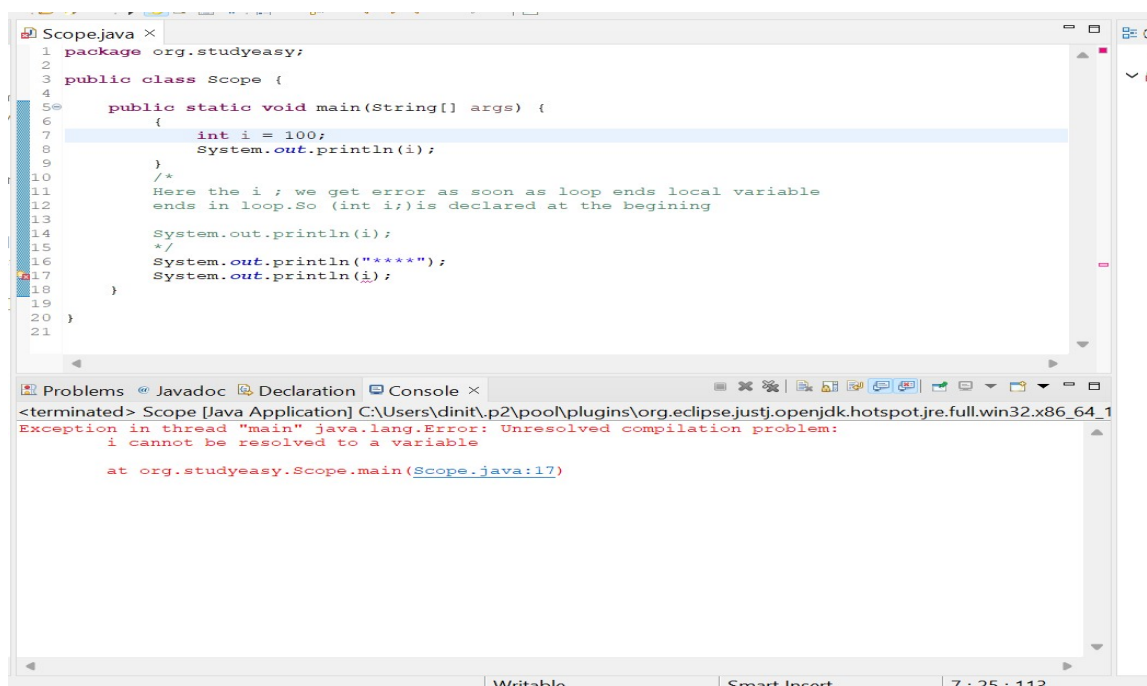


The screenshot shows the Eclipse IDE with a file named `Scope.java` open. The code defines a package `org.studyeasy` and a public class `Scope`. Inside the class, there is a `main` method that takes a `String[] args` array. Within the `main` method, a local variable `i` of type `int` is declared and assigned the value 100. A block of code is enclosed in curly braces, containing a `System.out.println(i);` statement. Below this block, there is a comment: `/* Here the i ; we get error as soon as loop ends local variable ends in loop. So (int i;) is declared at the beginning */`. This is followed by two more `System.out.println` statements: `System.out.println(i);` and `System.out.println(i);`. The IDE's console window at the bottom shows the output of the program: `<terminated> Scope [Java Application] C:\Users\dinit\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_100 **** 100`.

```
1 package org.studyeasy;
2
3 public class Scope {
4
5     public static void main(String[] args) {
6         int i = 100;
7         {
8             System.out.println(i);
9         }
10        /*
11         Here the i ; we get error as soon as loop ends local variable
12         ends in loop. So (int i;) is declared at the beginning
13
14         System.out.println(i);
15        */
16        System.out.println("****");
17        System.out.println(i);
18    }
19 }
20
21
```

<terminated> Scope [Java Application] C:\Users\dinit\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_100

100

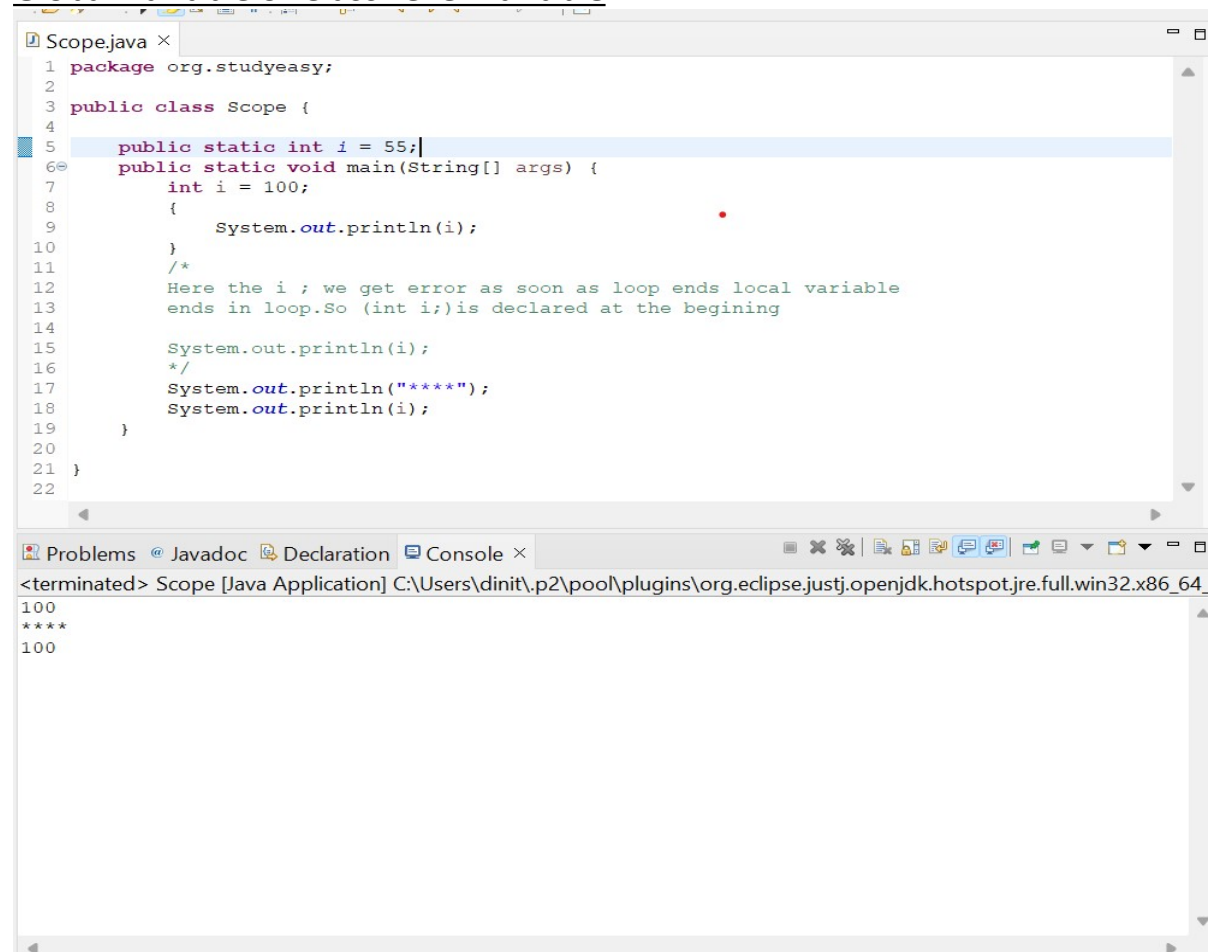


The screenshot shows the Eclipse IDE with the same `Scope.java` file. The code is identical to the previous screenshot. However, the IDE's console window now shows a compilation error. The error message is: `Exception in thread "main" java.lang.Error: Unresolved compilation problem: i cannot be resolved to a variable at org.studyeasy.Scope.main(Scope.java:17)`. The error points to line 17 of the file, which is the second `System.out.println(i);` statement.

```
1 package org.studyeasy;
2
3 public class Scope {
4
5     public static void main(String[] args) {
6         {
7             int i = 100;
8             System.out.println(i);
9         }
10        /*
11        Here the i ; we get error as soon as loop ends local variable
12        ends in loop. So (int i;) is declared at the beginning
13
14        System.out.println(i);
15       */
16       System.out.println("****");
17       System.out.println(i);
18   }
19 }
20
21
```

<terminated> Scope [Java Application] C:\Users\dinit\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
i cannot be resolved to a variable
at org.studyeasy.Scope.main(Scope.java:17)

Global Variable or Class Level Variable



```
Scope.java x
1 package org.studyeasy;
2
3 public class Scope {
4
5     public static int i = 55;
6     public static void main(String[] args) {
7         int i = 100;
8         {
9             System.out.println(i);
10        }
11        /*
12        Here the i ; we get error as soon as loop ends local variable
13        ends in loop.So (int i;)is declared at the begining
14
15        System.out.println(i);
16        */
17        System.out.println("****");
18        System.out.println(i);
19    }
20 }
21
22
```

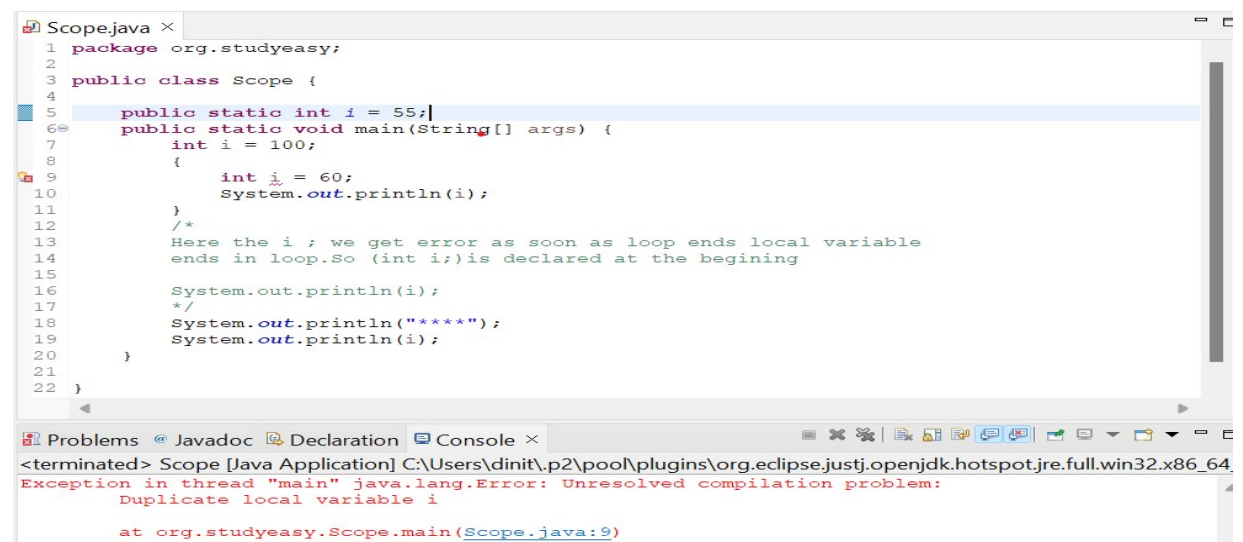
Problems @ Javadoc Declaration Console x

<terminated> Scope [Java Application] C:\Users\dinit\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

100

100

We have a method inside a method, we cannot repeat the name of the variable, which is a generic rule. Okay, but a class level variable or a global variable can have a same name. A global variable and a local variable. A method level variable and a class level variable can have same name. But in a method we cannot have duplicate variable names.



```
Scope.java x
1 package org.studyeasy;
2
3 public class Scope {
4
5     public static int i = 55;
6     public static void main(String[] args) {
7         int i = 100;
8         {
9             int i = 60;
10            System.out.println(i);
11        }
12        /*
13        Here the i ; we get error as soon as loop ends local variable
14        ends in loop.So (int i;)is declared at the begining
15
16        System.out.println(i);
17        */
18        System.out.println("****");
19        System.out.println(i);
20    }
21 }
22
```

Problems @ Javadoc Declaration Console x

<terminated> Scope [Java Application] C:\Users\dinit\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64

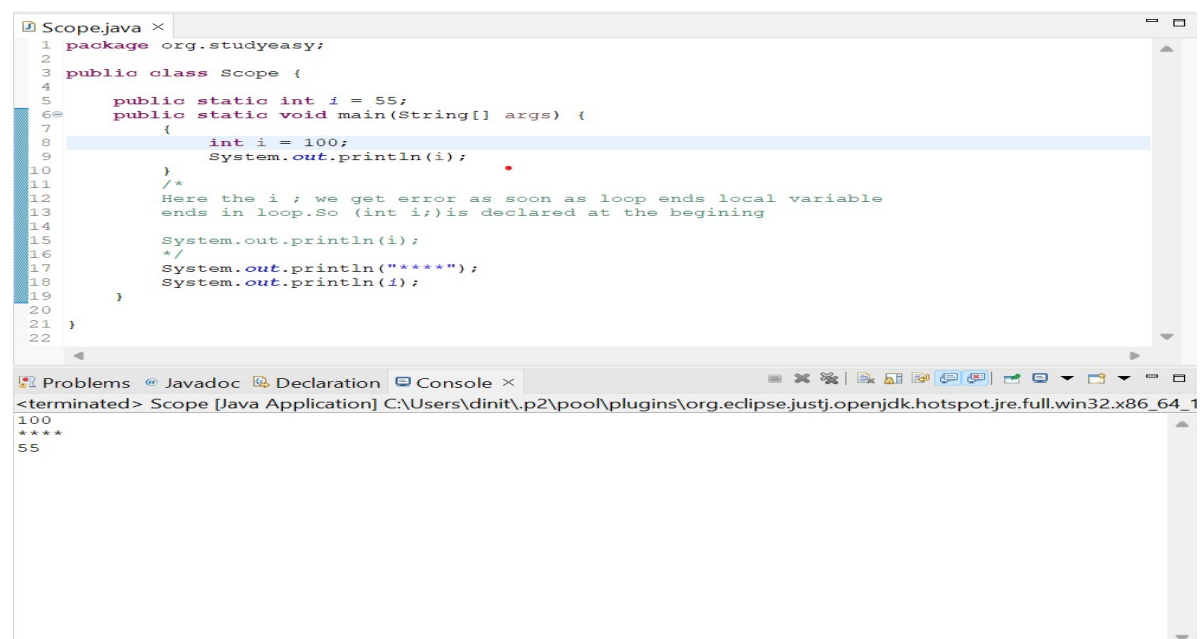
Exception in thread "main" java.lang.Error: Unresolved compilation problem:
Duplicate local variable i
at org.studyeasy.Scope.main(Scope.java:9)

If we have the local variable, the method will actually prefer the local variable.

But if there is a global variable and there is no local variable with that name, then the global or the method level.

Pardon me.

Class level variable will be used.



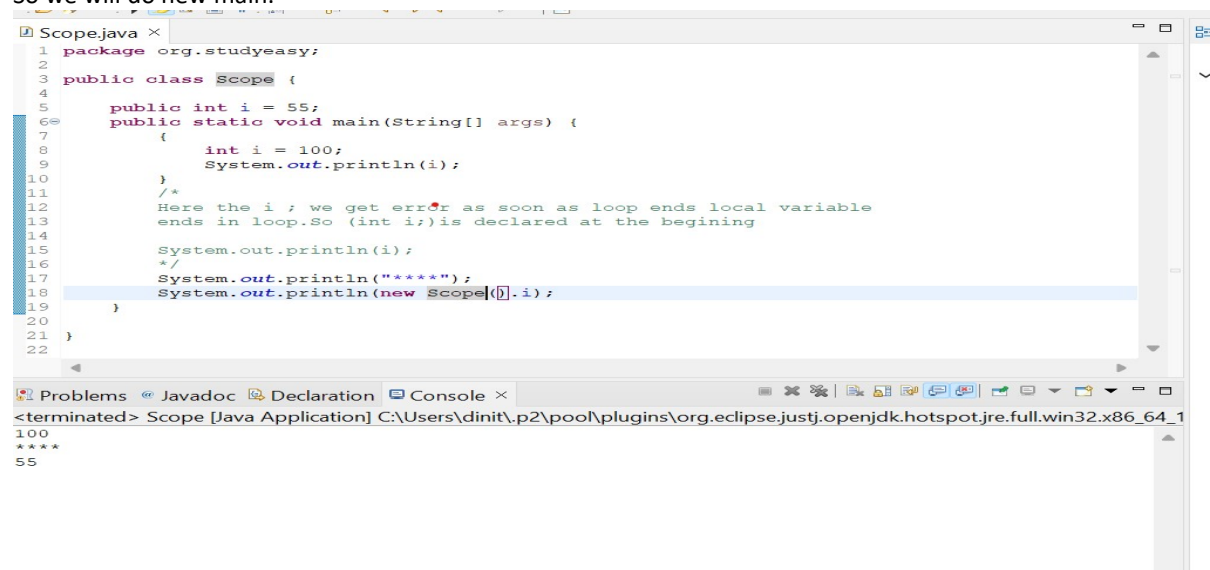
The screenshot shows the Eclipse IDE with a file named `Scope.java` open. The code defines a package `org.studyeasy` and a public class `Scope`. Inside the class, there is a public static integer `i` initialized to 55. A public static void method `main` takes a `String[] args` array. Inside the `main` method, a local integer `i` is declared and initialized to 100. The local `i` is printed, followed by a comment explaining that a global `i` would cause an error if declared at the beginning of the loop. Then, the class-level `i` is printed, followed by a line of four asterisks, and finally the class-level `i` is printed again. The console output at the bottom shows the execution results: 100, four asterisks, and 55.

```
1 package org.studyeasy;
2
3 public class Scope {
4
5     public static int i = 55;
6     public static void main(String[] args) {
7         {
8             int i = 100;
9             System.out.println(i);
10        }
11        /*
12         Here the i ; we get error as soon as loop ends local variable
13         ends in loop. So (int i;) is declared at the beginning
14
15        System.out.println(i);
16        */
17        System.out.println("****");
18        System.out.println(i);
19    }
20 }
21
22
```

<terminated> Scope [Java Application] C:\Users\dinit.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1
100

55

To access a variable, a class level variable where we don't have the static keyword or a variable is non static. Class level variable. In this case, as the name suggests it's a non static class level variable. So we will do new main.



The screenshot shows the Eclipse IDE with the same `Scope.java` file. The code is similar to the previous one, but the `main` method is now public and static. Inside the `main` method, the local `i` is printed, followed by a comment, then the class-level `i` is printed, followed by a line of four asterisks, and finally a new instance of the `Scope` class is created and its `i` attribute is printed. The console output at the bottom shows the execution results: 100, four asterisks, and 55.

```
1 package org.studyeasy;
2
3 public class Scope {
4
5     public int i = 55;
6     public static void main(String[] args) {
7         {
8             int i = 100;
9             System.out.println(i);
10        }
11        /*
12         Here the i ; we get error as soon as loop ends local variable
13         ends in loop. So (int i;) is declared at the beginning
14
15        System.out.println(i);
16        */
17        System.out.println("****");
18        System.out.println(new Scope().i);
19    }
20 }
21
22
```

<terminated> Scope [Java Application] C:\Users\dinit.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1
100

55

The screenshot shows the Eclipse IDE with a Java file named `Scope.java` open. The code defines a package `org.studyeasy` and a public class `Scope`. Inside the class, there is a private static variable `i` initialized to 55. A `main` method is defined, which contains a block of code where `i` is re-declared as a local variable and set to 100. The code prints the value of `i` at three different points: before the re-declaration, inside the block, and after the block. A comment explains that after the block ends, the local variable `i` is no longer in scope, so the program uses the class-level `i` (55). The console output shows the values 100, *****, and 55.

```
1 package org.studyeasy;
2
3 public class Scope {
4
5     private int i = 55;
6     public static void main(String[] args) {
7         {
8             int i = 100;
9             System.out.println(i);
10        }
11        /*
12         Here the i ; we get error as soon as loop ends local variable
13         ends in loop. So (int i;) is declared at the beginning
14
15         System.out.println(i);
16         */
17        System.out.println("****");
18        System.out.println(new Scope().i);
19    }
20 }
21
22
```

Problems @ Javadoc Declaration Console ×

<terminated> Scope [Java Application] C:\Users\dinit\p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_1

100

55

- `int i = 100;`
- `public static int i = 100;`
- `private int i = 100;` (Within Class itself)