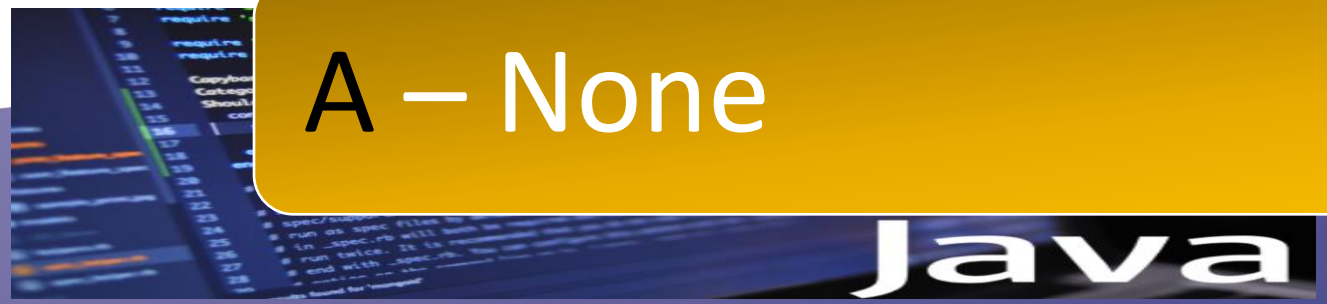
CJ Information

S – Variables scope

IC – explanation + examples

HW – 2024.11.11 Mon – Java classes

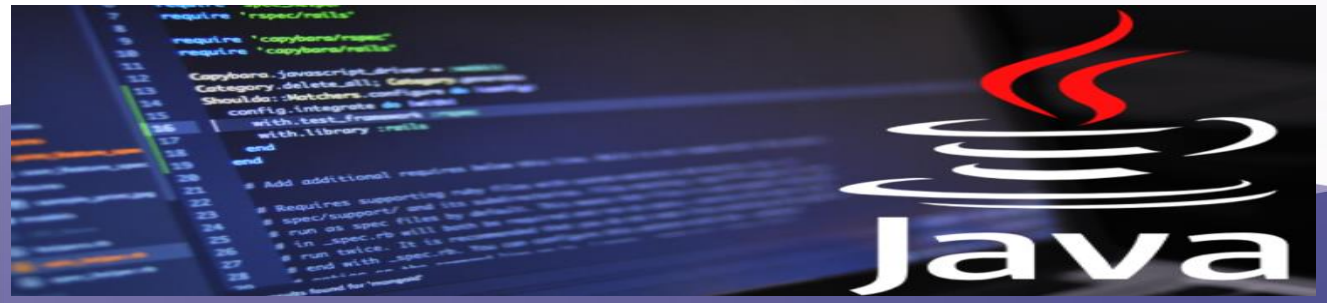
A – None



Variables Scope & Access

Outline :

- Meaning of Scope and Access
- How to use it



What we learn today:

- Understand the visibility of a variable
- Mistake : to access a variable out of its scope
- Examples

Scope & Access

Rules for variables visibility in java

Java scope defines where a certain variable or method is accessible in a program. Briefly:

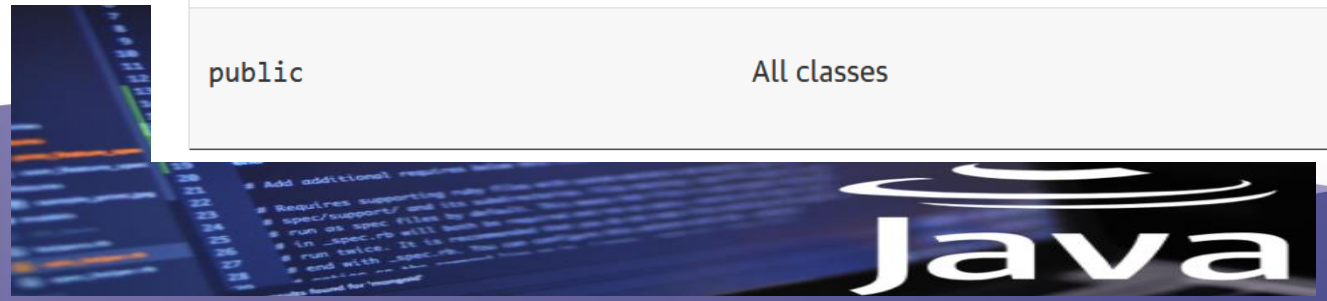
- A variable declared in a method is visible from the beginning of the declaration to the end of the method (method scope).
- A variable declared in a code block exists until the end of that code block.
- Variables that are method arguments exist till the end of the method.
- Class/object variables exist for the lifetime of the containing object. Their visibility is regulated by special access modifiers.
- Static class variables exist all the time the program is running. Their visibility is also determined by access modifiers.

Scope & Access

Rules for variables visibility in java

Table 6-1. Visibility modifiers

Modifier	Visibility outside the class
`private`	None
No modifier (default)	Classes in the package
`protected`	Classes in package and subclasses inside or outside the package
`public`	All classes



Scope & Access

Rules for variables visibility in java

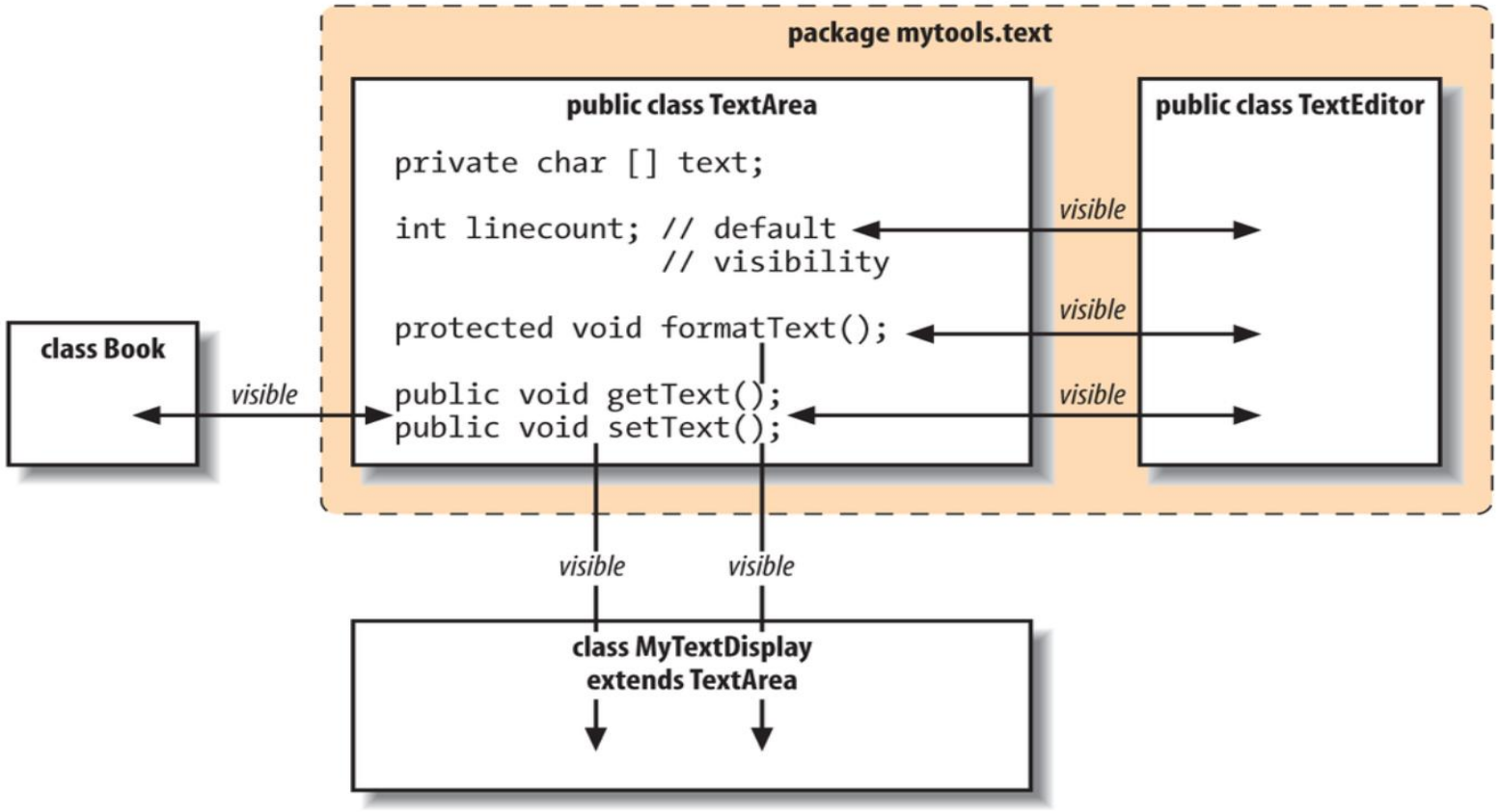


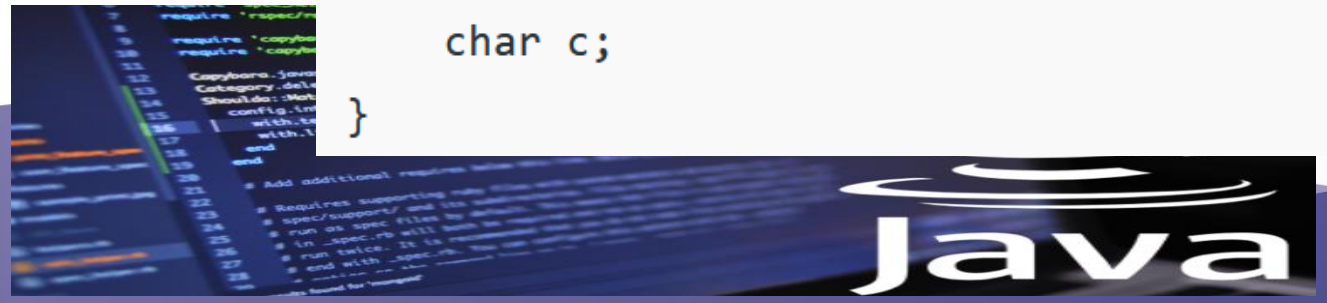
Figure 6-7. Private, default, protected, and public visibility

Scope & Access

Member Variables (Class Level Scope)

These variables must be declared inside class (outside any function). They can be directly accessed anywhere in class. Let's take a look at an example:

```
public class Test
{
    // All variables defined directly inside a class
    // are member variables
    int a;
    private String b;
    void method1() {....}
    int method2() {....}
    char c;
}
```

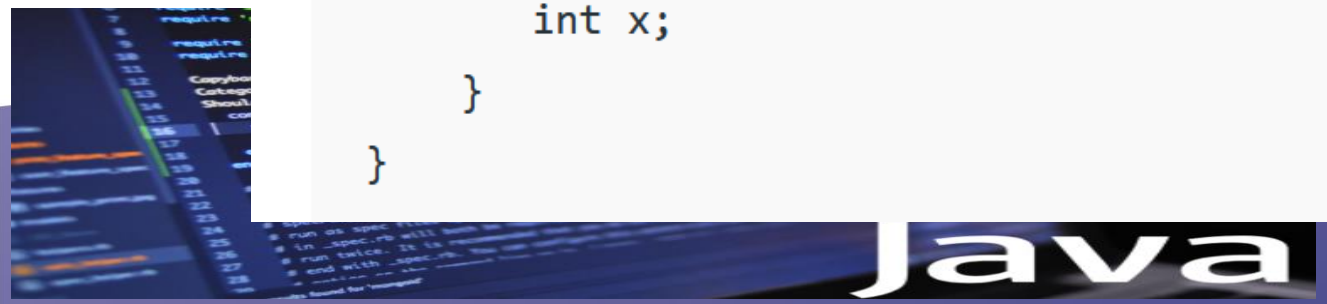


Scope & Access

Local Variables (Method Level Scope)

Variables declared inside a method have method level scope and can't be accessed outside the method.

```
public class Test
{
    void method1()
    {
        // Local variable (Method level scope)
        int x;
    }
}
```



Scope & Access

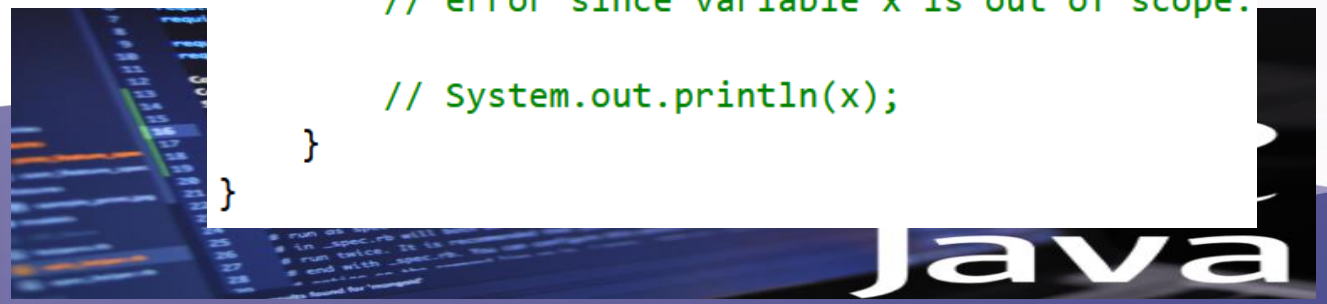
Loop Variables (Block Scope)

A variable declared inside pair of brackets “{” and “}” in a method has scope within the brackets only.

```
public class Test
{
    public static void main(String args[])
    {
        {
            // The variable x has scope within
            // brackets
            int x = 10;
            System.out.println(x);
        }

        // Uncommenting below line would produce
        // error since variable x is out of scope.

        // System.out.println(x);
    }
}
```



Output:

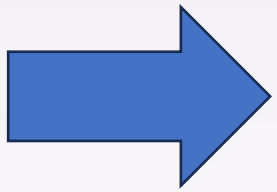
```
10
```

Scope & Access

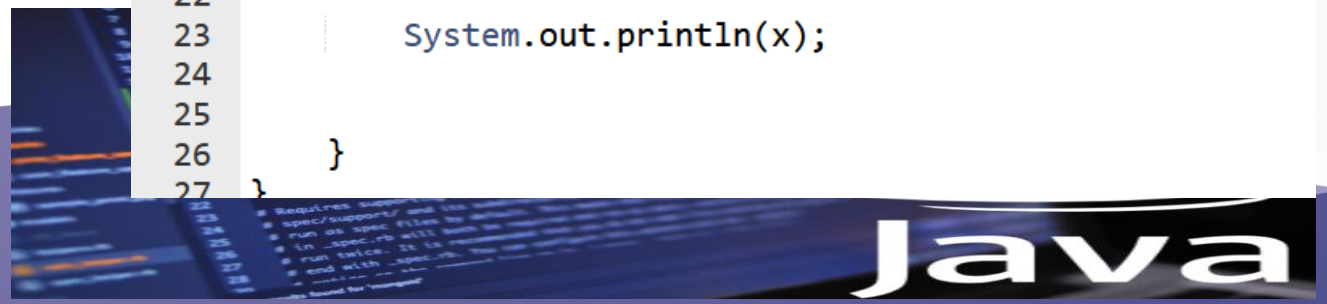
Loop Variables (Block Scope)

A variable declared inside pair of brackets “{” and “}” in a method has scope within the brackets only.

```
 9  public class Main
10  {
11      public static void main(String[] args) {
12
13          {
14              // The variable x has scope within
15              // brackets
16              int x = 10;
17              //System.out.println(x);
18          }
19
20          // Uncommenting below line would produce
21          // error since variable x is out of scope.
22
23          System.out.println(x);
24
25
26      }
27  }
```



```
Main.java:23: error: cannot find symbol
                System.out.println(x);
                                   ^
  symbol:   variable x
  location: class Main
1 error
```



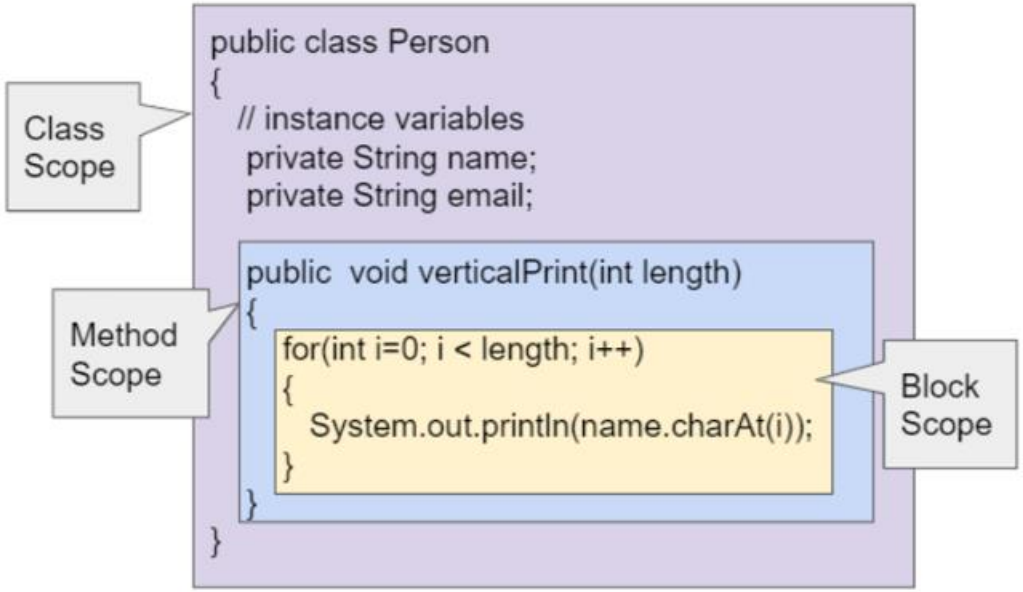
Scope & Access

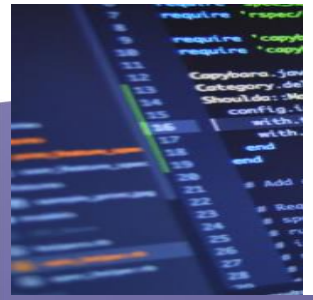
The following picture summarizes the different types of variables:

Java has 3 levels of scope that correspond to different types of variables:

- **Class Level Scope** for **instance variables** inside a class.
- **Method Level Scope** for **local variables** (including **parameter variables**) inside a method.
- **Block Level Scope** for **loop variables** and other local variables defined inside of blocks of code with { }.

The image below shows these 3 levels of scope.





Scope & Access

Exercises :

```
class Scope
{
    float pi = 3.14;


    public static void scopeSample (double a)
    {
        double b = 0;
        ...
        for (int c = 1; ...)
        {
            b = a + pi;
            ...
        }
    }

    public static void main (String[] args)
    {
        double x;
        ...
        scopeSample(x);

    }

}
```

Identify the scope of every single variable in this Java code

Scope & Access

Exercises :

```
class Scope
{
    float pi = 3.14;                          // scope of pi begins
                                              // GLOBAL VARIABLE

    public static void scopeSample (double a) // scope of a begins
    {
        double b = 0;                         // scope of b begins
        ...
        for (int c = 1; ...)                  // scope of c begins
        {
            b = a + pi;
            ...
        }                                     // scope of c ends
    }                                         // scope of a,b ends

    public static void main (String[] args)
    {
        double x;                             // scope of x begins
        ...
        scopeSample(x);                       // x is passed as
                                              // a parameter
    }
                                              // scope of pi ends
}                                             // spans entire class
```