

Variables and Data Types in Java

What is Variables?

- A variable provides us with named storage that our programs can manipulate.
- Each variable in Java has a specific type, which determines the size and layout of the variable's memory.
- Every variable has a name, called the variable name, and a data type.

$x = 9$

Why we use Variable?

- Variables play an important role in computer programming because they enable programmers to write flexible programs.
- Rather than entering data directly into a program, a programmer can use variables to represent the data.
- Variables store data temporarily in memory.

Types of Variable

```
graph TD; A[Types of Variable] --> B[Local]; A --> C[Instance]; A --> D[Static];
```

Local

Instance

Static

Local Variable

- A variable which is declared inside the method is called local variable.

Instance Variable

- A variable which is declared inside the class but outside the method, is called instance variable . It is not declared as static.

Static Variable

- A variable that is declared as static is called static variable. It cannot be local.

Print.java

var.java ✕

```
1 package try2;
2
3 public class var {
4     int x=9; //instance variable
5     static int y=24; //static variable
6     public static void main(String[] args) {
7         // TODO Auto-generated method stub
8         int z=2; //local variable
9         System.out.println(z);
10    }
11
12 }
13
```

A stylized landscape illustration featuring rolling green hills in the foreground, a small tree with a brown trunk and purple and pink foliage on the left, and blue and white wavy bands representing hills or clouds in the background.

Data Types in Java

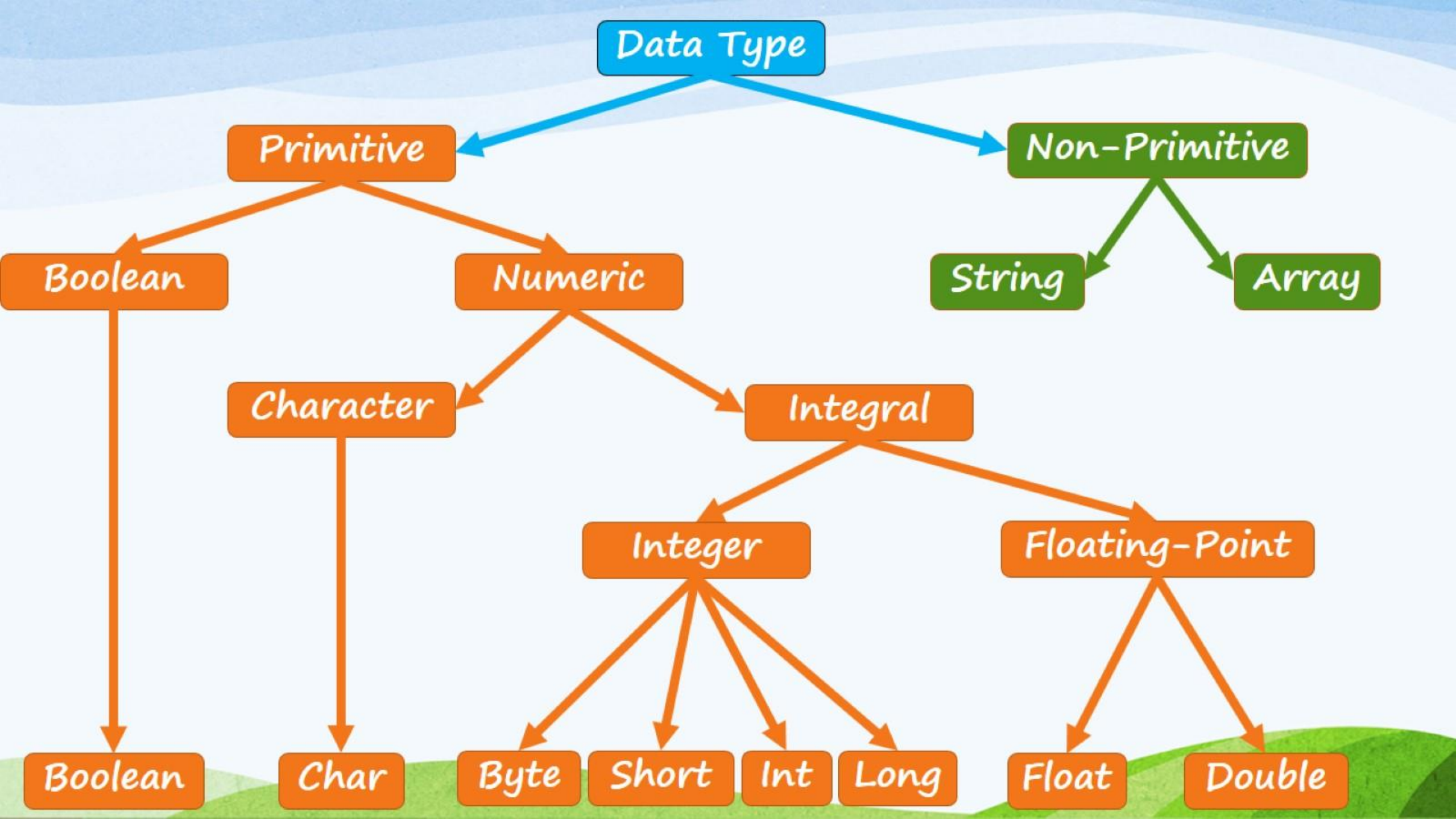
Data Types

```
graph TD; A[Data Types] --> B[Primitive]; A --> C[Non-primitive];
```

Primitive

Non-primitive

Data types represent the different values to be stored in the variable.



int

- Int data type is a 32-bit signed two's complement integer.
- Minimum value is - 2,147,483,648
- Maximum value is 2,147,483,647
- Integer is generally used as the default data type for integral values.
- The default value is 0
- Example: `int a = 9, int b = -24`

byte

- Byte data type is an 8-bit signed two's complement integer
- Minimum value is -128
- Maximum value is 127
- Default value is 0
- Byte data type is used to save space in large arrays, mainly in place of integers, since a byte is four times smaller than an integer.
- Example: byte a = 99, byte b = -24

short

- Short data type is a 16-bit signed two's complement integer
- Minimum value is -32,768
- Maximum value is 32,767
- Short data type can also be used to save memory as byte data type. A short is 2 times smaller than an integer
- Default value is 0.
- Example: short s = 99, short r = -24

long

- Long data type is a 64-bit signed two's complement integer
- Minimum value is -9,223,372,036,854,775,808
- Maximum value is 9,223,372,036,854,775,807. This type is used when a wider range than int is needed
- Default value is 0L
- Example: long a = 9000000L, long b = -22000000L

float

- Float data type is a single-precision 32-bit IEEE 754 floating point
- Float is mainly used to save memory in large arrays of floating point numbers
- Default value is 0.0f
- Example: float f1 = 914.2f

double

- double data type is a double-precision 64-bit IEEE 754 floating point
- This data type is generally used as the default data type for decimal values, generally the default choice
- Default value is 0.0d
- Example: double d1 = 924.2

boolean

- *boolean data type represents one bit of information*
- *There are only two possible values: true and false*
- *This data type is used for simple flags that track true/false conditions*
- *Default value is false*
- *Example: boolean a = true*

char

- *char data type is a single 16-bit Unicode character*
- *Minimum value is '\u0000' (or 0)*
- *Maximum value is '\uffff' (or 65,535 inclusive)*
- *Char data type is used to store any character*
- *Example: char name = 'Yash'*

A stylized landscape illustration featuring rolling green hills in the foreground, a small tree with a brown trunk and purple and pink foliage on the left, and blue and white wavy bands in the background representing a sky or distant hills. The text 'Type Casting' is written in a brown, cursive font in the center-right area.

Type Casting

Type Casting

- Assigning a value of one type to a variable of another type is known as Type Casting.

Type casting is classified into two types:

- Widening Casting

- byte → short → int → long → float → double

- Narrowing Casting

- double → float → long → int → short → byte

Widening

```
int a=9;  
float f=a;
```

Narrowing

```
float f=10.5f;  
int a=(int)f;
```

A stylized landscape illustration featuring rolling green hills in the foreground, a small tree with a brown trunk and purple and pink foliage on the left, and layered blue and white hills in the background. The word "Modifiers" is written in a brown, cursive font in the center-right area.

Modifiers

Modifier

- *Modifiers are keywords that you add to those definitions to change their meanings.*


```
graph TD; A[Modifiers] --> B[Access Modifiers]; A --> C[Non-Access Modifiers]
```

Modifiers

*Access
Modifiers*

*Non-Access
Modifiers*

Access Modifiers

- The access modifiers in java specifies accessibility (scope) of a data member, method, constructor or class.
- There are 4 types of java access modifiers:
 - `private`
 - `default`
 - `protected`
 - `public`
- There are many non-access modifiers such as `static`, `abstract`, `synchronized`, `native`, `volatile`, `transient` etc.

private access modifier

- The private access modifier is accessible only within class.
- Example:



The screenshot shows an IDE window with three tabs: Print.java, var.java, and *priv.java. The *priv.java tab is active, displaying the following code:

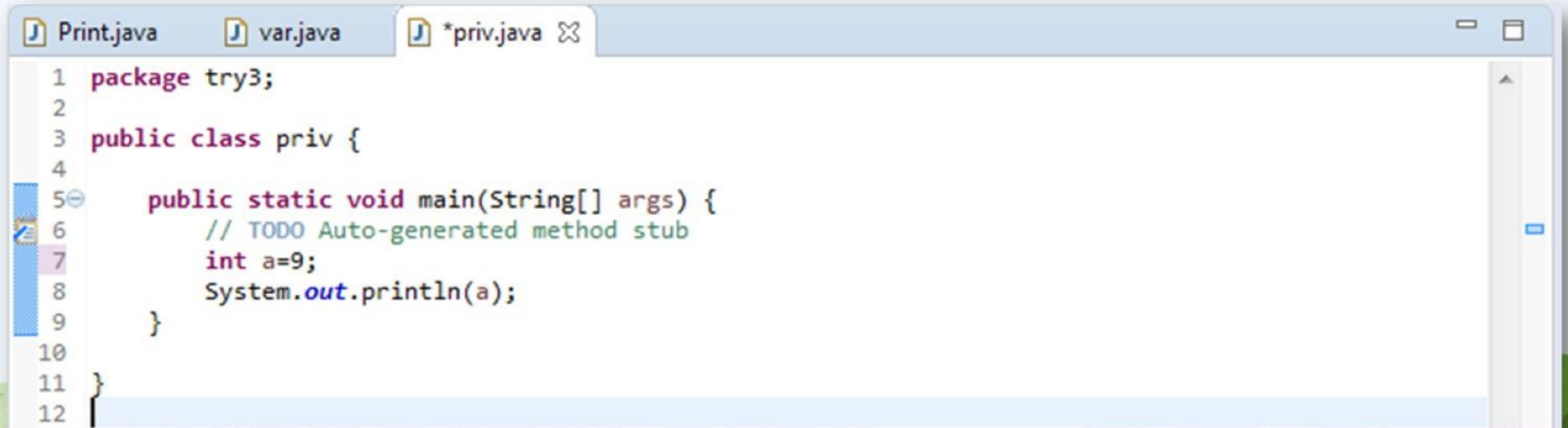
```
1 package try3;
2
3 public class priv {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         private int a=9;
8         System.out.println(a);
9     }
10
11 }
12 }
```

Line 7 contains the code `private int a=9;`. A red squiggly line is under the word `private`, indicating a compile-time error. The error message, partially visible on the right, states: "The modifier private is not allowed for the variable 'a' because it is already a public member of the class." This error occurs because the `private` access modifier cannot be used on a variable that is already declared as `public` within the same class.

Show compile time error

default access modifier

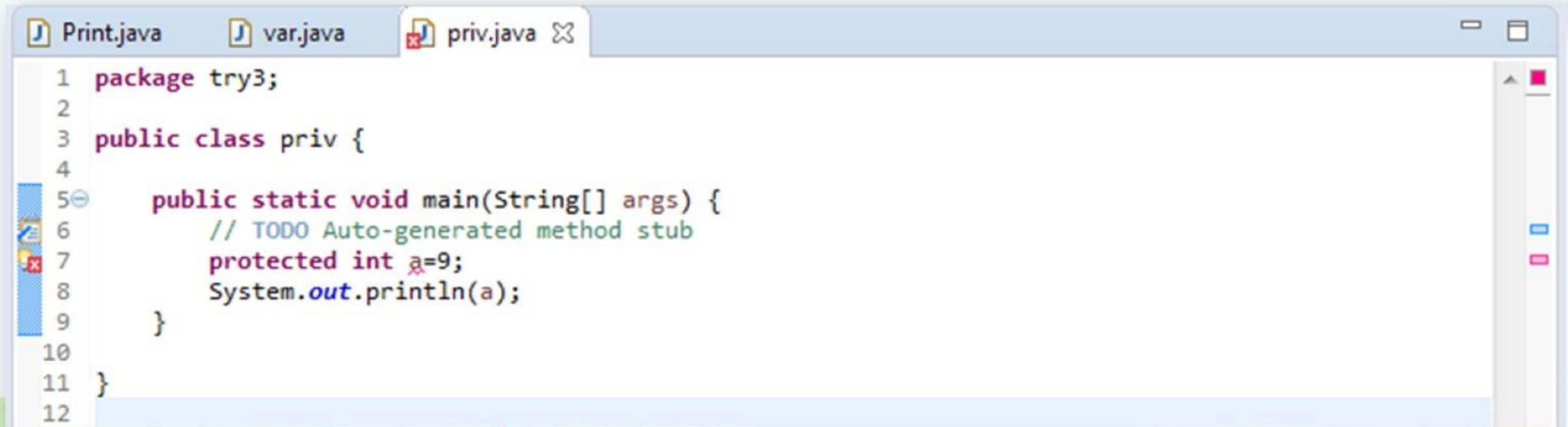
- If you don't use any modifier, it is treated as default by default. The default modifier is accessible only within package.
- Example:



```
Print.java  var.java  *priv.java X
1 package try3;
2
3 public class priv {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         int a=9;
8         System.out.println(a);
9     }
10
11 }
12 }
```

protected access modifier

- The protected access modifier is accessible within package and outside the package but through inheritance only.
- Example:



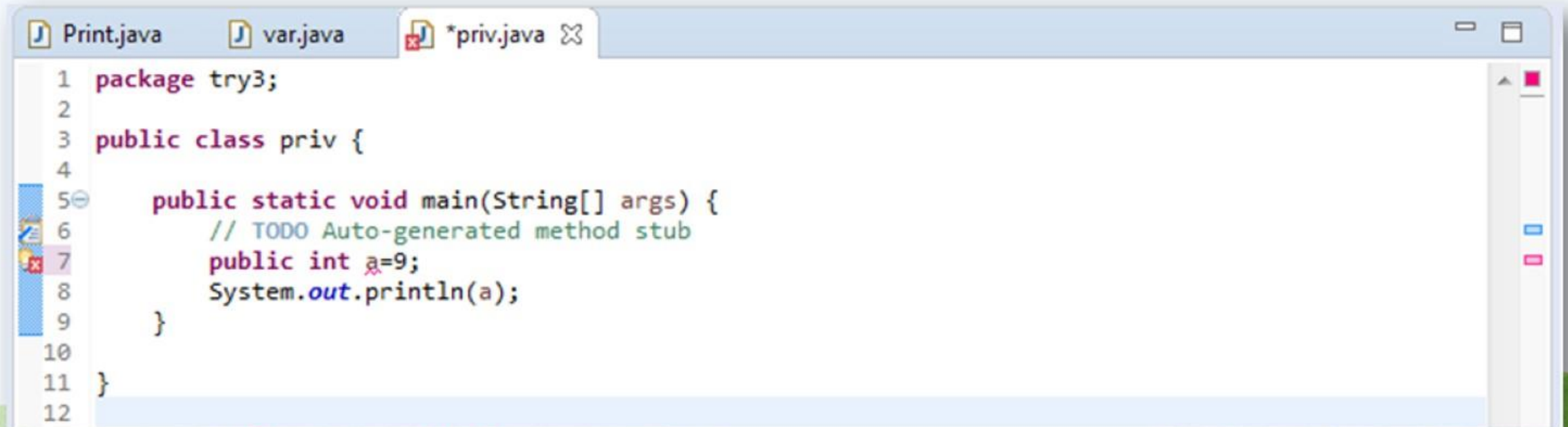
```
1 package try3;
2
3 public class priv {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         protected int a=9;
8         System.out.println(a);
9     }
10
11 }
12
```

The screenshot shows an IDE window with three tabs: Print.java, var.java, and priv.java. The priv.java tab is active, displaying the code above. A red squiggly line under the variable 'a' on line 7 indicates a compile-time error. The error is related to the 'protected' access modifier being used on a variable that is not a static field of the class.

Show compile time error

public access modifier

- The public access modifier is accessible everywhere. It has the widest scope among all other modifiers.
- Example:



The screenshot shows an IDE window with three tabs: Print.java, var.java, and *priv.java. The *priv.java tab is active, displaying the following code:

```
1 package try3;
2
3 public class priv {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7         public int a=9;
8         System.out.println(a);
9     }
10
11 }
12
```

Line 7 contains a compile-time error: "public int a=9;". The error is highlighted with a red squiggly line and a red 'x' icon in the left margin. The error message is "public int a=9;".

Show compile time error