Data Structures

Sec 1 : Hi Java!

By: Eng.Rahma Osama Eng.Sandra Sameh

Overview

What is Java?

Java is a popular programming language, created in 1995. It is owned by Oracle, and more than 3 billion devices run Java

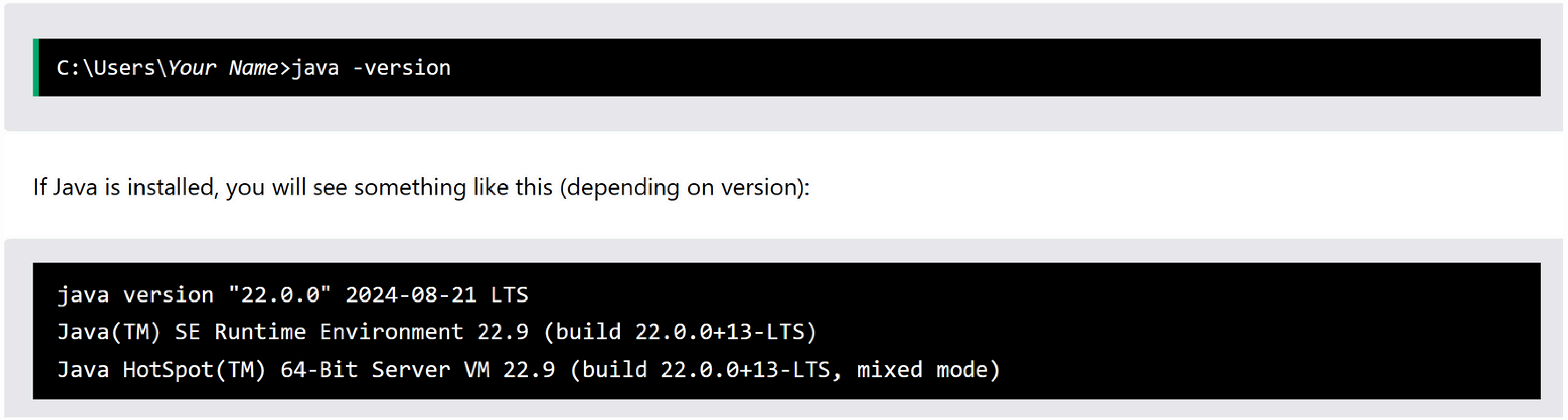
It is used for:

- Mobile applications (specially Android apps)
- Desktop applications
- Web applications
- Web servers and application servers
- Games
- Database connection

Environment Setup

Java Development Kit (Jdk):

Some PCs **might** have Java already installed.
To **check** if you have Java installed on a Windows PC, search in the start bar for **Java** or type the following in Command Prompt **(cmd.exe)**:



```
C:\Users\Your Name>java -version
```

If Java is installed, you will see something like this (depending on version):

```
java version "22.0.0" 2024-08-21 LTS
Java(TM) SE Runtime Environment 22.9 (build 22.0.0+13-LTS)
Java HotSpot(TM) 64-Bit Server VM 22.9 (build 22.0.0+13-LTS, mixed mode)
```

If you do not have Java installed on your computer, you can download it at oracle.com.

Environment Setup

- In **Java**, every application begins with a **class** name, and **that class must match the filename**.

Let's create our first Java file, called **Main.java**, which can be done in any text editor (like Notepad).
The file should contain a "Hello World" message, which is written with the following code:

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

Save the code in Notepad as "**Main.java**". Open Command Prompt **(cmd.exe)**, navigate to the directory where you saved your file, and type **"javac Main.java"**:

```
C:\Users\Your Name>javac Main.java
```

This will **compile** your code. If there are no errors in the code, the command prompt will take you to the next line. Now, type "**java Main**" to run the file:

```
C:\Users\Your Name>java Main
```

The **Output** :

```
Hello World
```

Environment Setup

Integrated Development environment (IDE):

There are several Java IDEs (Integrated Development Environments) you can use depending on your needs. Here are the most popular ones:

- IntelliJ IDEA (by JetBrains)
- Eclipse
- NetBeans (by Apache)
- VS Code
- JDeveloper (by Oracle)
- ...etc

You can also use any online IDE.

Get Started with Java !

Every line of code that runs in Java must be inside a **class**. The class **name** should always start with an **uppercase first letter**. In our example, we named the class **Main**.

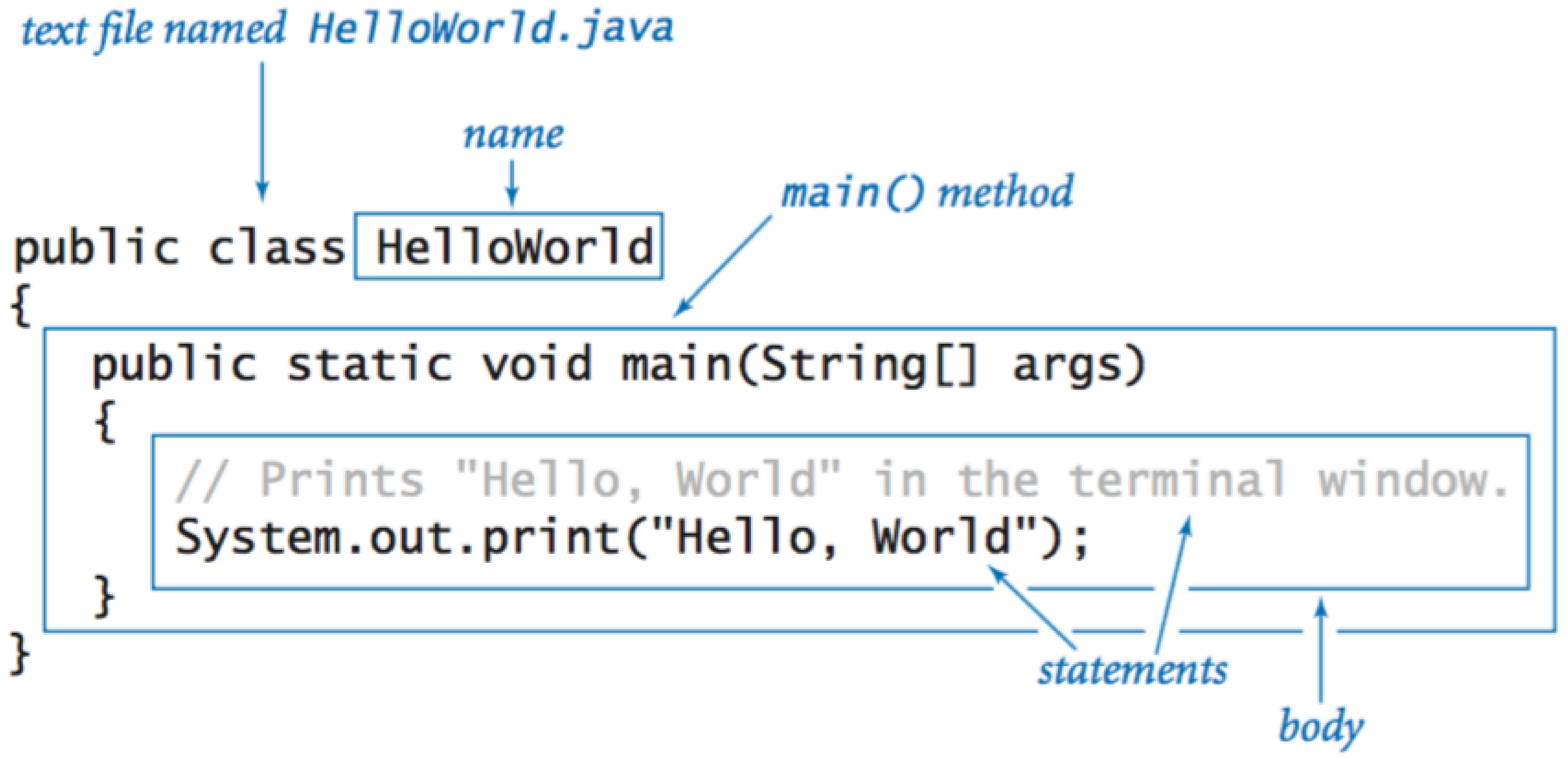
Note: Java is **case-sensitive**. MyClass and myclass would be treated as two completely different names.

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

- The **main()** method is required in every Java program. It is where the program **starts** running.
- Any code placed inside the **main()** method will be executed.

- **System.out.println() :**

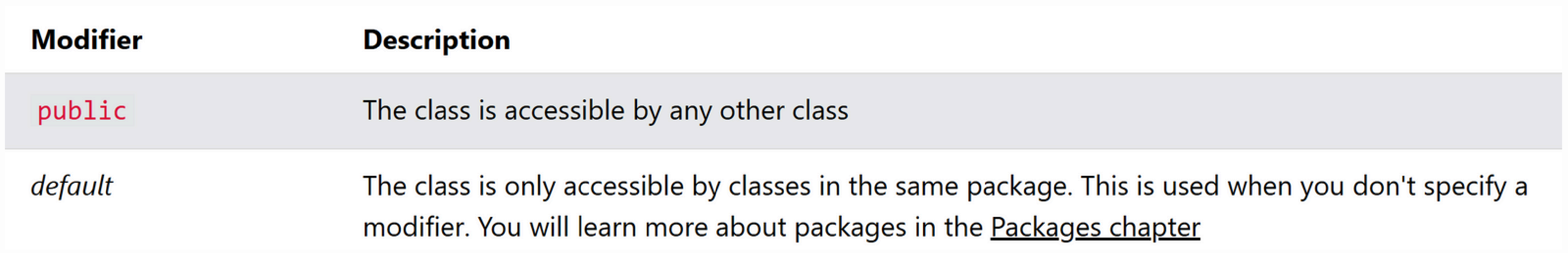
Inside the **main()** method, we can use the **println()** method to **print** a line of text to the screen



```
public class HelloWorld
{
   public static void main(String[] args)
   {
      // Prints "Hello, World" in the terminal window.
      System.out.print("Hello, World");
   }
}
```

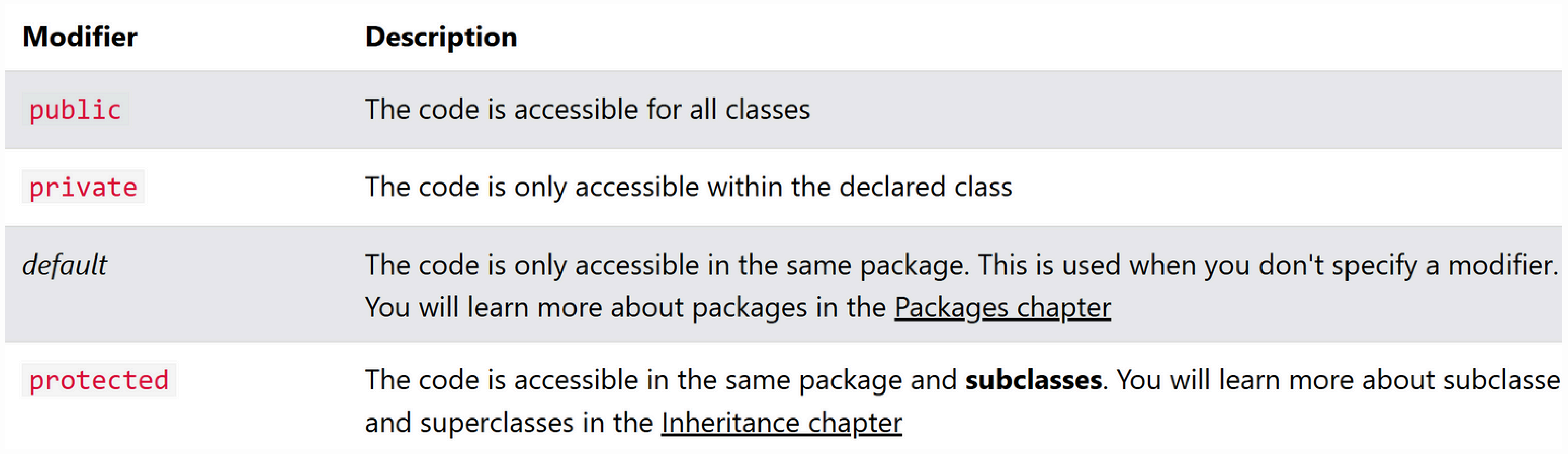
Access Modifiers in Java

- For **classes**, you can use either **public** or **default**:



Modifier	Description
`public`	The class is accessible by any other class
default	The class is only accessible by classes in the same package. This is used when you don't specify a modifier. You will learn more about packages in the Packages chapter

- For **attributes**, **methods** and **constructors**, you can use the one of the following:



Modifier	Description
`public`	The code is accessible for all classes
`private`	The code is only accessible within the declared class
default	The code is only accessible in the same package. This is used when you don't specify a modifier. You will learn more about packages in the Packages chapter
`protected`	The code is accessible in the same package and **subclasses**. You will learn more about subclasse and superclasses in the Inheritance chapter

Variables in Java

In **Java**, there are different **types** of variables, for **example:**

- **String** - stores **text**, such as "Hello". String values are surrounded by double quotes
- **int** - stores integers (whole **numbers**), without decimals, such as 123 or -123
- **float** - stores **floating point** numbers, with decimals, such as 19.99 or -19.99
- **char** - stores **single characters**, such as 'a' or 'B'. Char values are surrounded by single quotes
- **boolean** - stores values with two states: **true** or **false**

```
type variableName = value;
```

```
public class Main {
  public static void main(String[] args) {
    int myNum = 5;
    float myFloatNum = 5.99f;
    char myLetter = 'D';
    boolean myBool = true;
    String myText = "Hello";
    System.out.println(myNum);
    System.out.println(myFloatNum);
    System.out.println(myLetter);
    System.out.println(myBool);
    System.out.println(myText);
  }
}
```

Java Conditions and If Statements

```java
if (condition1) {
  // block of code to be executed if condition1 is true
} else if (condition2) {
  // block of code to be executed if the condition1 is
false and condition2 is true
} else {
  // block of code to be executed if the condition1 is
false and condition2 is false
}
```

```java
int weather = 2; // 1 = raining, 2 = sunny, 3 = cloudy

if (weather == 1) {
  System.out.println("Bring an umbrella.");
} else if (weather == 2) {
  System.out.println("Wear sunglasses.");
} else {
  System.out.println("Just go outside normally.");
}
// Outputs "Wear sunglasses."
```

Java Switch Statements

```
switch(expression) {
  case x:
    // code block
    break;
  case y:
    // code block
    break;
  default:
    // code block
}
```

- The **switch** expression is evaluated **once**.
- The result is **compared** with each case value.
- If there is a match, the **matching block of code runs**.
- The **break** statement stops the switch after the matching case has run.
- The **default** statement runs if there is **no match.**

```
int day = 4;
switch (day) {
  case 1:
    System.out.println("Monday");
    break;
  case 2:
    System.out.println("Tuesday");
    break;
  case 3:
    System.out.println("Wednesday");
    break;
  case 4:
    System.out.println("Thursday");
    break;
  case 5:
    System.out.println("Friday");
    break;
  case 6:
    System.out.println("Saturday");
    break;
  case 7:
    System.out.println("Sunday");
    break;
}
// Outputs "Thursday" (day 4)
```

Java While Loop

```
while (condition) {
  // code block to be executed
}
```

```
int i = 0;
while (i < 5) {
  System.out.println(i);
  i++;
}
```

```
int countdown = 3;

while (countdown > 0) {
  System.out.println(countdown);
  countdown--;
}

System.out.println("Happy New Year!!");
```

The Do/While Loop

```
do {
  // code block to be executed
}
while (condition);
```

- A **do/while** loop always runs **at least once**, <u>even if the condition is false</u> at the start. This is the **key difference** from a **while** loop, which would skip the code block completely in the same situation.

- This **behavior** makes do/while **useful** when you want something to happen **at least once**, such as showing a message or asking the user for input.

```
int i = 0;
do {
  System.out.println(i);
  i++;
}
while (i < 5);
```

Java For Loop

```
for (statement 1; statement 2; statement 3) {
  // code block to be executed
}
```

- **Statement 1** is executed **(one time)** before the execution of the code block.
- **Statement 2** defines the condition for executing the code block.
- **Statement 3** is executed (every time) after the code block has been executed.

```
for (int i = 0; i < 5; i++) {
  System.out.println(i);
}
```

```
int sum = 0;
for (int i = 1; i <= 5; i++) {
  sum = sum + i;
}
System.out.println("Sum is " + sum);
```

The for-each Loop

```
for (type variableName : arrayName) {
  // code block to be executed
}
```

The for-each loop is simpler and more readable than a regular for loop, since you don't need a counter (like i < array.length).

```
String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

for (String car : cars) {
  System.out.println(car);
}
```

THANK YOU!