CSC 125 Object Oriented Programming

Ch02_Elementary programming

Dr. Fadi Alzhouri

Formatting output: Escape sequence

- An escape sequence in programming is a series of characters that represents a special character or format.
- Escape sequences typically start with a backslash (\) followed by one or more characters.

TABLE 4.5 Escape Sequences

Escape Sequence	*Name*	*Unicode Code*	*Decimal Value*
`\b`	Backspace	`\u0008`	8
`\t`	Tab	`\u0009`	9
`\n`	Linefeed	`\u000A`	10
`\f`	Formfeed	`\u000C`	12
`\r`	Carriage Return	`\u000D`	13
`\\`	Backslash	`\u005C`	92
`\"`	Double Quote	`\u0022`	34

commonly-used escape sequences

Formatting output: Escape sequence (cont.)

- Example:

```
String str1 = "Hello\tworld\n";    // tab and newline
String str2 = "Double quoted \"hello\"";
String str3 = "A back-slash \\, another 2 back-slashes \\\\";
System.out.println(str1);
System.out.println(str2);
System.out.println(str3);
```

```
Hello	world

Double quoted "hello"
A back-slash \, another 2 back-slashes \\
```

End-of-Line (EOL)

- Newline (0AH) and Carriage Return (0DH) are used to represent the escape sequences `\n` and `\r`, respectively.
- They are used as line delimiters (or end-of-line) in text files.
- Unix and macOS (modern versions) use `\n` (0AH) as the EOL character.
- Windows uses a combination of `\r\n` (0D 0AH) as the EOL sequence.

Arithmetic Operators

- The operators for numeric data types include the standard arithmetic operators.

Operator	Mode	Usage	Description	Examples
+	Binary	x + y	Addition	1 + 2 = 3
	Unary	+x	Unary positive	1.1 + 2 = 3.1
-	Binary	x - y	Subtraction	1 - 2 = -1
	Unary	-x	Unary negate	1.1 - 2 = -0.9
*	Binary	x * y	Multiplication	2 * 3 = 6 2 * 3.0 = 6.0
/	Binary	x / y	Division	2/ 4 = 0 2 / 4.0 = 0.5
%	Binary	x % y	Modulus (Remainder)	2 % 3 = 2 2 % 3.0 = 2.0 7 % 3.0 = 1.0

Arithmetic Operators (cont.)

- Division (/): when both operands (numerator & denominator) of a division are integers, the result of the division is integer, and the fractional part is truncated.
- To get a double-point (real) result, one of the operands must be a double number.
  - Example: 3 / 2.0 = 1.0
- Modulus (%): the remainder is negative only if the dividend is negative.
- When both operators (dividend & divisor) are integers, the remainder is integer, while if one of them is double, the remainder is a double.

Arithmetic Expressions in Java

- Write a program to calculate the following expression: $\frac{3 + 4x}{5}$

Arithmetic Expressions in Java

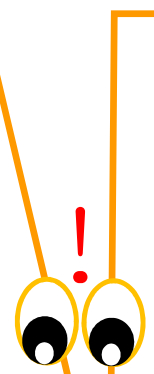
- Write a program to calculate the following expression:

```
/*********************************************************
Arithmetic operations
Author: Dr. Fadi Alzhouri
Example 8: Arithmetic expression

*********************************************************/
import java.util.*;
public class MathExpressions
{
    public static void main(String[] args) {

        Scanner number = new Scanner(System.in);
        int y;
        int x = number.nextInt();
        y = 3 + 4 * x /5;
        System.out.println("if x = " + x + ", y = " + y);
    }
}
```

```
4
if x = 4, y = 6
```

$$\frac{3 + 4x}{5}$$

Is it correct!!!!



Arithmetic Expressions in Java (cont.)

- $\frac{3 + 4x}{5} \neq$ `3 + 4 * x /5;`
- $\frac{3 + 4x}{5} =$ `(3 + 4 * x) /5;`

- Java evaluates arithmetic expressions based on operator precedence and associativity rules.

Arithmetic Expressions in Java (cont.)

```java
/*****************************************************
Arithmetic operations
Author: Dr. Fadi Alzhouri
Example 8: Arithmetic expression

*****************************************************/
import java.util.*;
public class MathExpressions
{
    public static void main(String[] args) {
        Scanner number = new Scanner(System.in);
        int y;
        int x = number.nextInt();
         y= (3 + 4 * x) /5;
        System.out.println("if x = " + x + ", y = " + y);
    }
}
```

```
4
if x = 4, y = 3
```

Operators Precedence

1. Parentheses () have the highest precedence and can be used to change the order of evaluation.
2. Unary '-' (negate) and '+' (positive) have next higher precedence.
3. The multiplication (*), division (/) and modulus (%) have the same precedence.
4. Addition (+) and subtraction (-)
5. Within the same precedence level (i.e., addition/subtraction and multiplication/division/modulus), the expression is evaluated from left to right (called left-associative).

Operators Precedence (cont.)

Example:

```
3 + 4 * 4 + 5 * (4 + 3) - 1
                        ↑
                        (1) inside parentheses first
3 + 4 * 4 + 5 * 7 - 1
        ↑
        (2) multiplication
3 + 16 + 5 * 7 - 1
             ↑
             (3) multiplication
3 + 16 + 35 - 1
  ↑
  (4) addition
19 + 35 - 1
   ↑
   (5) addition
54 - 1
   ↑
   (6) subtraction
53
```

Operators Precedence (cont.)

Exercise: evaluate each expression

```
System.out.println( 5 + 3 - 4 * 4 / 3);

System.out.println( 5 + (3 - 4) * 4 / 3);

System.out.println( (5 + 3) - 4 * 4 / 3);

System.out.println( (5 + 3 - 4) * 4 / 3);

System.out.println( ((5 + 3) - 4) * 4 / 3);

System.out.println( 5 + 3 - 4 * (4 / 3));
```

Operators Precedence (cont.)

Exercise: evaluate each expression

```
System.out.println( 5 + 3 - 4 * 4 / 3);
System.out.println( 5 + (3 - 4) * 4 / 3);
System.out.println( (5 + 3) - 4 * 4 / 3);
System.out.println( (5 + 3 - 4) * 4 / 3);
System.out.println( ((5 + 3) - 4) * 4 / 3);
System.out.println( 5 + 3 - 4 * (4 / 3));
```

```
3
4
3
5
5
4
```

Compound Assignment Operators

- The operators +, -, *, /, and % can be combined with the assignment operator to form augmented operators.
- The current value of a variable is used, modified, then reassigned back to the same variable.
- `count = count + 1;` → `count += 1;`

Compound addition

Compound Assignment Operators (cont.)

Operator	*Name*	*Example*	*Equivalent*
`+=`	Addition assignment	`i += 8`	`i = i + 8`
`-=`	Subtraction assignment	`i -= 8`	`i = i - 8`
`*=`	Multiplication assignment	`i *= 8`	`i = i * 8`
`/=`	Division assignment	`i /= 8`	`i = i / 8`
`%=`	Remainder assignment	`i %= 8`	`i = i % 8`

- Caution: there are no spaces in the compound assignment operators.
  - For example, + = should be +=.

Compound Assignment Operators (cont.)

```
int x=3;
x+=2;
System.out.println(x);
```

```
int x=3;
System.out.println(x/=2);
```

- Caution: there are no spaces in the compound assignment operators.
  - For example, + = should be +=.

Compound Assignment Operators (cont.)

- Exercise 1:

```
double x=8.5, y;

x%=x*2;

y = x + 1.5;

System.out.println(x + "   " + y);
```

Compound Assignment Operators (cont.)

- Exercise 1:

```
double x=8.5, y;

x%=x*2;

y = x + 1.5;

System.out.println(x + "    " + y);
```

```
8.5    10.0
```

Compound Assignment Operators (cont.)

- Exercise 2:

```
double x=8, y=1;

x *= 2 + y;

System.out.println("x is " + x);
```

Compound Assignment Operators (cont.)

- Exercise 2:

```
double x=8, y=1;

x *= 2 + y;

System.out.println("x is " + x);
```

x = x * (2 + y)

Increment and Decrement Operators

- The increment operator (+ +) increments a variable by 1.
- The decrement operator (- -) decrements a variable by 1.

TABLE 2.5 Increment and Decrement Operators

Operator	*Name*	*Description*	*Example* (assume i = 1)
`++var`	preincrement	Increment `var` by `1`, and use the new `var` value in the statement	`int j = ++i;` // j is 2, i is 2
`var++`	postincrement	Increment `var` by `1`, but use the original `var` value in the statement	`int j = i++;` // j is 1, i is 2
`--var`	predecrement	Decrement `var` by `1`, and use the new `var` value in the statement	`int j = --i;` // j is 0, i is 0
`var--`	postdecrement	Decrement `var` by `1`, and use the original `var` value in the statement	`int j = i--;` // j is 1, i is 0

old value

new value

- Work with all primitive data types except Boolean.

Increment and Decrement Operators (cont.)

- Exercise:

```
/*********************************************************
Declaring and Using Variables
Author: Dr. Fadi Alzhouri
Example 9: Increment and Decrement Operators
*********************************************************
public class Main
{
    public static void main(String[] args) {

        int  x=2, y=4;

        y = x++;

        y +=++x;

        y = y - --x;

    }
}
```

Relational (comparison) Operators (cont.)

Java Operator	*Mathematics Symbol*	*Name*	*Example (radius is 5)*	*Result*
`<`	<	Less than	`radius < 0`	`false`
`<=`	≤	Less than or equal to	`radius <= 0`	`false`
`>`	>	Greater than	`radius > 0`	`true`
`>=`	≥	Greater than or equal to	`radius >= 0`	`true`
`==`	=	Equal to	`radius == 0`	`false`
`!=`	≠	Not equal to	`radius != 0`	`true`

- The equality testing operator is two equal signs (==), not a single equal sign (=).
- The equal sign (=) symbol is for assignment.

Relational (comparison) Operators (cont.)

Exercise:

```
System.out.println(5 == 5);
System.out.println(5 != 4);
System.out.println(5 <= 5);
System.out.println(5 < 4);
System.out.println(5 <= 8);
System.out.println(5 =< 8);
```

Relational (comparison) Operators (cont.)

Exercise:

```
System.out.println(5 == 5);
System.out.println(5 != 4);
System.out.println(5 <= 5);
System.out.println(5 < 4);
System.out.println(5 <= 8);
System.out.println(5 =< 8);
```

```
true
true
true
false
true
```

error

Relational (comparison) Operators (cont.)

Exercise 2:

```
boolean isValid = true;
System.out.println(isValid != true);
```

Logical Operators

- The logical operators can be used to create a compound boolean expression.
- It operates on boolean operands only, in descending order of precedence

Operator	*Name*	*Description*
!	not	Logical negation
&&	and	Logical conjunction
||	or	Logical disjunction
^	exclusive or	Logical exclusion

Logical Operators (cont.)

- Truth tables for all logical operators

p	!p
true	false
false	true

Not

p_1	p_2	p_1 && p_2
false	false	false
false	true	false
true	false	false
true	true	true

And

p_1	p_2	p_1 || p_2
false	false	false
false	true	true
true	false	true
true	true	true

Or

p_1	p_2	p_1 ^ p_2
false	false	false
false	true	true
true	false	true
true	true	false

XOR

Logical Operators (cont.)

- Exercise:

```java
/*************************************************************
Declaring and Using Variables
Author: Dr. Fadi Alzhouri
Example 10: logical Operators
*************************************************************
public class Main
{
    public static void main(String[] args) {

        int x = 4;
        int y = 5;

        System.out.println((x > 3) && (y < x) );

        System.out.println((x > 3) || (y < x) );
    }
}
```

Logical Operators (cont.)

- Exercise:

```java
/*********************************************************
Declaring and Using Variables
Author: Dr. Fadi Alzhouri
Example 10: logical Operators
*********************************************************
public class Main
{
    public static void main(String[] args) {

        int x = 4;
        int y = 5;

        System.out.println((x > 3) && (y < x) );

        System.out.println((x > 3) || (y < x) );
    }
}
```

```
false
true
```

Some IT Terminologies (cont.)

- API: An API is a set of rules and protocols that allows different software applications to communicate with each other.
  - It defines the methods and data formats that applications can use to request and exchange information.
- Java's API:
  - Library APIs: Programming languages like Java provide APIs that offer built-in functionalities, such as data structures and file handling.

Java Editions

- Java Standard Edition (Java SE): to develop client-side applications, run on desktop.
- Java Enterprise Edition (Java EE): to develop server-side applications, such as Java servlets, JavaServer Pages (JSP), and JavaServer Faces (JSF).
- Java Micro Edition (Java ME): to develop applications for mobile devices

Characteristics of Java

- Java Is Object-Oriented
  - Object-oriented programming provides great flexibility, modularity, clarity, and reusability through encapsulation, inheritance, and polymorphism.
- Java Is Interpreted
  - You need an interpreter to run Java programs. The programs are compiled into the Java Virtual Machine code called bytecode. The bytecode is machine-independent and can run on any machine that has a Java interpreter, which is part of the Java Virtual Machine (JVM).
- Java Is Portable
  - They can be run on any platform without being recompiled.

Object-Oriented Programming (OOP) vs Procedural Programming (PP)

Aspect	Object-Oriented Programming (OOP)	Procedural Programming
Definition	A paradigm based on objects that combine data and behavior.	A paradigm based on procedures or routines.
Structure	Organized around classes and objects.	Organized around procedures and functions.
Data Handling	Encapsulates data and functions into objects.	Data is often shared globally or passed to functions.
Flexibility	More flexible due to polymorphism.	Less flexible; changes may require altering many functions.
Examples	Java, C++, Python	C, Pascal, Fortran.

Exercise

- To write your first program on your laptop or PC, you'll need:

1. The Java SE Development Kit
   - For Microsoft Windows, Solaris OS, and Linux: Java SE Downloads Index
   - For Mac OS X: developer.apple.com

2. The NetBeans IDE
   - For all platforms: NetBeans IDE Downloads Index

Just for training

3. If you are busy and want to have a quick cup of Java coffee, use the following online compiler: https://www.onlinegdb.com/online_java_compiler

First taste of Java

1. Open your IDE or text editor.
2. Create a new file:
   - File Name: HelloWorld.java (the name must match the public class name).

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

3. Save the File
   - Save the file with the .java extension (e.g., HelloWorld.java).

First taste of Java (cont.)

4. Compile the Code
   - Use the compiler javac as shown.

```
javac HelloWorld.java
```

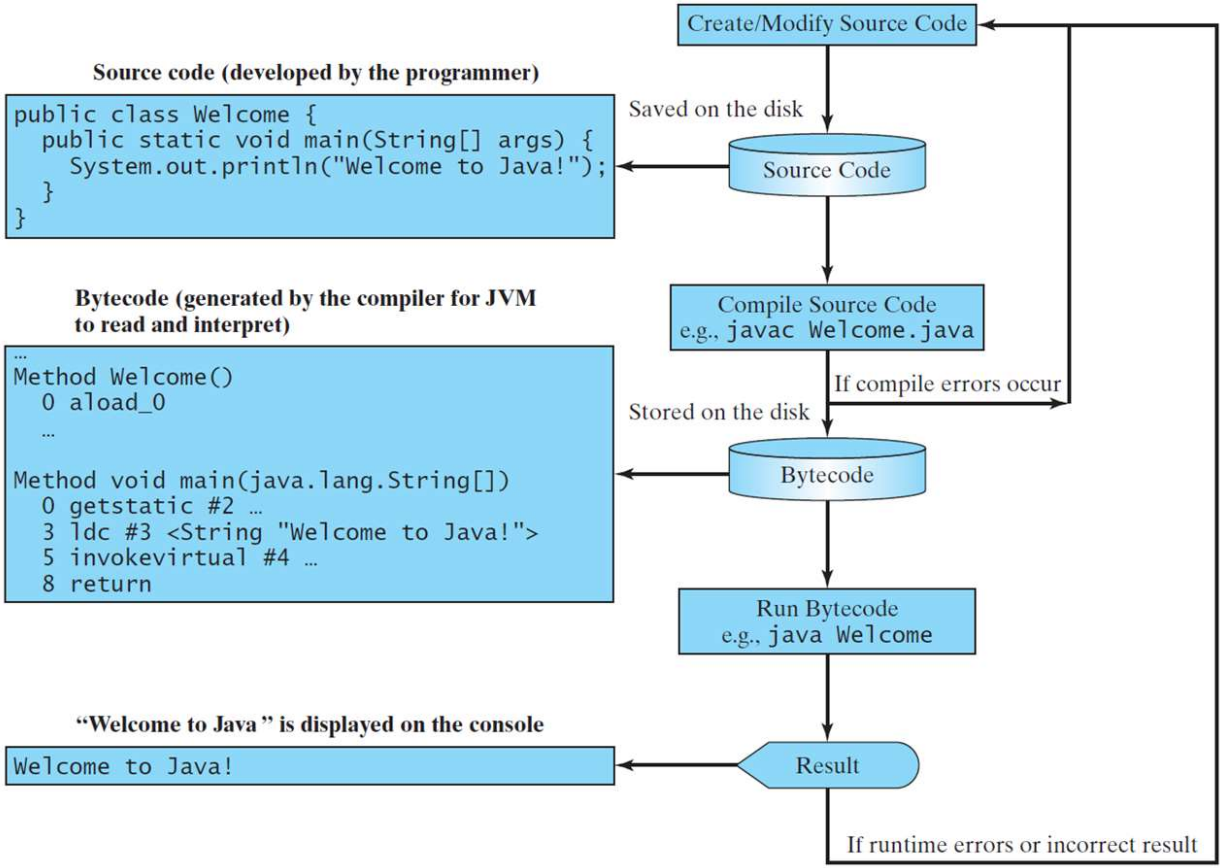
   - The compiler generates a HelloWorld.class file (bytecode).
5. Run the java program using Java command

```
java HelloWorld
```

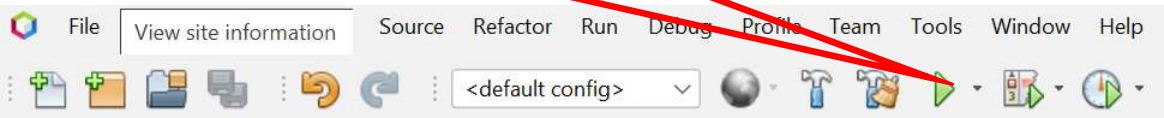
   - You should see the output

```
Hello, World!
```

Review 1



The beauty of the IDE is that the run button does it all.



Review 2

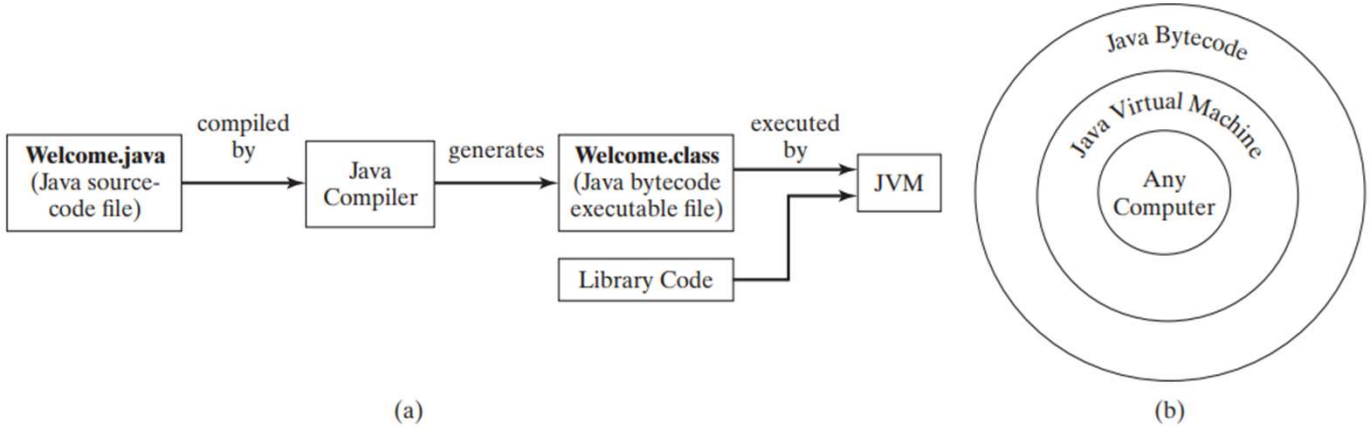
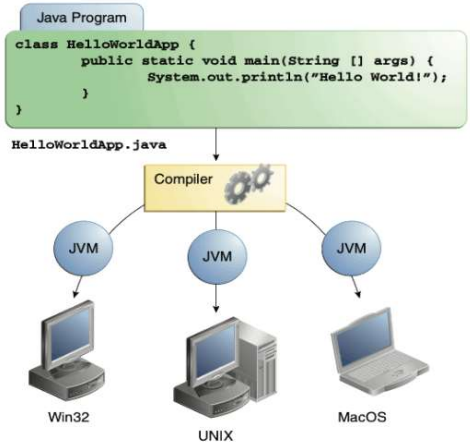


FIGURE 1.8 (a) Java source code is translated into bytecode. (b) Java bytecode can be executed on any computer with a Java Virtual Machine.



Through the Java VM, the same application is capable of running on multiple platforms.

The second cup of Java

Run the following code:

```java

/**
* Course: CSC 125 OOP
* Author: Dr. Fadi Alzhouri
* Week: 1
*/

// Define a public class named CSC125
public class CSC125
{
    // The main method is the entry point of the Java program
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Welcome to the CSC 125");
        System.out.println("**********************");
        System.out.println("   @ Gust University");
    }
}

```

The output

```
Welcome to the CSC 125
************ ***********
   @ Gust University
```

Code analysis

```
/**
* Course: CSC 125 OOP
* Author: Dr. Fadi Alzhouri
* Week: 1
*/

// Define a public class named CSC125
public class CSC125
{
    // The main method is the entry point of the Java program
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Welcome to the CSC 125");
        System.out.println("**********************");
        System.out.println("   @ Gust University");
    }
}
```

Multi line Comment

Single line Comment

Code analysis: Comments

- Comments are used to clarify different parts of the code for the person who is reading the program
- Its **not** used by compiler.
- It does **not** affect the program performance
- It's just useful information about the program
- // is a single-line comment while /* */ is a multi-lines comment.
- Its highly recommended to use comments throughout your code

Code analysis: Comments (cont.)

- As you work on your assignments and projects, please remember the importance of using comments in your code.
- Include your name, class section, instructor, date, and a brief description at the beginning of the program.
- Blank lines do not affect the code result but they make the code easier to understand.
- Java ignores extra white spaces.

```
// White Space
    1- Space
    2- Tab
    3- Blank line
```

Code analysis: Class Name

- Every Java program must have at least one class.
- Each class has a name.
- By convention, class names start with an uppercase letter.
- In this example, the class name is CSC125.

```
2
3  /**
4  * Course: CSC 125 OOP
5  * Author: Dr. Fadi Alzhouri
6  * Week: 1
7  */
8
9  // Define a public class named CSC125
10 public class CSC125
11 {
12     // The main method is the entry poi
13     public static void main(String[] ar
14         // Print a message to the conso
15         System.out.println("Welcome to
16         System.out.println("***********
17         System.out.println("   @ Gust U
18     }
19 }
```

Class name

Class body

Code analysis: Main Method

- In order to run a class, the class must contain a method named main.
- The main method is where the program starts executing.

```
// Define a public class named CSC125
public class CSC125
{
    // The main method is the entry point of the Java pr
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Welcome to the CSC 125");
        System.out.println("**********************");
        System.out.println("   @ Gust University");
    }
}
```

The main method

Method body

Code analysis: Statement

- A statement represents an action or a sequence of actions.
- The statement `System.out.println("Welcome to the CSC 125")` in the program is a statement to display the greeting "Welcome to the CSC 125 ".

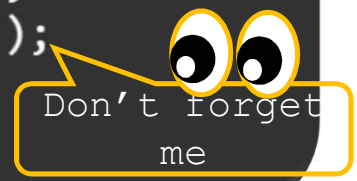
Statement

```java
// Define a public class named CSC125
public class CSC125
{
    // The main method is the entry point of the Java pr
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Welcome to the CSC 125");
        System.out.println("***********************");
        System.out.println("   @ Gust University");
    }
}
```

Code analysis: Statement Terminator

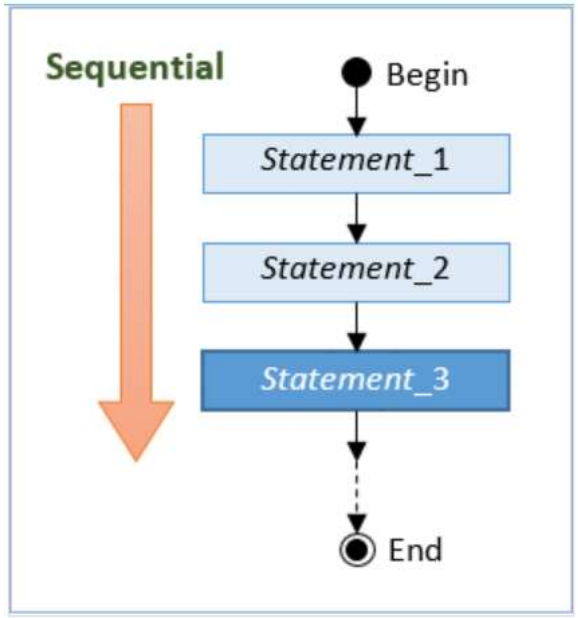
- Every statement in Java ends with a semicolon (;).

```java
// Define a public class named CSC125
public class CSC125
{
    // The main method is the entry point of the Java program
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Welcome to the CSC 125");
        System.out.println("**********************");
        System.out.println("   @ Gust University");
    }
}
```



Code analysis: Program

- A program is a sequence of instructions (called statements), executing one after another.
- Programming statements are executed in the order that they are written - from top to bottom in a sequential manner.



How many statement are there in the previous program?

Code analysis: Reserved words

- Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program.

Keywords

```
// Define a public class named CSC125
public class CSC125
{
    // The main method is the entry point of the Java program
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Welcome to the CSC 125");
        System.out.println("**********************");
        System.out.println("   @ Gust University");
    }
}
```

Code analysis: Reserved words

- Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program.

Keywords

```
// Define a public class named CSC125
public class CSC125
{
    // The main method is the entry point of the Java program
    public static void main(String[] args) {
        // Print a message to the console
        System.out.println("Welcome to the CSC 125");
        System.out.println("**********************");
        System.out.println("   @ Gust University");
    }
}
```