

CSC 125

Object Oriented Programming

Week 1

Dr. Fadi Alzhouri



Course Information

- Course outline
- Can I become a professional developer at the end of the course?

Email Format Guidelines

- To ensure effective communication and timely responses, please follow the format below when sending me emails:
- Subject Line Format: [Course Name] - [Section] - [Topic]
 - Example: CSC125 - 90 - Assignment Submission Question
- Email Content Guidelines:
 - Start with a proper salutation (e.g., "Dr. [Instructor's Name]").
 - Clearly state your purpose in the first line of your email.
 - Provide any necessary details (e.g., assignment name, due date, specific question).
 - Sign off with your full name and student ID.

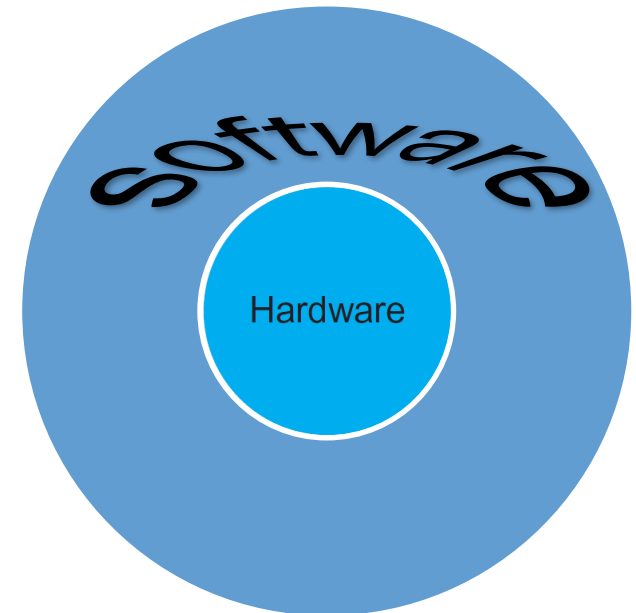
Textbook

- Introduction to Java Programming: Brief Version, (11th Edition)

Introduction

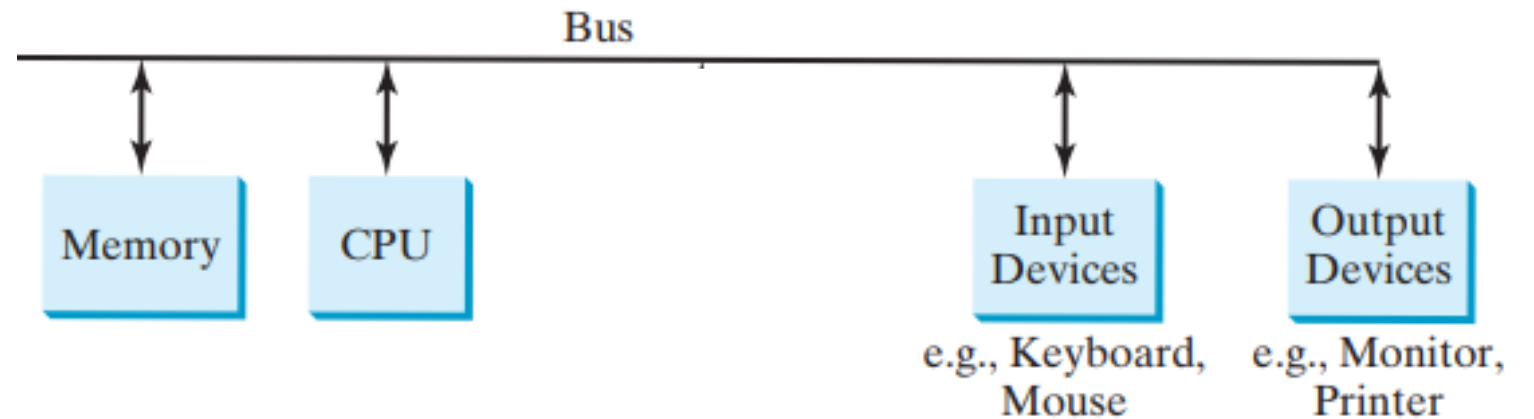
What are the two main components of any computer system?

- Hardware
- Software



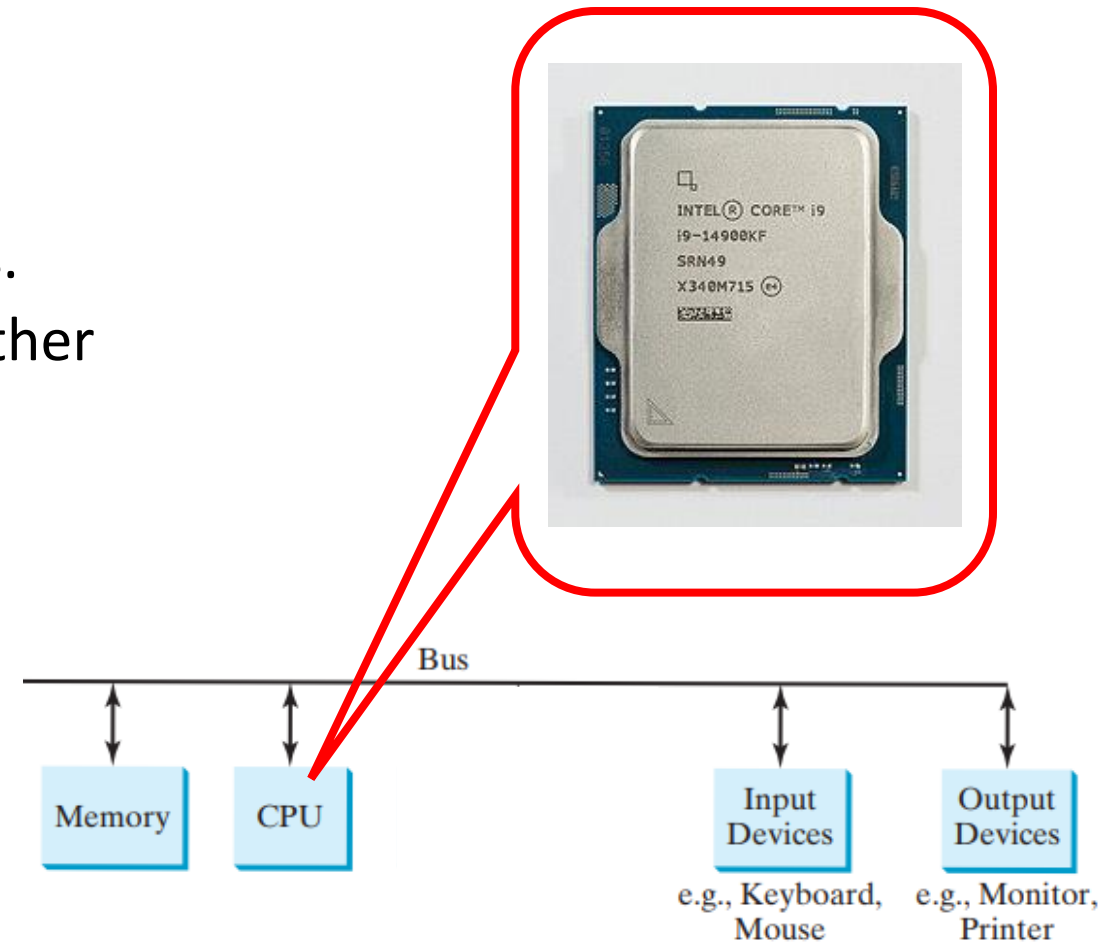
Hardware

- What are the hardware parts of a computer?



Hardware (cont.)

- Central Processing Unit (**CPU**)
 - It is the computer's brain
 - It performs arithmetic and logical calculations.
 - Controls and coordinates the actions of the other components.



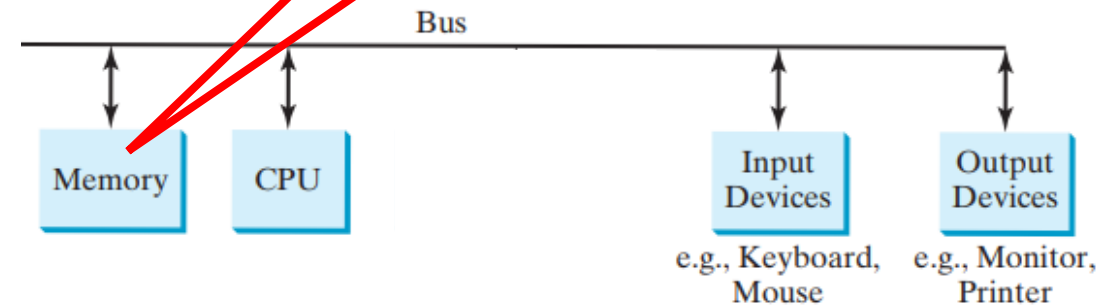
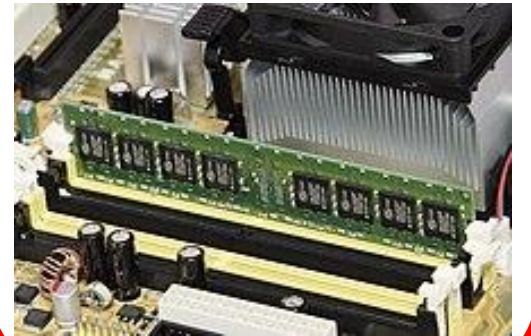
Hardware (cont.)

- **Memory**

- Memory stores data and program instructions in uniquely addressed memory locations.

- Types of Memory:

1. Volatile Memory (RAM)
2. Non-volatile memory (ROM, SSD, HDD)

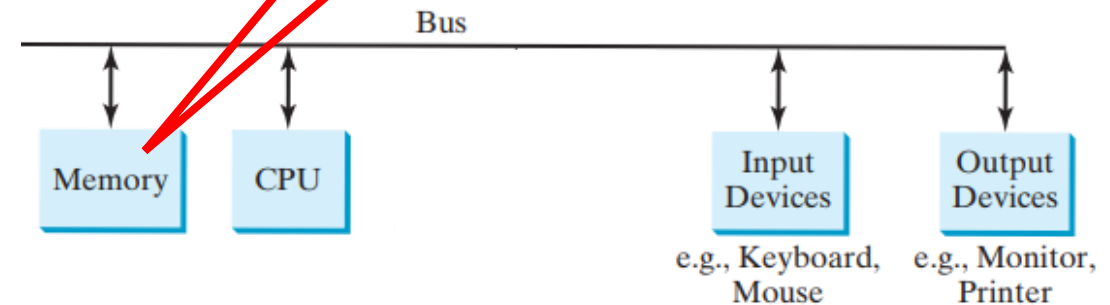


https://en.wikipedia.org/wiki/Computer_data_storage#Primary_storage

Hardware (cont.)

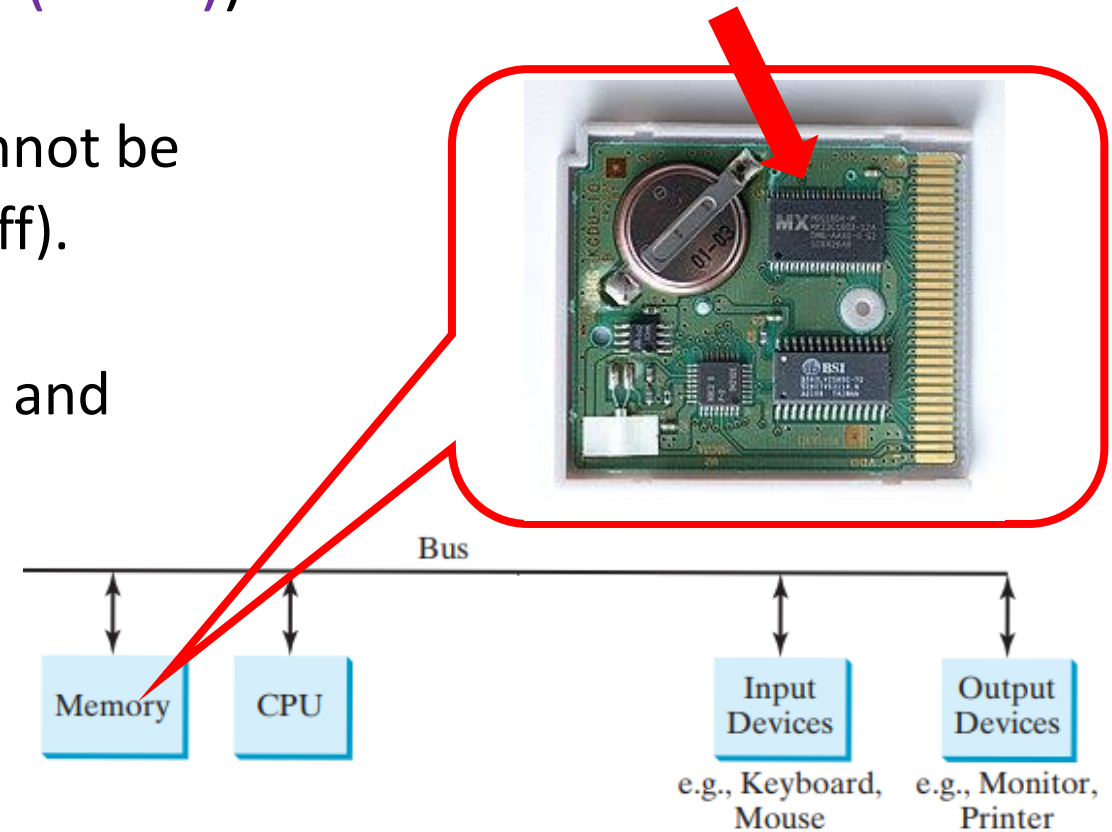
- Volatile Memory (**R**andom **A**ccess **M**emory (**RAM**)):

- **Temporary** storage.
- Data is **lost** when power is **turned off**.
- Used for running programs and processes.
 - All programs must be loaded into main memory (RAM) before they can be executed
 - All data must be brought into main memory before it can be manipulated
- Known as internal or **main memory**



Hardware (cont.)

- Non-volatile memory (**R**ead **O**nly **M**emory (**ROM**)):
 - **Permanent** storage.
 - Retains data even when power is off (data cannot be deleted or changed even if the PC is turned off).
 - Used to boot up the system.
 - Used for storing part of the operating system, and sometimes applications.
 - Instructions are typically recorded at factory



Hardware (cont.)

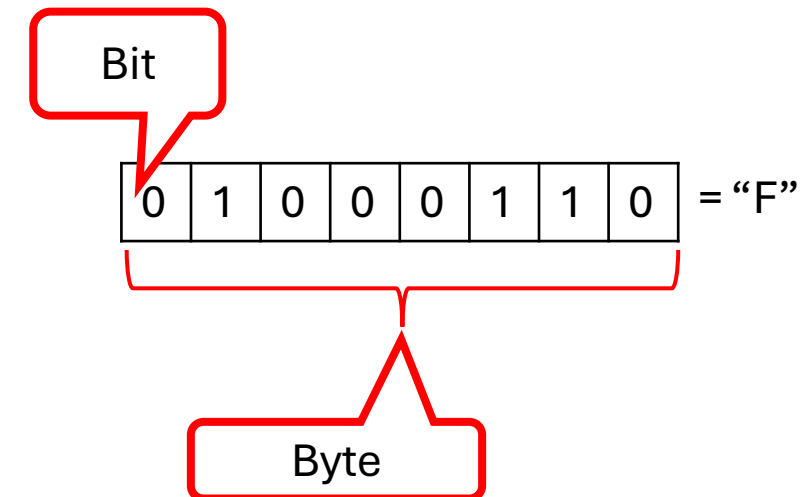
- Memory Address:
 - A computer's memory consists of an ordered sequence of locations (bytes).
 - Every location (**byte**) in the memory has a **unique address**.
 - Each memory location can contain either an **instruction** or **data**

Memory address	Memory content	
.	.	
.	.	
.	.	
2000	01000011	Encoding for character 'C'
2001	01110010	Encoding for character 'r'
2002	01100101	Encoding for character 'e'
2003	01110111	Encoding for character 'w'
2004	00000011	Decimal number 3
.	.	

Hardware (cont.)

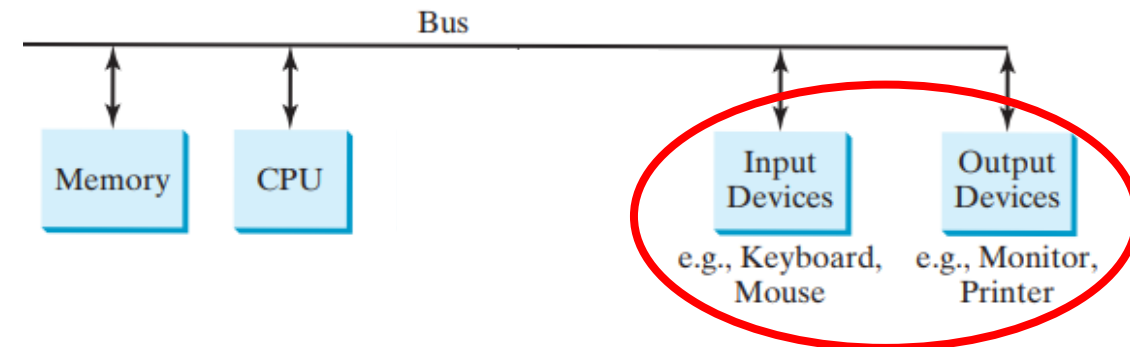
- Storing and retrieving information from Memory:
 - **Bit** (binary digit) is the smallest unit of memory.
 - It is a circuit that holds voltage levels that can be represented by 0 (Gnd) or 1 (+5v). It simply looks like a **switch** with two states (**ON/Off**) and is therefore called a **binary digit** (bit).
 - All types of data (numbers, characters, images, etc.) are converted into **binary format** (a series of bits).
 - A group of **8** bits represents a **byte**.
- Activity: represent your name in binary.

Memory address	Memory content
.	.
.	.
.	.
2000	01000011 Encoding for character 'C'
2001	01110010 Encoding for character 'r'



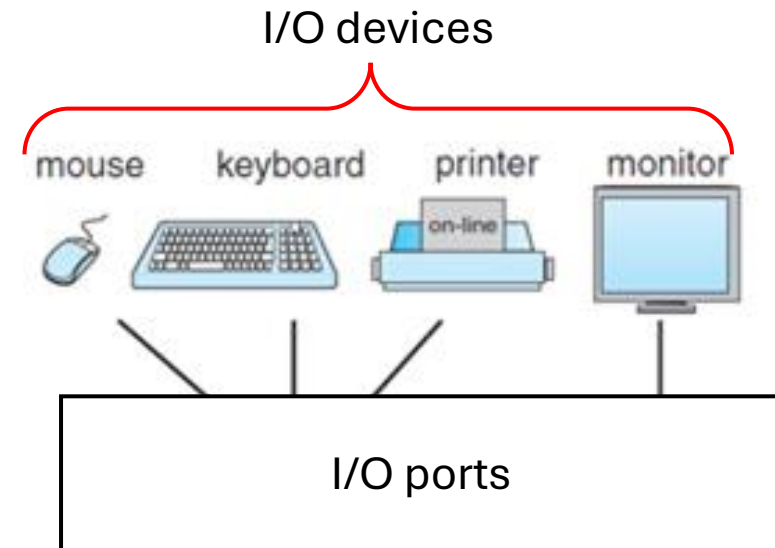
Hardware (cont.)

- Input & output (I/O) ports and devices:
 - I/O ports are interfaces through which data is transferred between a computer and external devices.
 - Types of I/O Ports:
 - Serial Ports
 - Parallel Ports
 - USB Ports
 - HDMI Ports
 - Ethernet Ports



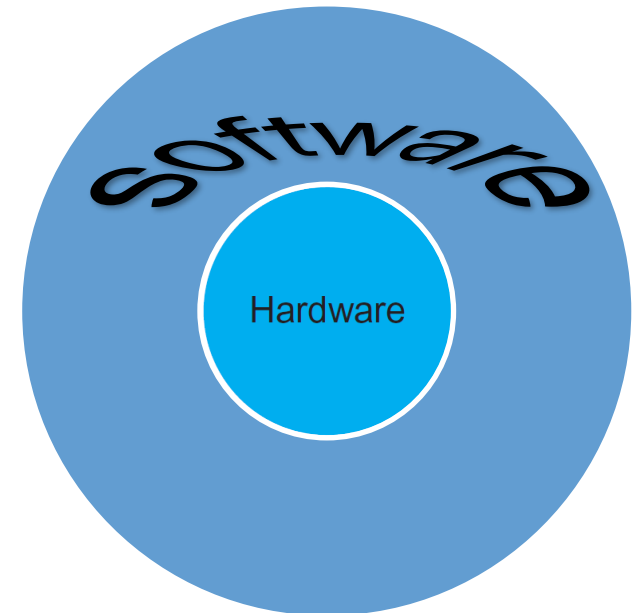
Hardware (cont.)

- Input & output (I/O) **ports** and **devices**:
 - I/O **devices** are hardware components that allow users to interact with a computer or allow the computer to communicate with other systems.
 - Types of Input Devices:
 - Keyboard
 - Mouse
 - Scanner
 - Microphone
 - Types of Output Devices:
 - Monitor
 - Printer
 - Speakers



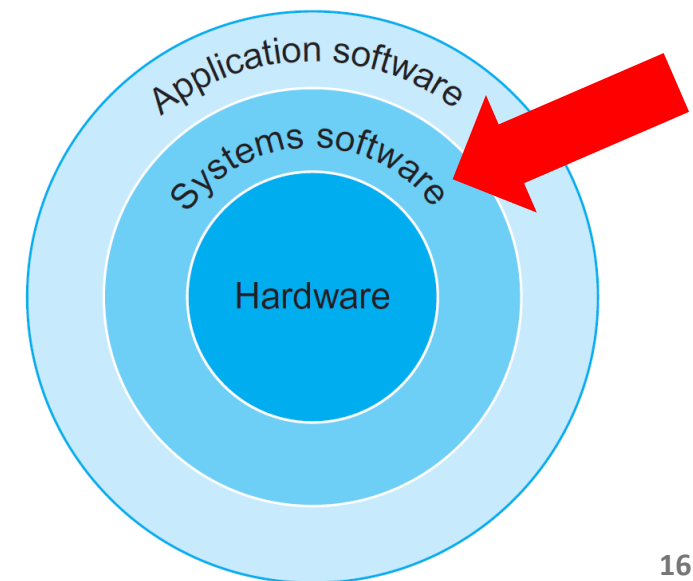
Software

- Software: Without software, the computer is useless
 - Software is developed with programming languages
 - Python is one of today's most popular software development languages
- Categories :
 - **System** software
 - **Application** software



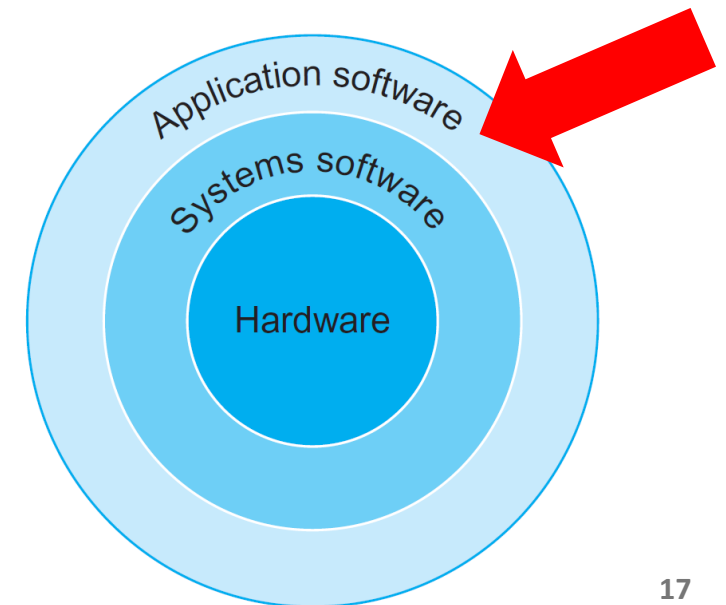
Software: System software

- **System Software:** System software is designed to manage and control the hardware components of a computer, providing a platform for running application software.
 - Examples of system software (aka. operating systems): Windows, macOS, Linux.
- Purpose: To provide a stable environment for applications to run and to manage system resources efficiently.
- Usually, part of the operating system is stored permanently in a (ROM) chip so it will be available as soon as the computer is turned on.
 - This part of the program has instructions that loads the rest of the OS instructions into RAM.
 - Loading the operating system into RAM is called **booting** the computer



Software: Application software

- **Application software** is designed to perform specific tasks or applications for users
- Examples:
 - Applications for creating **documents**, spreadsheets, and presentations (e.g., Microsoft Office, Google Workspace).
 - Applications for editing **images** and creating graphics (e.g., Adobe Photoshop, CorelDRAW).
 - Applications for **entertainment** purposes (e.g., video games, media players).
- Purpose: To enable users to perform tasks and solve specific problems in various domains.

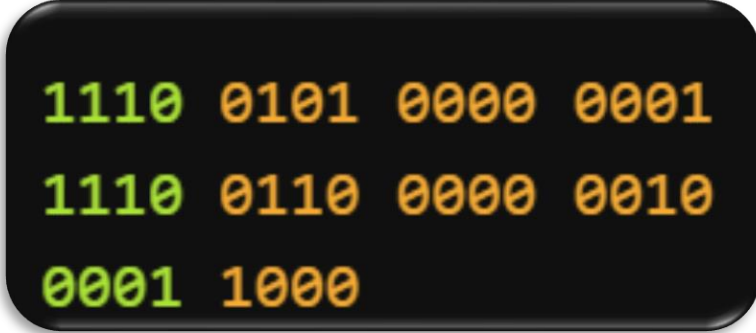


Software: Developing a Software

- Developing new software means **writing** a **program**
- A **program** is a list of instructions which are listed by the user and are performed sequentially by the computer
- In which **language** we must deliver the instructions to the computer?

Software: Developing a Software (cont.)

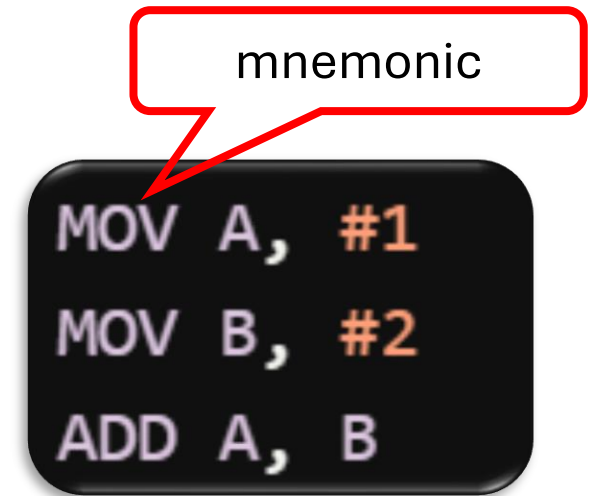
- The computer's native language is called **machine language**
 - **Why?**
 - Computers (CPUs) only understand 0s and 1s!
 - Early computers were programmed in machine language.
- Hard to code in binary code (machine language) so new languages were made starting with **assembly** language



```
1110 0101 0000 0001
1110 0110 0000 0010
0001 1000
```

Software: Assembly Language

- Assembly language is a **low-level** programming language that is closely related to machine code.
- It uses **mnemonic** codes and variable names to represent instructions.
- Characteristic: Provides direct control over hardware.
- Use Cases:
 1. Embedded systems
 2. Device drivers
 3. Performance-critical applications
- How does the computer understand the assembly language?



Software: Assembly Language (cont.)

- **Assembler**: An assembler is a tool that **translates** assembly language code into machine code (binary).
- The output of the assembler is machine code, which consists of binary instructions that the CPU can execute directly.

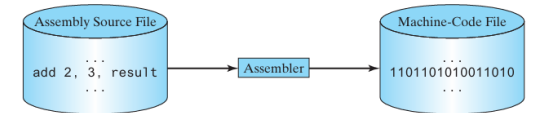
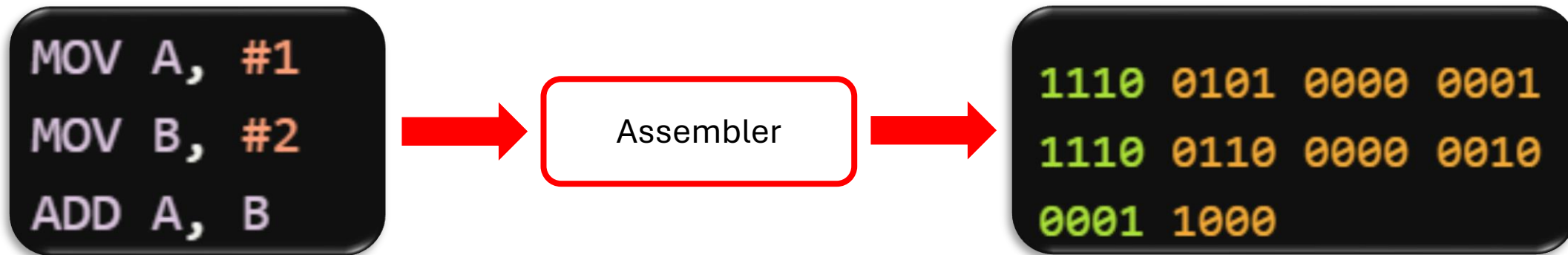


FIGURE 1.3 An assembler translates assembly-language instructions into machine code.



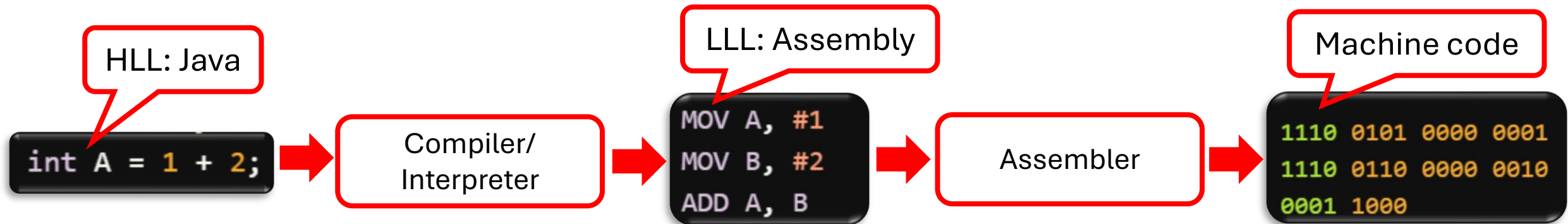
Is it still easy to write programs in assembly languages?

Software: High-level Language

- Developed to speed up the programming process.
- High-level languages use syntax and structures that are **closer** to **natural (human) language**, making it **easier** for programmers to read and understand the code.
- It reduces the complexity of software development.
- How does the computer understand the high-level language?

Software: High-level Language (cont.)

- A compiler translates high-level language (HLL) into assembly language.
- Some languages use an **intermediate** step where the source code is first translated into an intermediate representation (IR) or **bytecode**.
- The most common HLLs: Python, Java, and C++. [TIOBE Index – TIOBE](#)



Note: Compilers, interpreters, and assemblers are **software**.

Software: High-level Language (cont.)

- Interpreter: Translates high-level code into machine code **line** by **line** at runtime while a compiler translates the **entire** source code into a machine-code file, and the machine-code file is then executed
- Executes the code **immediately**, which means it can start running the program without a complete translation.
- Generally **slower** than compiled code because it translates code on-the-fly.
- Runs code line by line, good for scripting and quick testing.
- **JavaScript**: Most commonly used for client-side scripting. Browsers interpret JavaScript code, allowing for dynamic and interactive web pages

Some IT Terminologies

- **Syntax**: The set of rules that defines the structure of a programming language.
- **JDK** (Java Development Kit): A software development kit used for developing **Java** applications. It includes tools for compiling, running, and debugging Java programs.
 - It contains:
 - Java Compiler (**javac**): Converts Java source code into bytecode.
 - Java Runtime Environment (**JRE**): Allows you to run Java applications.
- **IDE** (Integrated Development Environment): A software application that provides comprehensive facilities to programmers for software development.
 - Features: Typically includes a **code editor**, **debugger**, **compiler** or interpreter, and tools for version control.
 - Examples: **NetBeans**, **Eclipse**.
- When you write Java code in NetBeans, the IDE uses the JDK's compiler to compile your code into bytecode.

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](#)
 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



Just for training

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**).

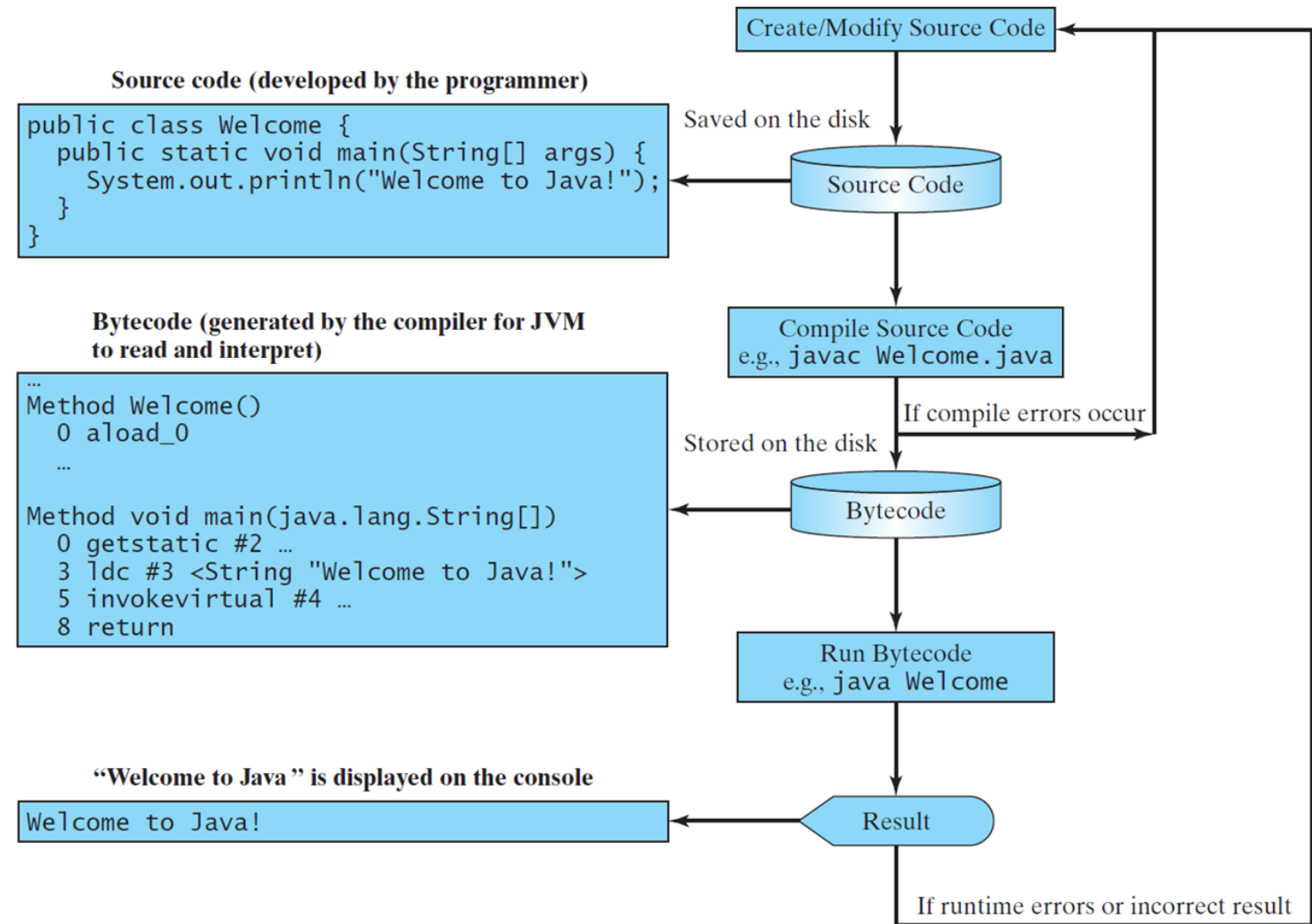
5. Run the java program using **Java** command

```
java HelloWorld
```

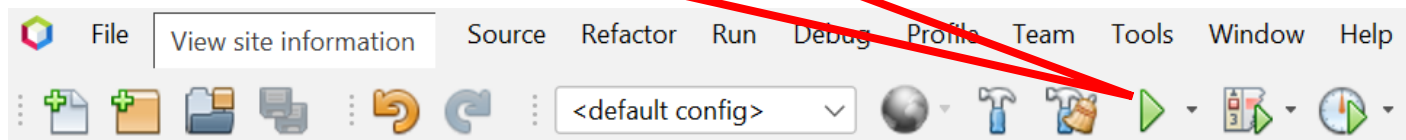
- You should see the output

```
Hello, World!
```


Review 1



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



Review 2

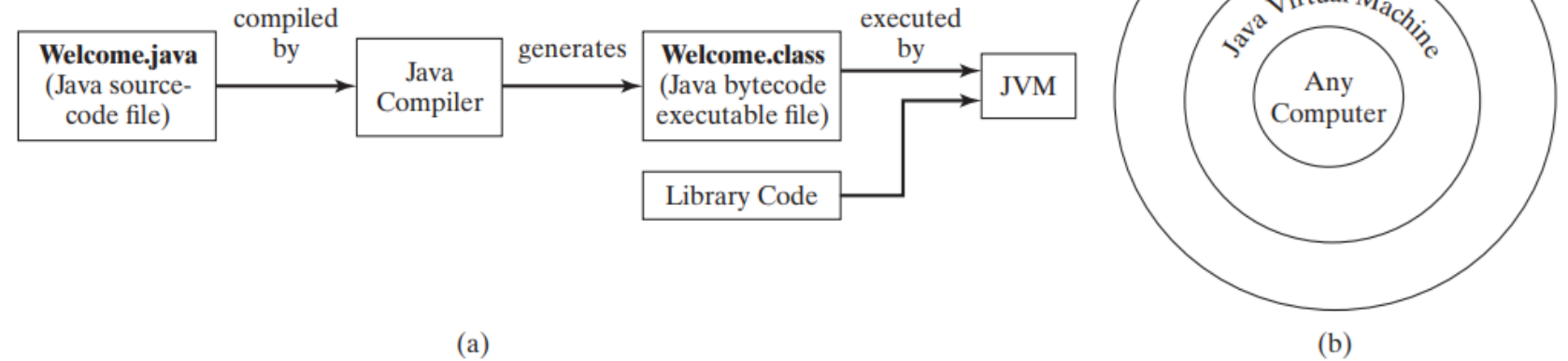
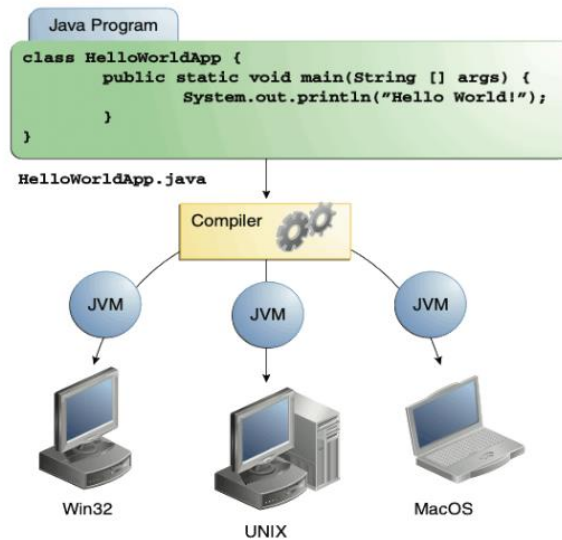


FIGURE 1.8 (a) Java source code is translated into bytecode. (b) Java bytecode can be executed on any computer with a Java Virtual Machine.



The second cup of Java



Run the following code:

```
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 point of the Java program
13     public static void main(String[] args) {
14         // Print a message to the console
15         System.out.println("Welcome to the CSC 125");
16         System.out.println("*****");
17         System.out.println("    @ Gust University");
18     }
19 }
20
```

The output

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

Code analysis

```
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 point of the Java program
13     public static void main(String[] args) {
14         // Print a message to the console
15         System.out.println("Welcome to the CSC 125");
16         System.out.println("*****");
17         System.out.println("    @ Gust University");
18     }
19 }
```

Multi line Comment

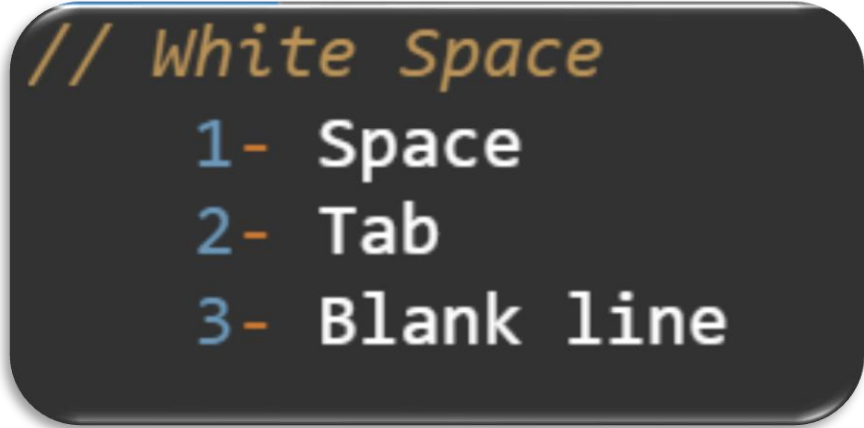
Single line Comment

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 does not affect the code result but it make the code easier to understand



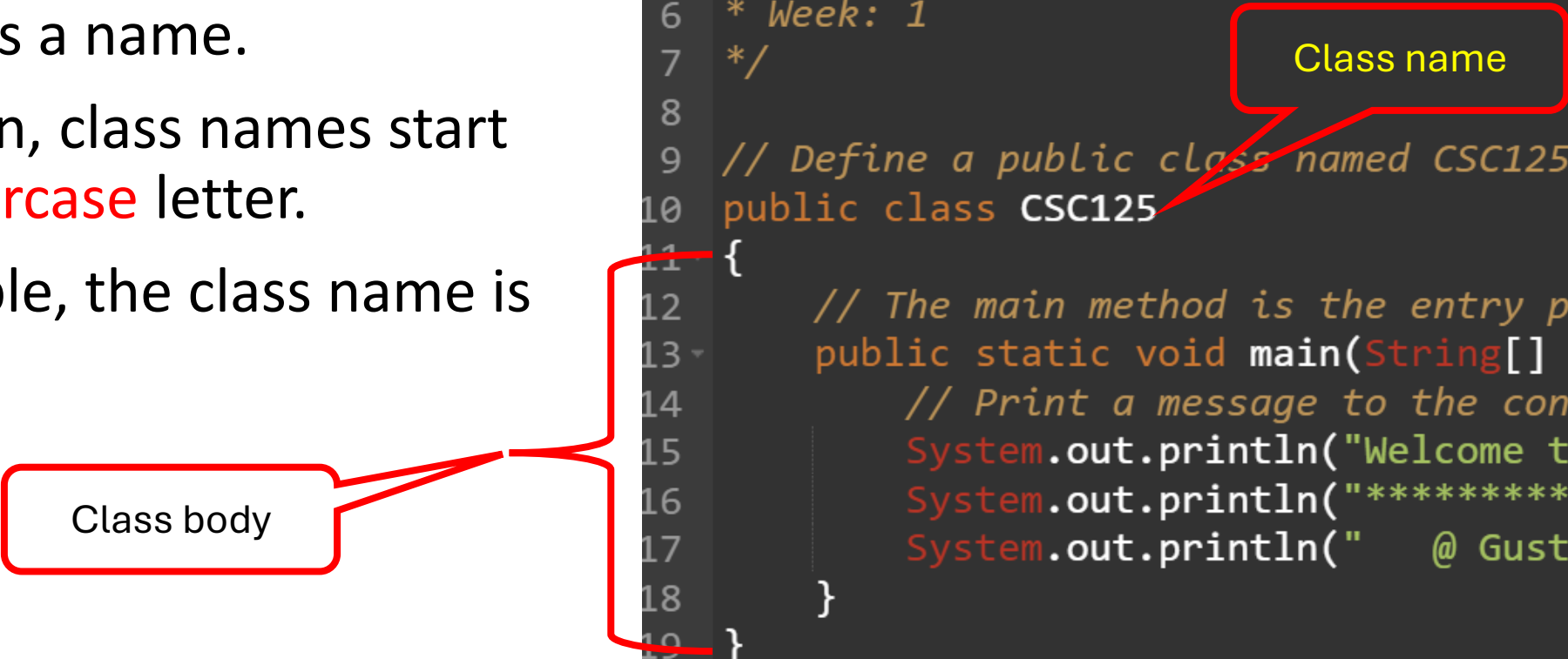
// White Space
1- Space
2- Tab
3- Blank line

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.

Class body

```
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 point
13     public static void main(String[] args) {
14         // Print a message to the console
15         System.out.println("Welcome to CSC125");
16         System.out.println("*****");
17         System.out.println("    @ Gust U");
18     }
19 }
```



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.

```
9 // Define a public class named CSC125
10 public class CSC125
11 {
12     // The main method is the entry point of the Java pr
13     public static void main(String[] args) {
14         // Print a message to the console
15         System.out.println("Welcome to the CSC 125");
16         System.out.println("*****");
17         System.out.println("    @ Gust University");
18     }
19 }
```

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

```
9 // Define a public class named CSC125
10 public class CSC125
11 {
12     // The main method is the entry point of the Java pr
13     public static void main(String[] args) {
14         // Print a message to the console
15         System.out.println("Welcome to the CSC 125");
16         System.out.println("*****");
17         System.out.println("    @ Gust University");
18     }
19 }
```

Code analysis: Statement Terminator

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

```
9 // Define a public class named CSC125
10 public class CSC125
11 {
12     // The main method is the entry point of the Java program
13     public static void main(String[] args) {
14         // Print a message to the console
15         System.out.println("Welcome to the CSC 125");
16         System.out.println("*****");
17         System.out.println("    @ Gust University");
18     }
19 }
```

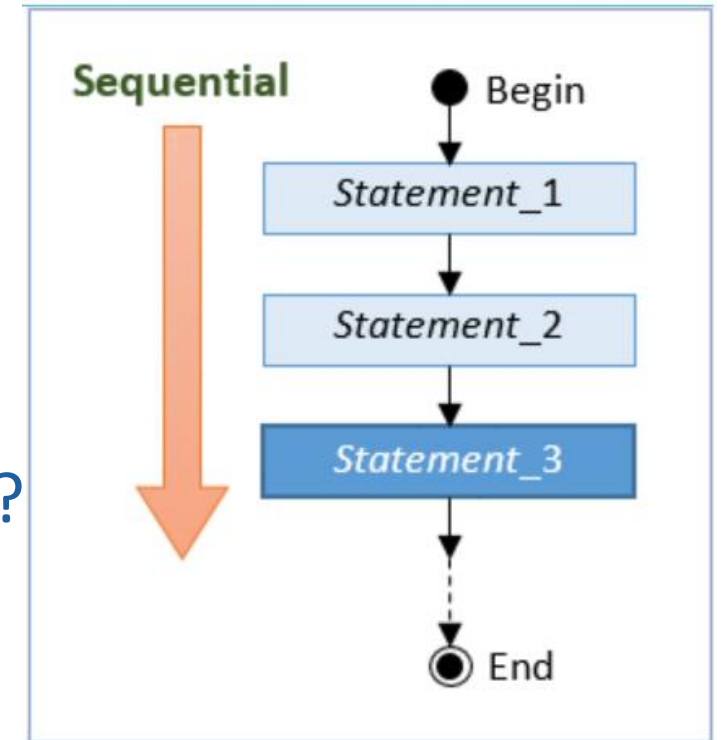


Don't forget me

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

```
9 // Define a public class named CSC125
10 public class CSC125
11 {
12     // The main method is the entry point of the Java program
13     public static void main(String[] args) {
14         // Print a message to the console
15         System.out.println("Welcome to the CSC 125");
16         System.out.println("*****");
17         System.out.println("    @ Gust University");
18     }
19 }
```