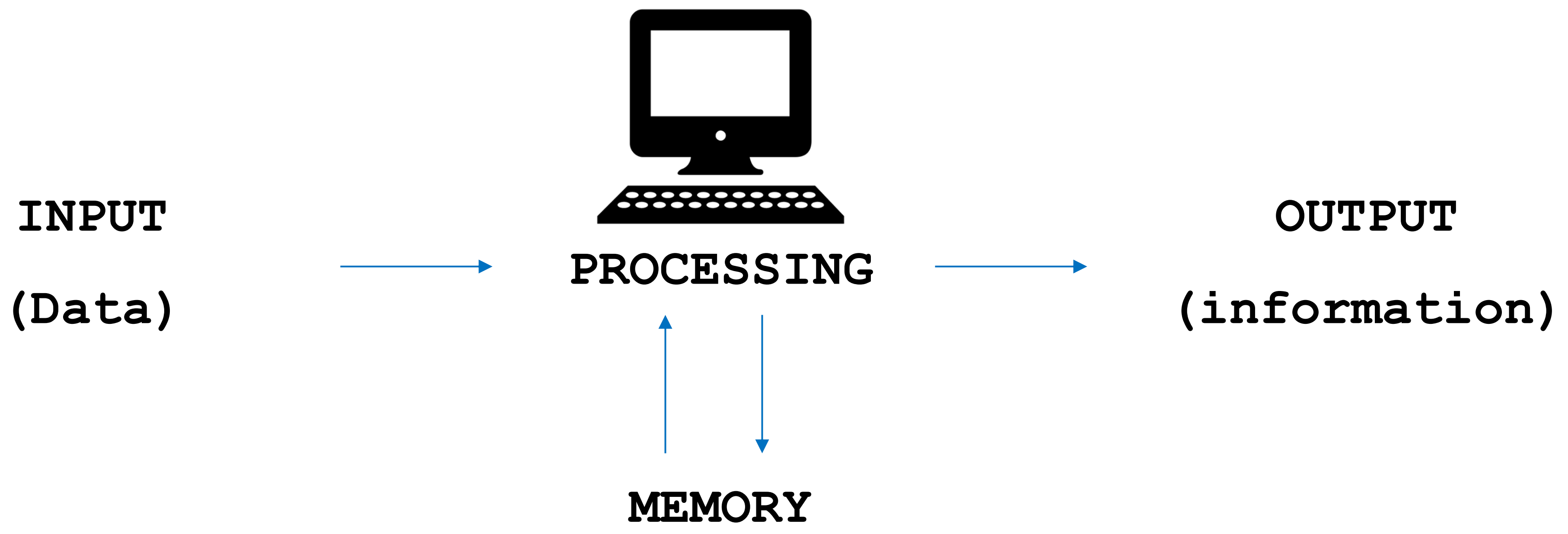
What You Will Learn

- What is a computer
- What is a computer program
- Type of programming languages
- Introduction to Java
- Compiler
- Interpreter
- Structure of Java program
- Print statements
- Type errors

What IS A COMPUTER

Computers are devices that process data to generate information.



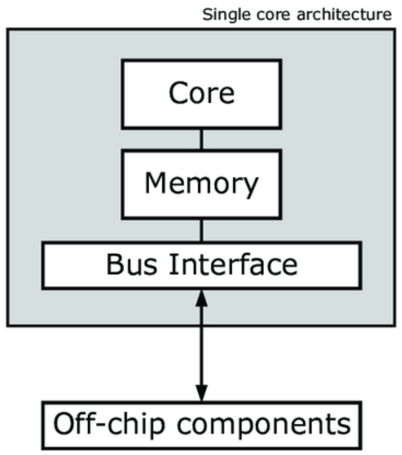
Functional View of a Computer

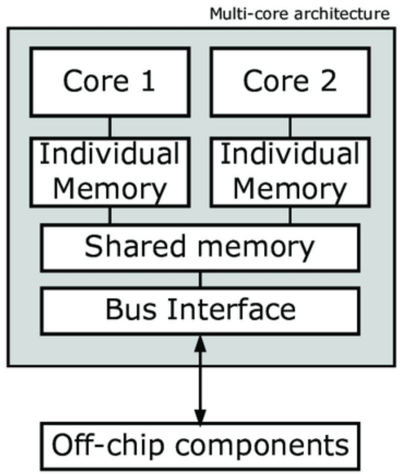
Central Processing unit (CPU)

- CPU is the "brain" of a computer.
- The CPU carries out all the basic operations on the data.
- The CPU communicates with the main memory

Number of processors (cores)

1 – uni
2 – dual
4 – quad
…
Multi/parallel

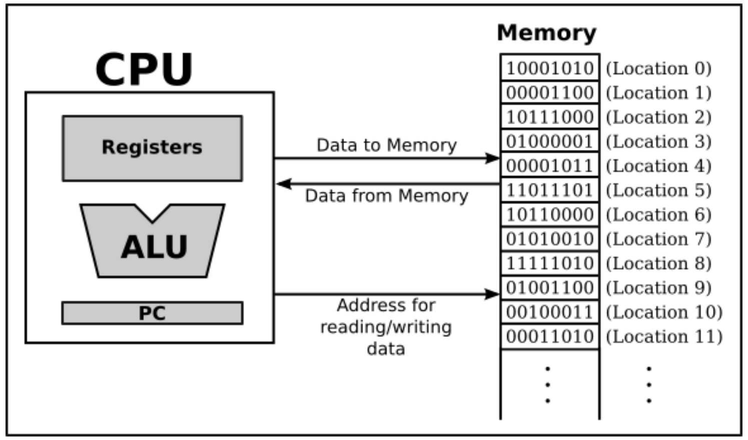




Functional View of a Computer

Memory stores programs and data.

- CPU can only directly access information stored in main memory (RAM or Random Access Memory).
- Main memory is fast, but volatile, i.e. when the power is interrupted, the contents of memory are lost.
- Secondary memory provides more permanent storage: magnetic (hard drive, floppy), optical (CD, DVD)



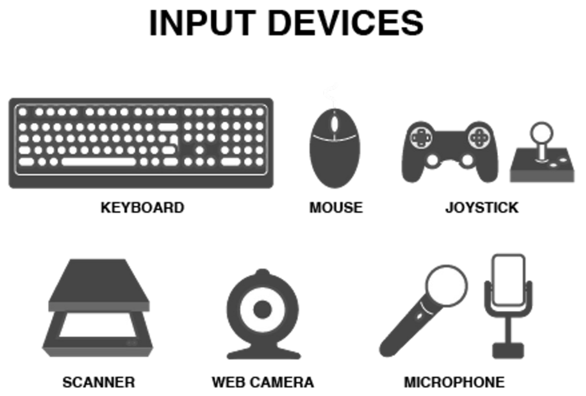
Functional View of a Computer

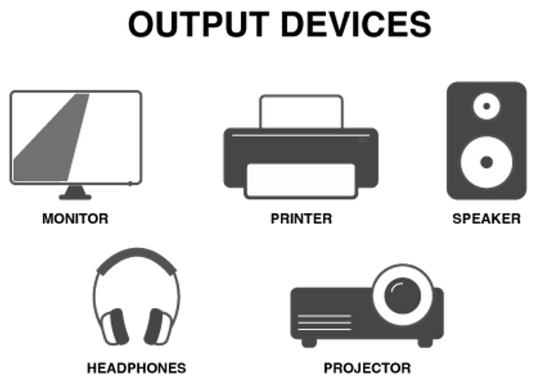
Input devices

Information is passed to the computer through keyboards, mice, etc.

Output devices

Processed information is presented to the user through the monitor, printer, etc.



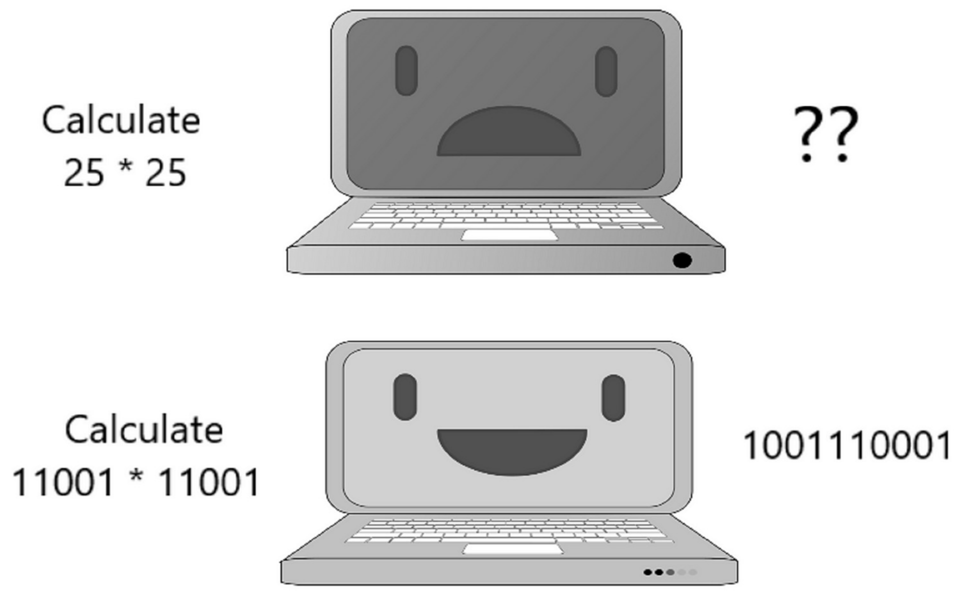


What computers understand

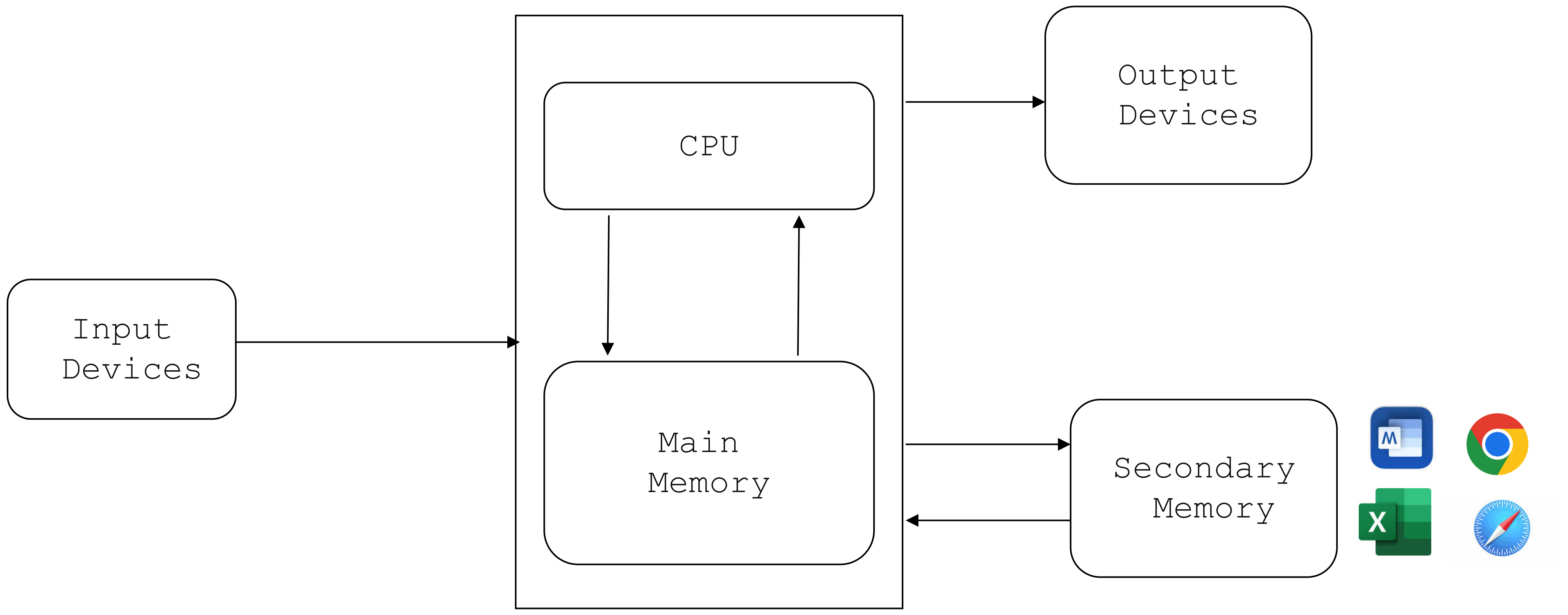
Computers understand things using a special language made of just two symbols: 0 and 1. It's like their own secret code. Everything, like words, pictures, and more, is turned into combinations of these symbols. Computers use this code to do math, show us stuff on screens, play games, and more.

Data Storage Units

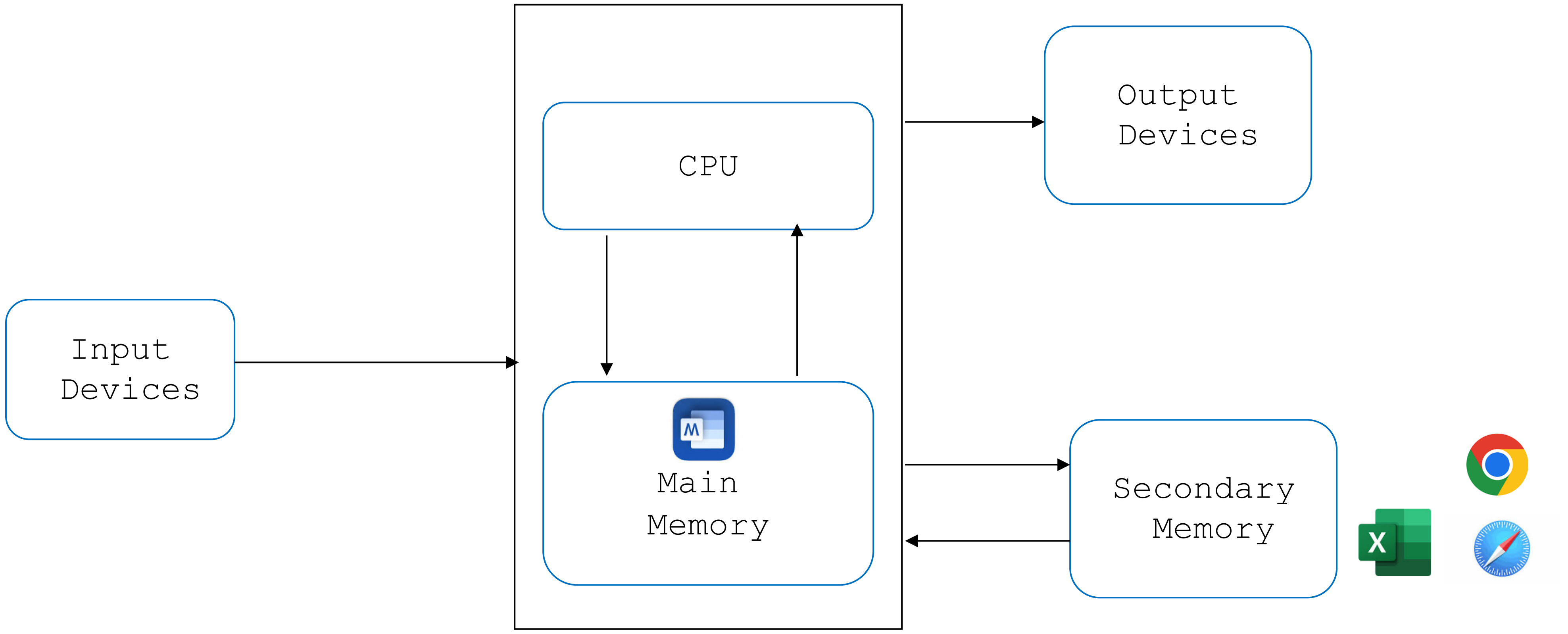
- Bit = 0 or 1
- Byte = 8 bits
- K (kilo) byte = 2^10 = 1024 bytes
- M (mega) byte = 10^6 bytes
- G (giga) byte = 10^9 bytes
- T (tera) byte = 10^12 bytes



Functional View of a Computer



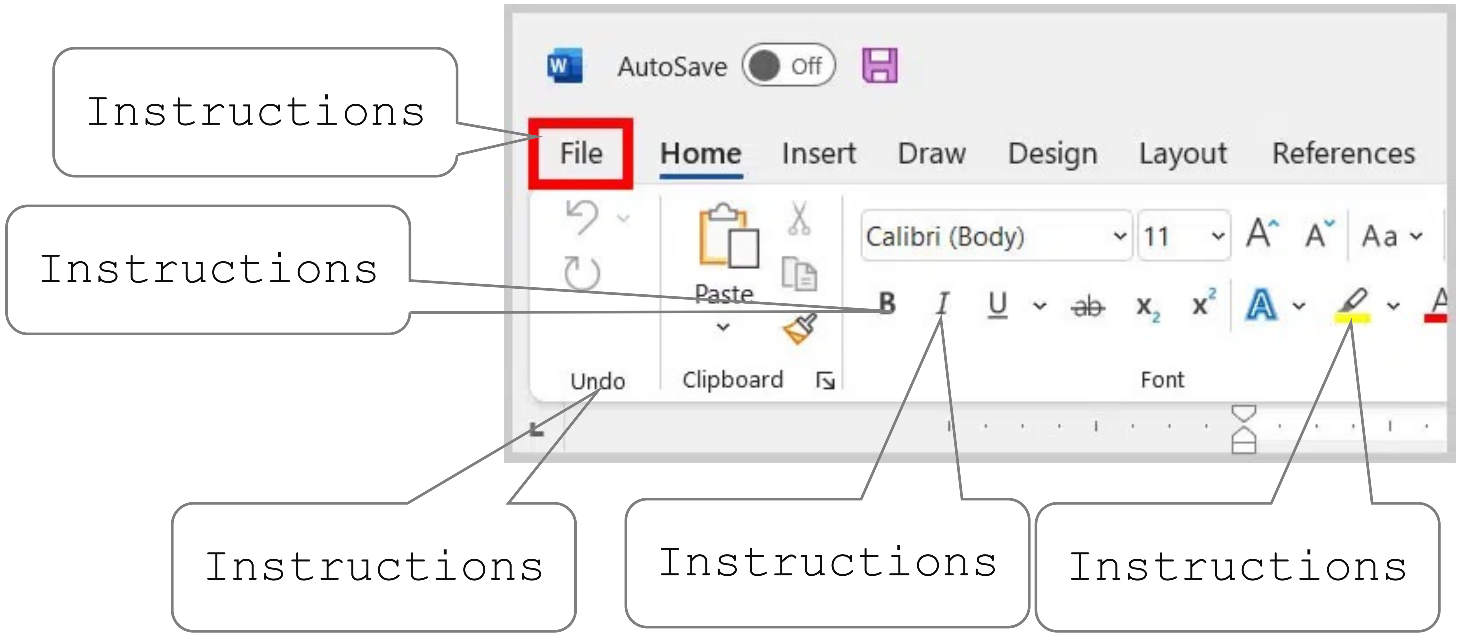
Functional View of a Computer



How Microsoft Word Looks Like

Instruction
Instruction
Instruction
Instruction
Instruction
Instruction
Instruction
Instruction

How Microsoft Word Looks Like

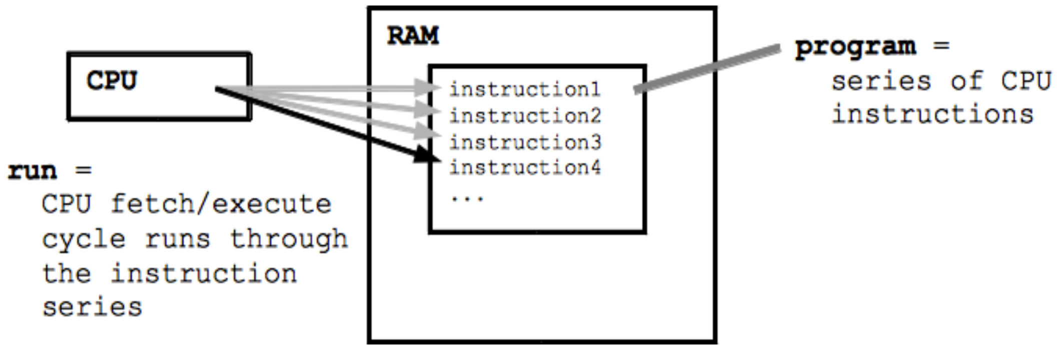


What is a computer program?

- A detailed, step-by-step set of instructions telling a computer what to do.
- If we change the program, the computer performs a different set of actions or a different task.
- The machine stays the same, but the program changes!

- Software (programs) rule the hardware (the physical machine).

- The process of creating this software is called programming.


CPU
RAM
instruction1
instruction2
instruction3
instruction4
...
program = series of CPU instructions
run = CPU fetch/execute cycle runs through the instruction series

Programming Languages

- High Level Languages
  - Not directly understood by the computer
  - Humanly readable
  - Requires the use of a compiler

- Low Level Languages
  - Is not not directly understood by the computer
  - Is not Humanly readable
  - Does not require a compiler

What is JAVA?

- Java is a high-level, object-oriented programming language that is designed to be platform-independent, meaning it can run on various computing platforms without modification. It was developed by Sun Microsystems (later acquired by Oracle Corporation) in the mid-1990s.

- Java employs a two-step execution process. First, a compiler translates human-readable source code into bytecode. Then, an interpreter, known as the Java Virtual Machine (JVM), executes this bytecode on various platforms, ensuring portability and consistent behavior across different systems.

What is JAVA?

1. Platform Independence: Java programs are compiled into bytecode, which can be executed on any system with a Java Virtual Machine (JVM). This "write once, run anywhere" principle enables Java applications to be portable across different platforms.

2. Object-Oriented: Java follows the object-oriented programming paradigm, emphasizing the use of objects and classes for structuring code, promoting code reusability and modularity.

3. Strongly Typed: Java enforces strong data typing, requiring explicit declaration of variable types and preventing type-related errors.

4. Automatic Memory Management: Java incorporates a garbage collector that automatically manages memory allocation and deallocation, helping developers avoid memory leaks and manual memory management.

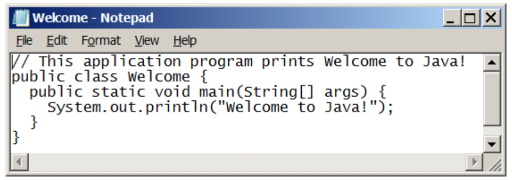
5. Security: Java includes built-in security features, such as the ability to run code in a sandboxed environment, helping prevent malicious code from damaging systems.

6. Rich Standard Library: Java provides a comprehensive standard library (Java Standard Library or Java API) that includes pre-built classes and methods for various tasks, simplifying development.

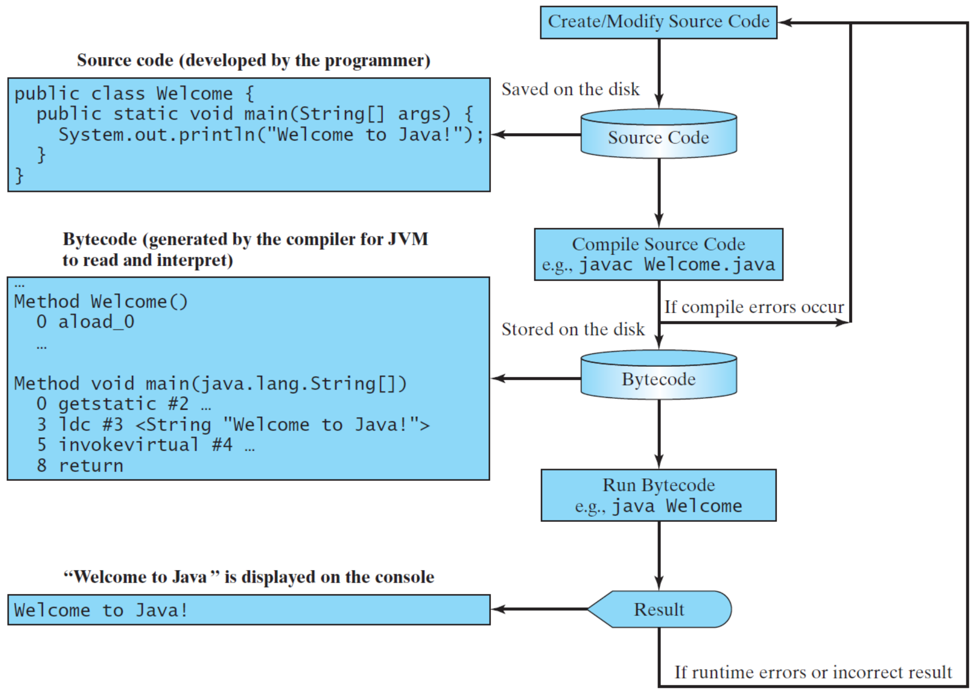
7. Community and Ecosystem: Java has a large and active developer community, contributing to a wide range of libraries, frameworks, and tools that extend its capabilities.

What is JAVA?

- High level language can be understood by the human but not the computer directly.
- Requires the use of a compiler and interpreter



```
// This application program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```



Create/Modify Source Code
Source code (developed by the programmer)
Saved on the disk
Source Code
Compile Source Code e.g., javac Welcome.java
Bytecode (generated by the compiler for JVM to read and interpret)
If compile errors occur
Stored on the disk
Bytecode
Run Bytecode e.g., java Welcome
"Welcome to Java" is displayed on the console
Result
If runtime errors or incorrect result

```
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

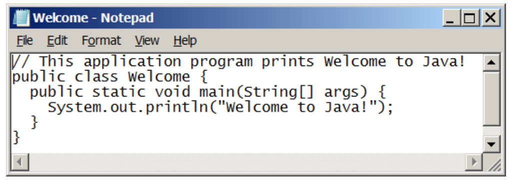
```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

```
Welcome to Java!
```

Dr. Sami Albouq

What is JAVA?

- High level language can be understood by the human but not the computer directly.
- Requires the use of a compiler and interpreter



```
// This application program prints Welcome to Java!
public class Welcome {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

Compile

```
…
Method Welcome()
  0 aload_0
  …

Method void main(java.lang.String[])
  0 getstatic #2 …
  3 ldc #3 <String "Welcome to Java!">
  5 invokevirtual #4 …
  8 return
```

Run

Java JVM
(Java Virtual Machine)

JAVA Concept

- **Class**: named group of related methods.

```
public class <name> {


}
```

JAVA Concept

- **Class**: named group of related methods.
- **Method**: group of programming statements that is given a name.

Every executable Java program consists of a **class**, that contains a method named **main**, that contains the statements (commands) to be executed.

```
public class <name> {
        public static void main(String[] args) {}
        public static void a(String[] args) {}
        public static void b(String[] args) {}
        public static void c(String[] args) {}

}
```

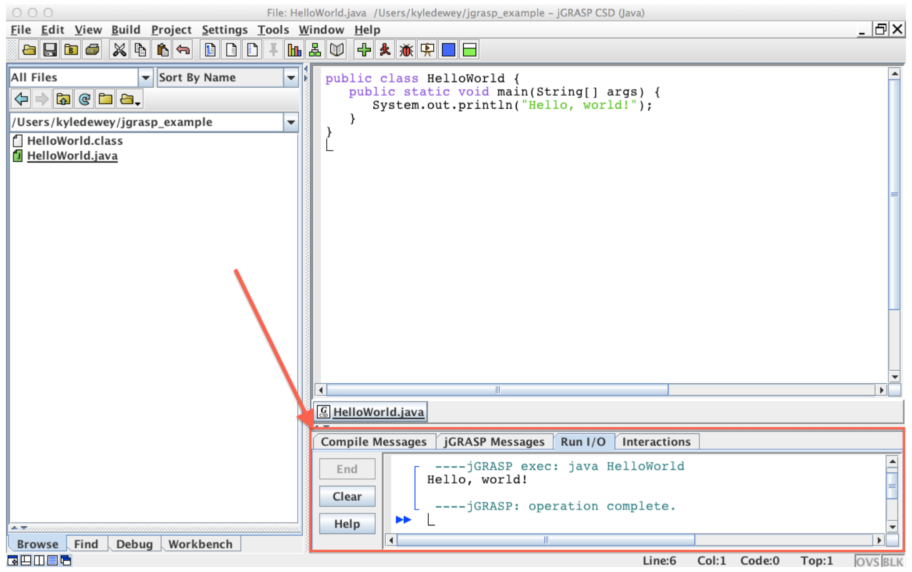
A Simple JAVA Program

```
public class <name> {
    public static void main(String[] args) {}
    public static void a(String[] args) {}
    public static void b(String[] args) {}
    public static void c(String[] args) {}
}
```

```
public class Hello{
    public static void main(String[] args) {



    }
}
```

A Simple JAVA Program

```
public class Hello{
    public static void main(String[] args) {

      System.out.println("Hello, world");

    }
}
```



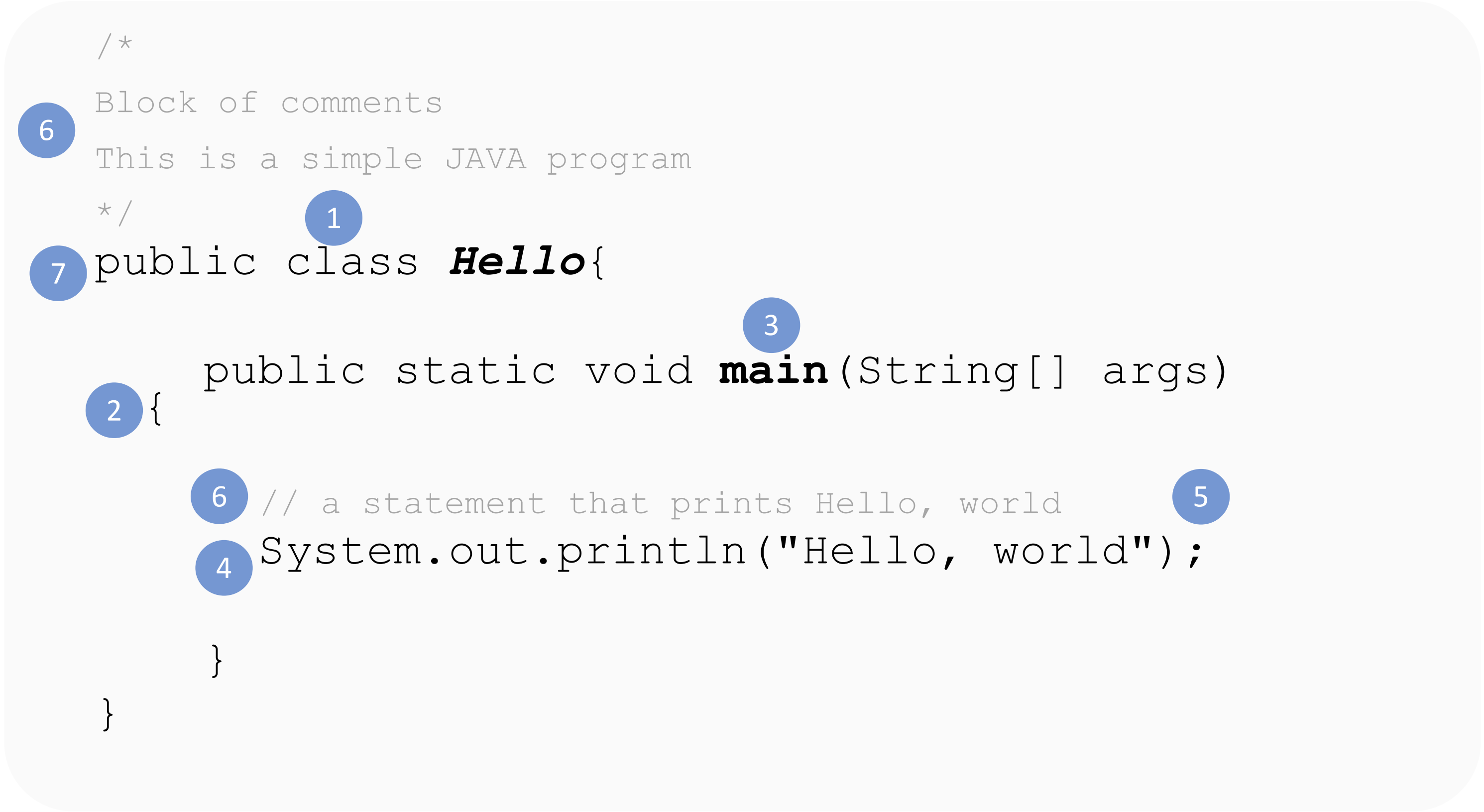
```
public class HelloWorld {
   public static void main(String[] args) {
      System.out.println("Hello, world!");
   }
}
```

```
 ----jGRASP exec: java HelloWorld
Hello, world!

 ----jGRASP: operation complete.
```

Dr. Sami Albouq

A Simple JAVA Program

1. Class name
2. Blocks
3. Main method
4. Statements
5. Statement terminator
6. Comments
7. Reserved words



```
/*
Block of comments
This is a simple JAVA program
*/
public class Hello{

     public static void main(String[] args)
  {

        // a statement that prints Hello, world
        System.out.println("Hello, world");

     }
}
```

How the Programme is Executed

```
/*
Block of comments
This is a simple JAVA program
*/
```

1

```
public class Hello{

     public static void main(String[] args) {

        // a statement that prints Hello, world
        System.out.println("Hello, world");

     }
}
```

How the Programme is Executed

```
/*
Block of comments
This is a simple JAVA program
*/
public class Hello{

2         public static void main(String[] args) {

            // a statement that prints Hello, world
            System.out.println("Hello, world");

          }
}
```

How the Programme is Executed

```
/*
Block of comments
This is a simple JAVA program
*/
public class Hello{

    public static void main(String[] args) {

       // a statement that prints Hello, world
3      System.out.println("Hello, world");


    }
}
```

How the Programme is Executed

```
/*
Block of comments
This is a simple JAVA program
*/
public class Hello{

    public static void main(String[] args) {

        // a statement that prints Hello, world
        System.out.println("Hello, world");

5       }
6   }
```

println Statement

A statement is used to output text or variable values to the console, followed by a line break.

Prints the given message as output.

- `System.out.println("<text>");`

Example:

- Print an empty line
  - `System.out.println("");     // output nothing`
- Print 1 + 2 as a text
  - `System.out.println("1 + 2"); // output 1 + 2`
- To perform an arithmetic operation you only use numbers with the sign
  - `System.out.println(1 + 2);   // output 3`

print Statement

A statement is used to output text or variable values to the console without a line break.

Prints the given message as output.

- `System.out.print("<text>");`

Example:

- Print an empty line
  - `System.out.print("");     // output nothing`
- Print 1 + 2 as a text
  - `System.out.print("1 + 2"); // output 1 + 2`
- To perform an arithmetic operation you only use numbers with the sign
  - `System.out.print(1 + 2);   // output 3`

Warning!

```
System.out.print(”Test” + 1 + 2);     OR      System.out.println(”Test” + 1 + 2);
 Output
 Test12
```

This result might be a bit surprising at first glance, but it can be explained by the way Java evaluates expressions from left to right.
Here's a step-by-step explanation of how this expression is evaluated:

1. "Test" is a string literal (text).
2. 1 and 2 are integer literals (numbers).

Now, let's break down the evaluation:

1. "Test" (a string) is concatenated (added) with 1, resulting in "Test1". At this point, you have a string (text).

2. "Test1" (the resulting string from step 1) is then concatenated with 2, resulting in "Test12".

In Java, when you use the + operator with strings and other types (like integers), it performs string concatenation. So, in this case, the numbers 1 and 2 are treated as strings after the initial concatenation with "Test".

If you wanted to perform arithmetic addition instead of string concatenation, you would need to use parentheses () to enforce the order of operations. For example:

```
System.out.println("Test" + (1 + 2));
```

In this case, (1 + 2) would be evaluated first as an arithmetic operation, resulting in 3, and then "Test" would be concatenated with 3, giving you the output: Test3.

Warning!

```
System.out.print(1 + 2 + " Test" );     OR       System.out.println(1 + 2 + "Test");
 Output
 3Test
```

This result is also determined by the left-to-right evaluation of expressions in Java.

Here's a step-by-step explanation of how this expression is evaluated:

1. 1 and 2 are integer literals.
2. 1 + 2 is evaluated first as an arithmetic operation, resulting in 3. At this point, you have an integer 3.

3. 3 (the resulting integer from step 2) is then concatenated with "Test". Since 3 is implicitly converted to a string during this concatenation, you get "3Test".

So, in this case, the arithmetic addition is performed first, and then the result is concatenated with the string "Test".

Example

```
System.out.print("I study at IU");

System.out.println("."); // output 1 + 2

System.out.print(1 + 2);   // output 3
```

Output:

```
I study at IU.
3
```

Practice

- Write a print statement that prints your name
- Write a print statement that prints two words
- Write a print statement that prints the result of adding two numbers
- Write a print statement that prints the result of subtracting two numbers

Practice

- Write a print statement that prints your name
    - `System.out.println(“Sami”);`

Practice

- Write a print statement that prints your name
    - System.out.println(“Sami”);
- Write a print statement that prints two words

Practice

- Write a print statement that prints your name
  - System.out.println(“Sami”);
- Write a print statement that prints two words
  - System.out.println(“Hello world”);

Practice

- Write a print statement that prints your name
  - System.out.println(“Sami”);
- Write a print statement that prints two words
  - System.out.println(“Hello world”);
- Write a print statement that prints the result of adding two numbers

Practice

- Write a print statement that prints your name
  - `System.out.println(“Sami”);`
- Write a print statement that prints two words
  - `System.out.println(“Hello world”);`
- Write a print statement that prints the result of adding two numbers
  - `System.out.println( 1 + 3);`

Practice

- Write a print statement that prints your name
  - `System.out.println(“Sami”);`
- Write a print statement that prints two words
  - `System.out.println(“Hello world”);`
- Write a print statement that prints the result of adding two numbers
  - `System.out.println( 1 + 3);`
- Write a print statement that prints the result of subtracting two numbers
  - `System.out.println( 1 - 3);`

Types of Program Errors

Compiler errors

is known as compilation errors, occur in Java when the Java compiler encounters syntax or semantic issues in the source code, preventing it from generating bytecode (executable code). These errors need to be fixed before the program can be successfully compiled and executed.

Compiler Error Example

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

In this example, there's a missing semicolon (;) at the end of the System.out.println statement. The compiler will report a syntax error with a message like: "';' expected" or "Syntax error on token '}'".

Types of Program Errors

Runtime error

Is known as exceptions, occur during the execution of a Java program when certain unexpected conditions or situations are encountered. These errors cause the program to terminate abruptly if not handled properly.

Runtime Error Example

```
public class SyntaxErrorExample {
    public static void main(String[] args) {
        System.out.println(1/0);
    }
}
```

The reason for the exception is that dividing any number by zero is mathematically undefined, and Java strictly follows this rule. The ArithmeticException is thrown to indicate that an illegal arithmetic operation (division by zero) has occurred.

Types of Program Errors

Logic error

A logic error in Java (or any programming language) refers to a situation where the code compiles and runs without throwing any syntax or runtime errors, but it does not produce the intended or correct results due to flaws in the algorithm or logical flow of the program.

Logic Error Example

```
public class SyntaxErrorExample {
    public static void main(String[] args) {
        System.out.println(”Islamic anniversary”);
    }
}
```

Practice

Write a java program that prints the following pattern

```
* * *
*   *
* * *
```

Practice

Write a java program that prints the following pattern

```
  *
 ***
*****
```

Practice

Write a java program that prints the following pattern

```
   *
  ***
 *****
*******
 *****
  ***
   *
```

Practice

Write a java program that prints the following pattern

```
*
**
***
****
```

Practice

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

Practice

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

Practice

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

Practice

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