Before we start

Outside class help:

- Labs (1 x 2 hrs per week, *at least*)
- Prof office hours
- Student Learning Hub
  - in the Library building
- Supplement Learning (SL Sessions)
  - www.students.ok.ubc.ca/learning-hub/supplemental-learning
- Math & Science Center
  - many TAs to answer your questions

Textbook

- older version is ok
- Notes could be enough

Clickers questions

- Repeat same from last lecture
- Basic rules of Java
- System.out.println

COSC 111
Computer Programming I

Introduction to Computers, Programs, and Java

Dr. Abdallah Mohamed

Acknowledgement: Slides mainly rely on the materials prepared by Y. D. Liang for his textbook "Introduction to Java Programming, 10th Ed., Pearson Edu. Inc.".

Outline

1) Computers: Hardware & Software

2) First Java Program and Java Class Anatomy

3) The Software Development Process

4) Programming Errors

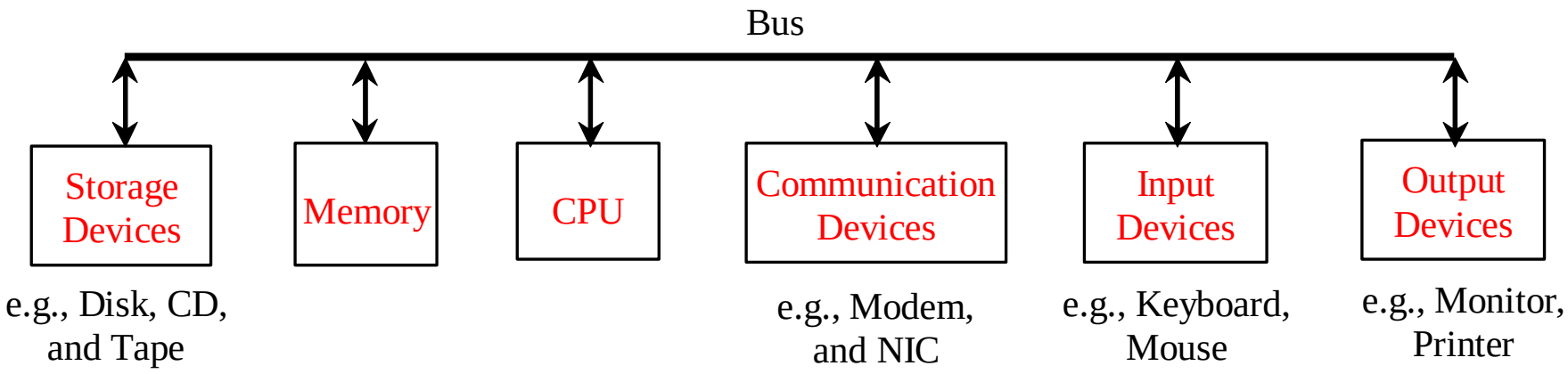
5) More About Java

Computers: Hardware and Software

Computers

A computer consists of

- CPU,
- Memory,
- Input & Output devices,
- ...

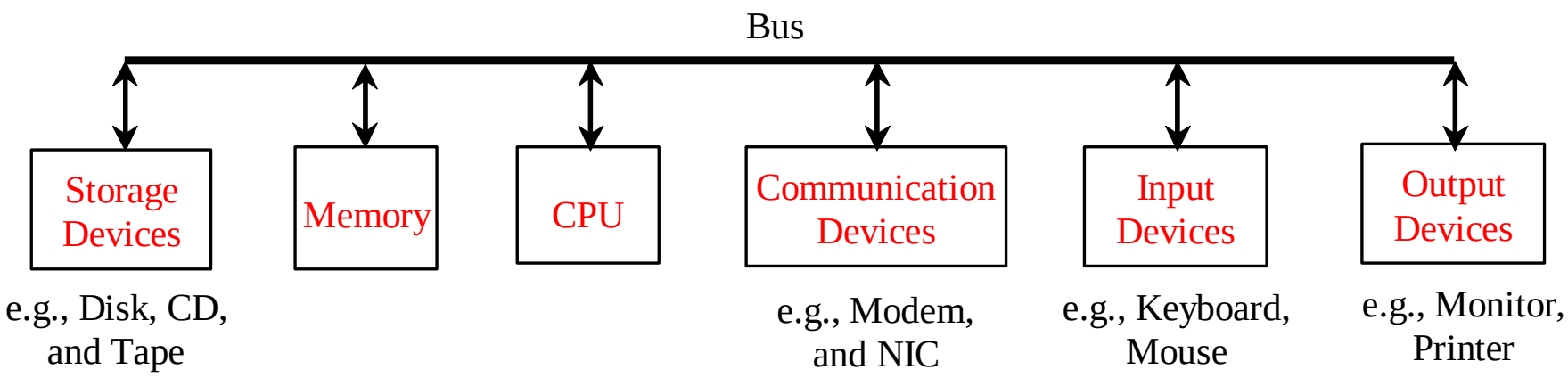


Computers

A computer consists of

- **CPU**,
- Memory,
- Input & Output devices,
- ...

- The Central processing Unit (CPU) is the brain of a computer. It retrieves instructions from memory and executes them.
- The CPU speed is measured in megahertz (MHz), with 1 megahertz equaling 1 million pulses per second.

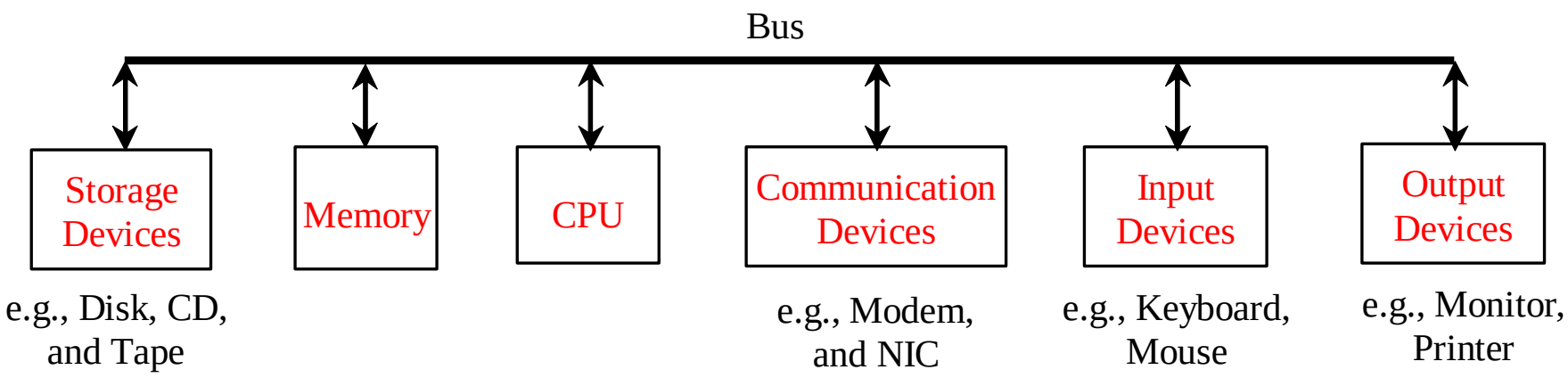


Computers

A computer consists of

- CPU
- **Memory**
- Input & Output devices
- …

- A memory unit is an ordered **sequence of bytes, each holds eight bits.**
- A **program** and its **data** must be brought to memory before they can be executed.
- The current content of a memory byte is lost whenever new information is placed in it.

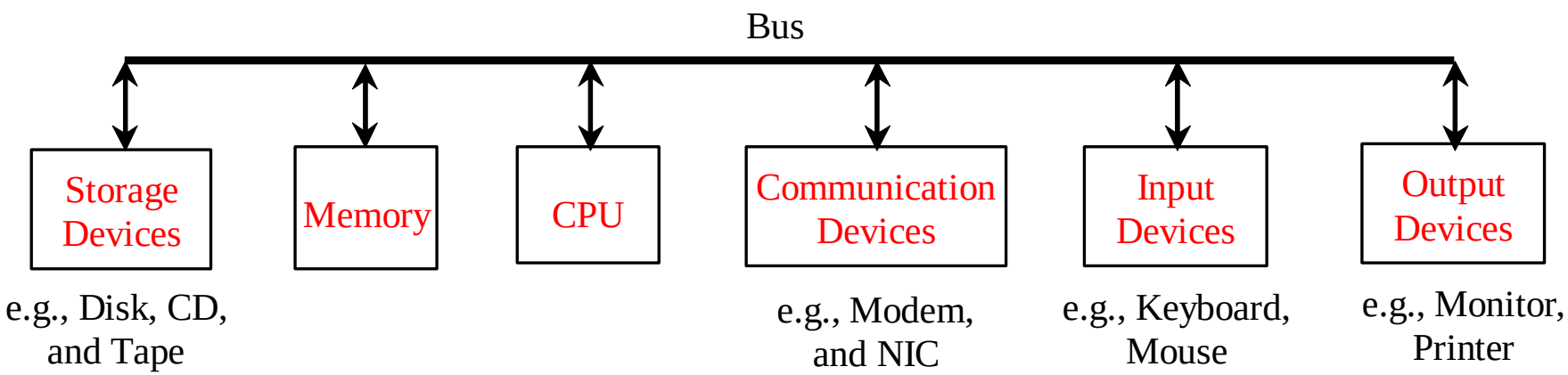


Computers

A computer consists of

- CPU
- Memory
- **Input & Output devices**
- …

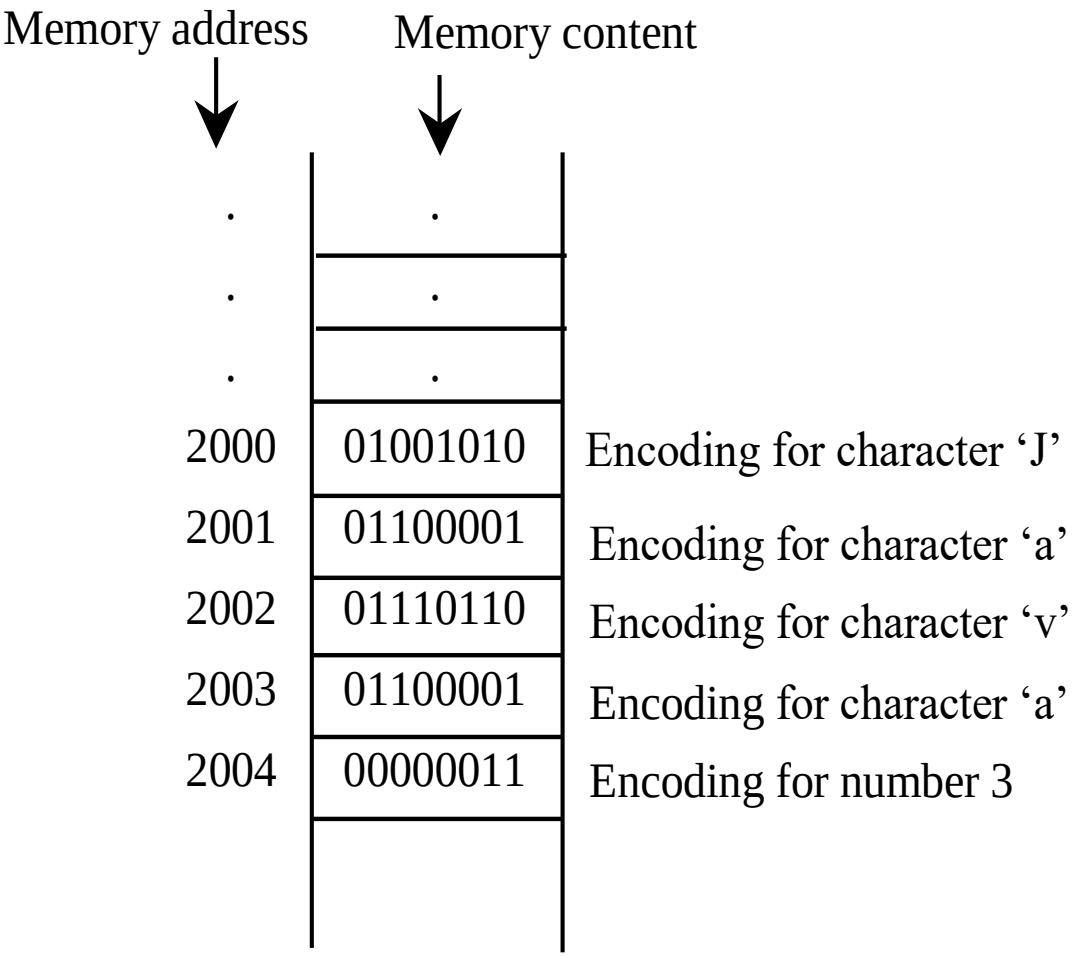
- Input devices: e.g., keyboard, mouse, etc.
- Output devices: e.g., monitor, printer.



How Data is Stored?

Data of various kinds, e.g., numbers, characters, and strings, are encoded as a series of bits (0's and 1's) and stored in the memory.

- If computer needs to store a large number that cannot fit into a single byte, it uses a number of adjacent bytes.
- No two data can share or split a same byte.



Software

Computer programs, known as **software**, are instructions to the computer.

- You tell a computer what to do through programs.
- Without programs, a computer is an empty machine.

Computers **do not understand human languages**, so you need to use computer languages to communicate with them.

Programs are written using **programming languages**.

Programming Languages

Machine Language Assembly Language High-Level Language

Machine language is a set of primitive instructions built into every computer. The instructions are in the form of binary code.

- Programming with native machine language is a tedious process.
- Moreover the programs are highly difficult to read and modify.

For example, to find the sum of two numbers, you might write an instruction in binary like this:

- 1101101010011010

Programming Languages

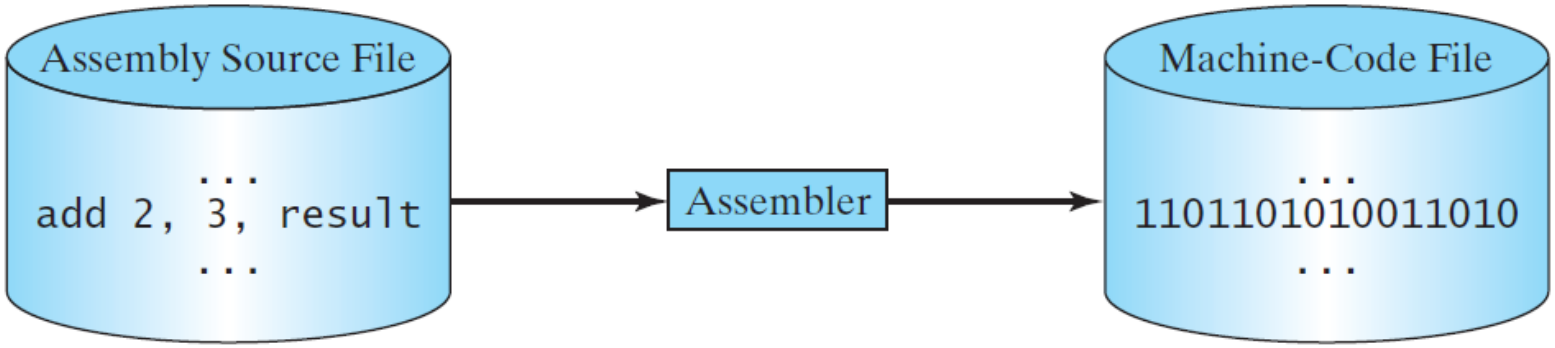
Machine Language Assembly Language High-Level Language

Assembly languages were developed to make programming easy.

Since the computer cannot understand assembly language, a program called **assembler** is used to convert assembly language programs into machine code.

For example, to add two numbers, you might write:

```
ADD R1, R2, R3
```



Programming Languages

Machine Language Assembly Language High-Level Language

The **high-level languages** are English-like and easy to learn and program.

For example, the following is a high-level language statement that computes the area of a circle with radius 5:

```
area = 5 * 5 * 3.1415;
```

Programming Languages

Machine Language Assembly Language High-Level Language

Examples of High-level programming languages

Ada	was developed for the Department of Defense and is used mainly in defense projects.
BASIC	Beginner's All-purpose Symbolic Instruction Code
C	combines the power of an assembly language with the ease of use and portability of a high-level language.
C++	C++ is an object-oriented language, based on C
C#	Pronounced "C Sharp." It is a hybrid of Java and C++ and was developed by Microsoft
FORTRAN	FORmula TRANslation. Popular for scientific and mathematical applications.
Java	Developed by Sun Microsystems, now part of Oracle. It is widely used for developing platform-independent Internet applications.
Visual Basic	Visual Basic was developed by Microsoft and it enables the programmers to rapidly develop graphical user interfaces.

Programming Languages

Machine Language Assembly Language High-Level Language

A program written in a high-level language is called a **source program** or **source code**.

Because a computer cannot understand a source program, a source program must be translated into machine code for execution.

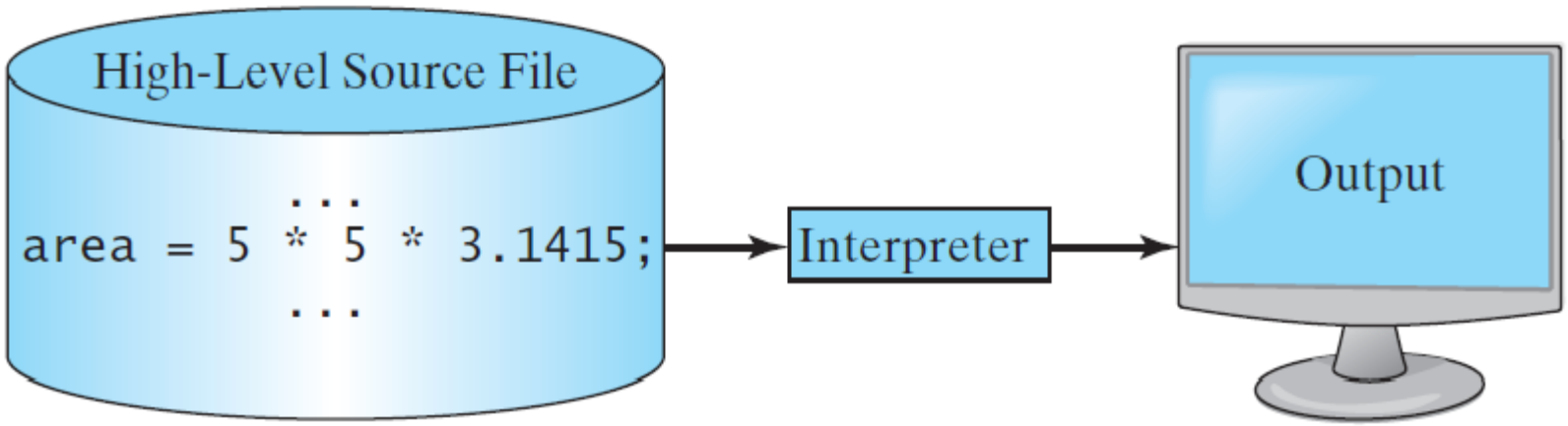
The translation can be done using another programming tool called

- an interpreter or
- a compiler.

Interpreting Source Code

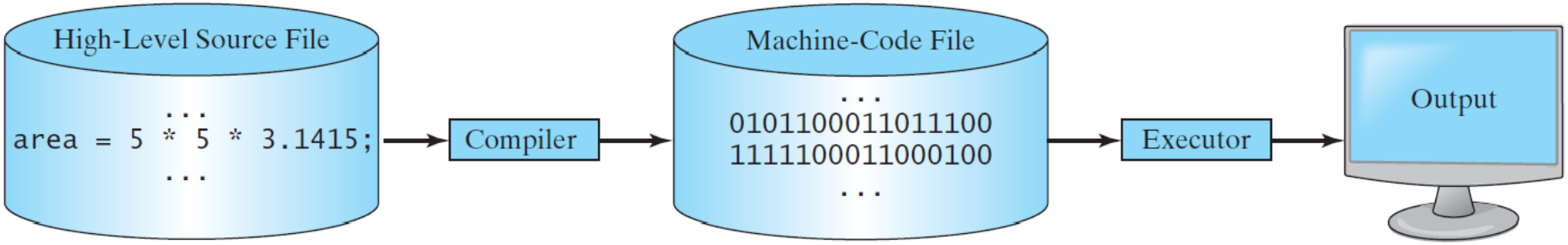
An **interpreter** reads one statement at a time from the source code, translates it to the machine code or virtual machine code, and then executes it right away.

- Note that a statement from the source code may be translated into several machine instructions.



Compiling Source Code

A **compiler** translates the entire source code into a machine-code file, and the machine-code file is then executed, as shown in the following figure.



Clicker Question

Java's source code is written in..

A. machine language

B. assembly language

C. high-level language

D. None of the above

Clicker Question

The part of the computer that is responsible for retrieving and executing instructions is the:

A. Memory

B. CPU

C. Input/output devices

D. Storage devices

E. Keyboard

First Java Program
and
Java Class Anatomy

Simple Java Program

Animation

1. Start at main method

2. Execute statement(s)

3. End program

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

Output:

Welcome to Java

Simple Java Program 2

Animation

```java
// This program prints Welcome to Java!
public class Welcome {
    public static void main(String[] args) {
        System.out.println("Programming is fun!");
        System.out.println("Fundamentals First");
        System.out.println("Problem Driven");
    }
}
```

<u>Output:</u>
Programming is fun!
Fundamentals First
Problem Driven

Simple Java Program 3

Animation

```
public class ComputeExpression {
    public static void main(String[] args) {
        System.out.println((10.5 + 2 * 3) / (45 - 3.5));
    }
}
```

Output:
0.39759036144578314

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- 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 Welcome.

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

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- Line 2 defines the main method. In order to run a class, the class must contain a method named main. The program is executed from the main method.

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

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- A statement represents an action or a sequence of actions. The statement System.out.println("Welcome to Java!") in the program in Listing 1.1 is a statement to display the greeting "Welcome to Java!“.

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

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

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

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

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

- Reserved words or keywords are words that have a specific meaning to the compiler and cannot be used for other purposes in the program. For example, when the compiler sees the word class, it understands that the word after class is the name for the class.

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

Anatomy of a Java Program

Class name

Main method

Statements

Statement terminator

Reserved words

Comments

Blocks

Used to document programs and improve their readability. Two types:

- End-of-line comment: it terminates at the end of the line on which it appears.

  // This is a comment!

- Block comment, can be spread over several lines

```
/* This comment. is split over
   multiple lines
*/
```

Compiler ignores comments.

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

Anatomy of a Java Program

Class name
Main method
Statements
Statement terminator
Reserved words
Comments
Blocks

- A pair of braces in a program forms a block that groups components of a program.

Class block
Method block

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

Special Symbols

Character	Name	Description
{}	Opening and closing braces	Denotes a block to enclose statements.
()	Opening and closing parentheses	Used with methods.
[]	Opening and closing brackets	Denotes an array.
//	Double slashes	Precedes a comment line.
" "	Opening and closing quotation marks	Enclosing a string (i.e., sequence of characters).
;	Semicolon	Marks the end of a statement.

Special Symbols

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

Guidelines

Naming Conventions

- Choose meaningful and descriptive names.
- Class names:
  - Capitalize the first letter of each word in the name. For example, the class name ComputeExpression.

User Proper Indentation and Spacing

- Indentation: Indent two spaces.
- Spacing: Use blank line to separate segments of the code.

Some rules

To program in Java you must follow a set of rules for specifying your commands. This set of rules is called a ***syntax***.

Important general rules of Java syntax:

- Java is ***case-sensitive***.
  - `Main()` is not the same as `main()` or `MAIN()`.
- Java accepts ***free-form layout***.
  - Spaces and line breaks are not important except to separate words.
  - You can have as many words as you want on each line or spread them across multiple lines.
  - However, you should be consistent and follow the programming guidelines given for assignments.
    - It will be easier for you to program and easier for the marker to mark.

More about System.out.println

System.out.println is a Java statement used to print the argument passed to it.

Possible arguments include numbers and strings

If an expression is used, the expression is evaluated first then the result is printed.

```
System.out.println("Hi");
System.out.println("Hi " + "There");
System.out.println(3 + 6);
System.out.println("High " + 5);
System.out.println(2 + 5 + "A");
System.out.println("A" + (2 + 5));
System.out.println("A" + 2 + 5);
```

Output

```
Hi
Hi There
9
High 5
7A
A7
A25
```

More about System.out.println

When we have a mathematical expression that involves (+,-,*,/), multiplication (*) and division(/) have ***higher precedence*** (i.e. computed first) than addition(+) and subtraction(-).

- e.g. 3 + 4 * 2 is the same as 3 + (4 * 2) and equal to 11

If an expression has two operators with the same precedence, then for the above mathematical operators they are evaluated from left to right..

- e.g. 3 + 4 - 2 is the same as (3 + 4) - 2 which becomes 7 - 2

```
System.out.println(2 + 5 * 3 - 1);
System.out.println("A" + (5 - 2) );
System.out.println("A" + 2 - 5);
```

Output

```
16
A3
Error (Why?)
```

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3 + 4 is: ");
      System.out.println(3+4);
   }
}
```

A. 7 is:
   7

B. 3 + 4 is:
   3+4

C. 7 is:
   3+4

D. 3 + 4 is: 7

E. 3 + 4 is:
   7

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3 + 4 is " + 3 + 4);
   }
}
```

A. 7 is 7

B. 3 + 4 is 3 + 4

C. 3 + 4 is 7

D. 3 + 4 is 34

E. None of the above

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3 + 4 is " + (3 + 4));
   }
}
```

A. 7 is 7

B. 3 + 4 is 3 + 4

C. 3 + 4 is 7

D. 3 + 4 is 34

E. None of the above

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("3" + 5);
   }
}
```

A. 8

B. 35

C. Error

D. None of the above

Clicker Question

What is the output of the following program

```
public class Q{
   public static void main(String[] args) {
      System.out.println("Result is " + 5 * 3);
   }
}
```

A. Result is 53

B. Result is 15

C. Result is 5 Result is 5 Result is 5

D. Error

E. None of the above

Eclipse IDE

Eclipse

It is possible to write Java programs using any text editor and compile them using the Java compiler.

An ***integrated development environment (IDE)*** makes it easier to write code, find errors, and run your programs.

We will use the ***Eclipse*** environment in this course.

- Eclipse is a generic, extensible development environment that can be used for Java and other languages.
- Eclipse makes coding easier with automatic error checking, code completion, and source debugging.
- Eclipse will **NOT** make it easier to figure out **WHAT** to write, but it will make **HOW** to write it easier.

Eclipse Initial Setup
Creating a Workspace and a Project

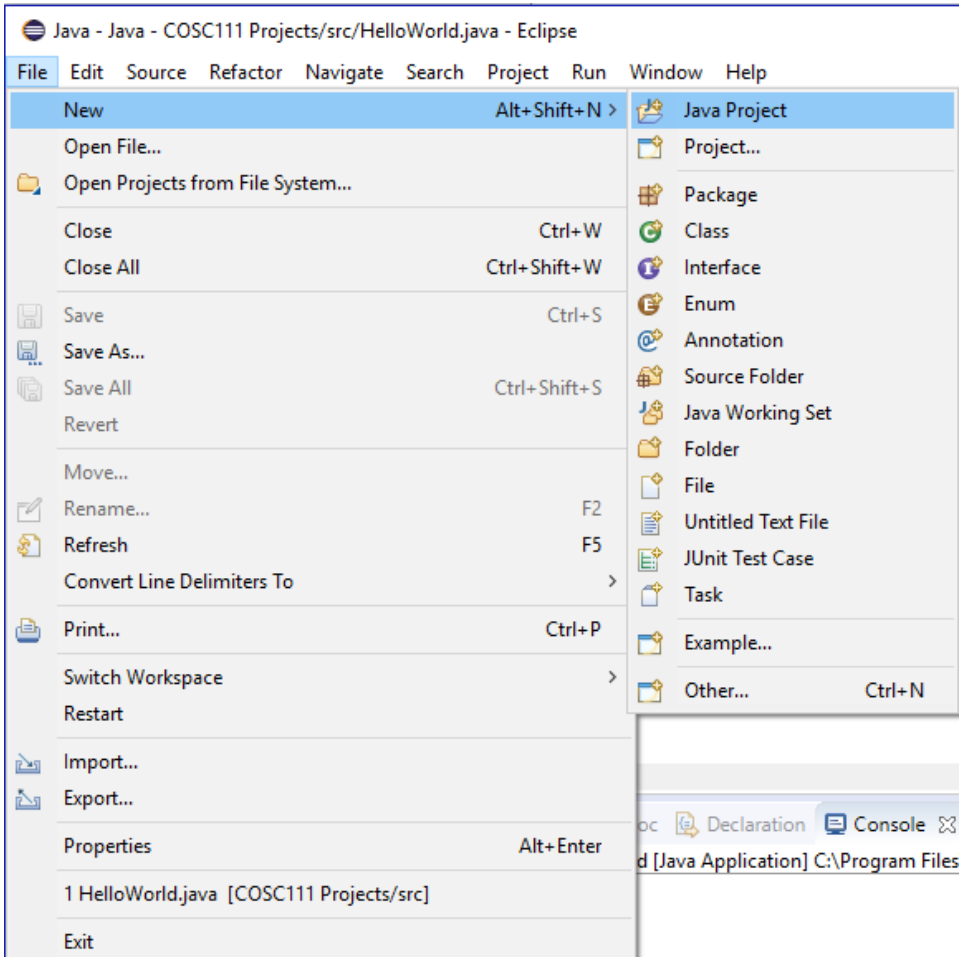
A ***workspace*** is the place where Eclipse will store all of your projects.

- You will be prompted for your workspace on start up if you have not selected one.

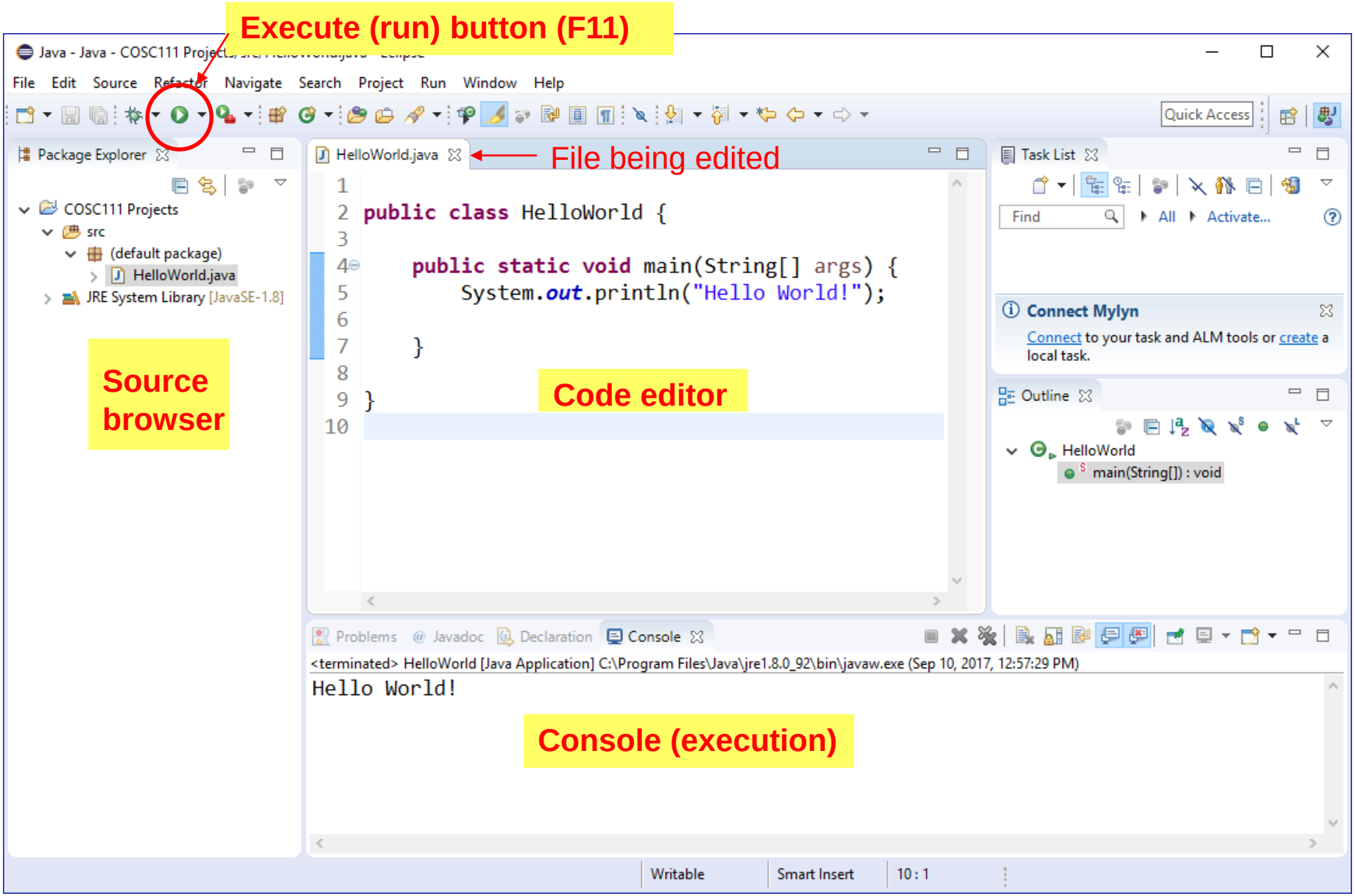
Create a new workspace on **`F:`** with a directory name **`workspace`**.

A ***project*** is a group of program files for some purpose. We will create a sample project called **`cosc111`**. You will also create projects for each assignment.

- Give the project a name and click finish. Ignore all options for now.



Eclipse Main Screen



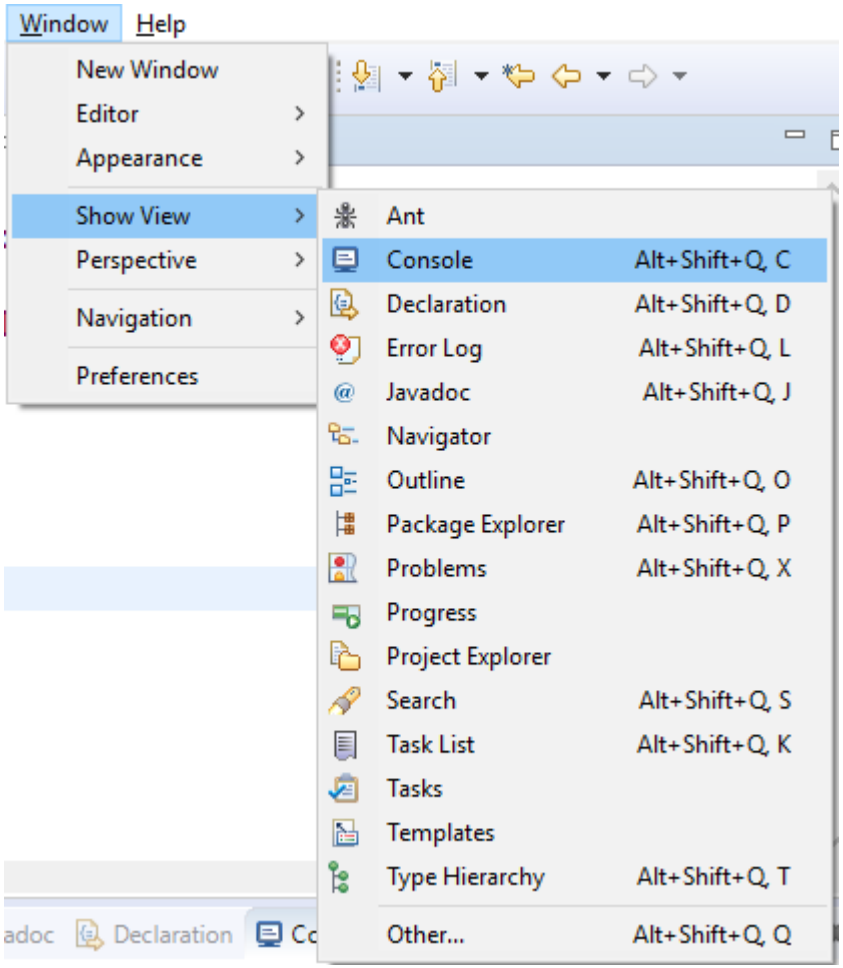
Eclipse
Perspectives and Views

A ***view*** is a window on the screen associated with a task.

- The major views are:
  - Navigator – shows files in project
  - Console – shows program output
  - Problems – shows errors in code
- You may open, close, and organize views in each perspective.

A ***perspective*** is an organization of views to accomplish a certain task (debugging, coding, etc.).

- The two perspectives we will use are Java and Debugging.
- Eclipse remembers how you place the views in each perspective.



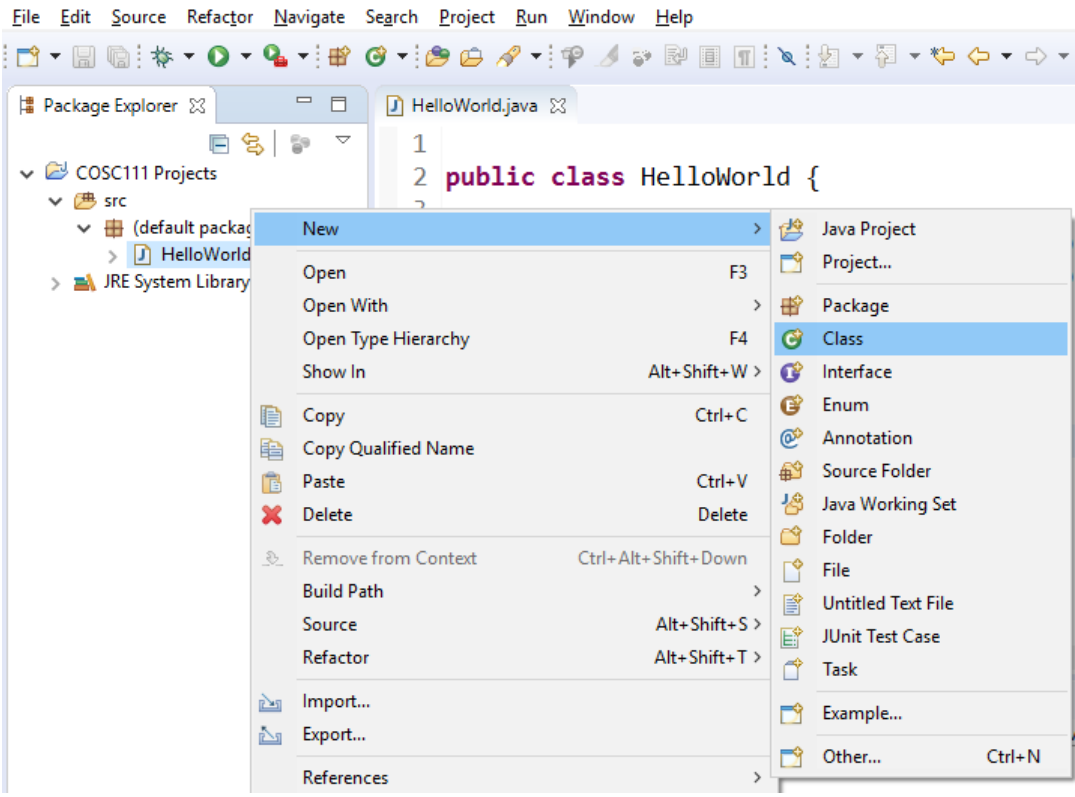
Eclipse
Creating a Program File

To create a program code file, right click on a folder in the navigation view and select **`New->File`**.

The other choice is to select **`File->New->File`** or **`File->New->Class`** and provide a folder and file name.

Type the file name (should end with **`.java`**) and click **`Finish`**.

To edit this file, double click on it, and it will open in the editor.



Programming Errors

Programming Errors

3 types of errors:

- Syntax Errors
  - Detected by the compiler
  - aka *compilation errors*

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

- Runtime Errors
  - Causes the program to abort during the runtime.

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

Can't divide by zero

- Logic Errors
  - Produces incorrect result during the runtime
  - no error message is shown

```
public class Errors {
    public static void main(String[] args) {
        System.out.println("35 Celsius in Fahrenheit:");
        System.out.println((9 / 5) * 35 + 32);
    }
}
```

Output is incorrect due to wrong formula

Eclipse: Debugging and Breakpoints

Debug button

Step and Play Buttons

Java - Debug - COSC111 Projects/src/HelloWorld.java - Eclipse

File Edit Source Refactor Navigate Search Project Run Window Help

Quick Access

Debug

HelloWorld.main(String[]) line: 5
C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (Sep 10, 2017, 1:08:11 PM)
HelloWorld [Java Application]
HelloWorld at localhost:52212
Thread [main] (Suspended)
HelloWorld.main(String[]) line: 6
C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (Sep 10, 2017, 1:08:20 PM)

(x)= Variables | Breakpoints

Name	Value
args	String[0] (id=16)

Variable view

HelloWorld.java

```
1
2 public class HelloWorld {
3
4     public static void main(String[] args) {
5         System.out.println("Hello World!");
6         System.out.println("Bye");
7     }
8
```

Code editor

Breakpoint

Next statement to execute

Outline

HelloWorld
main(String[]) : void

Console | Tasks

HelloWorld [Java Application] C:\Program Files\Java\jre1.8.0_92\bin\javaw.exe (Sep 10, 2017, 1:08:20 PM)

```
Hello World!
```

Console (execution)

Writable | Smart Insert | 6 : 1

Software Development Process

Software Development Process

Requirements	understand the problem and document in detail what the software system needs to do.
Design	design the system's components and the relationship among them.
Implementation	translate the system design into programs using a programming language like Java.
Testing	Ensure that the code meets the requirements specification and remove any bugs.
Maintenance	Maintenance is concerned with changing and improving the product.

More About Java

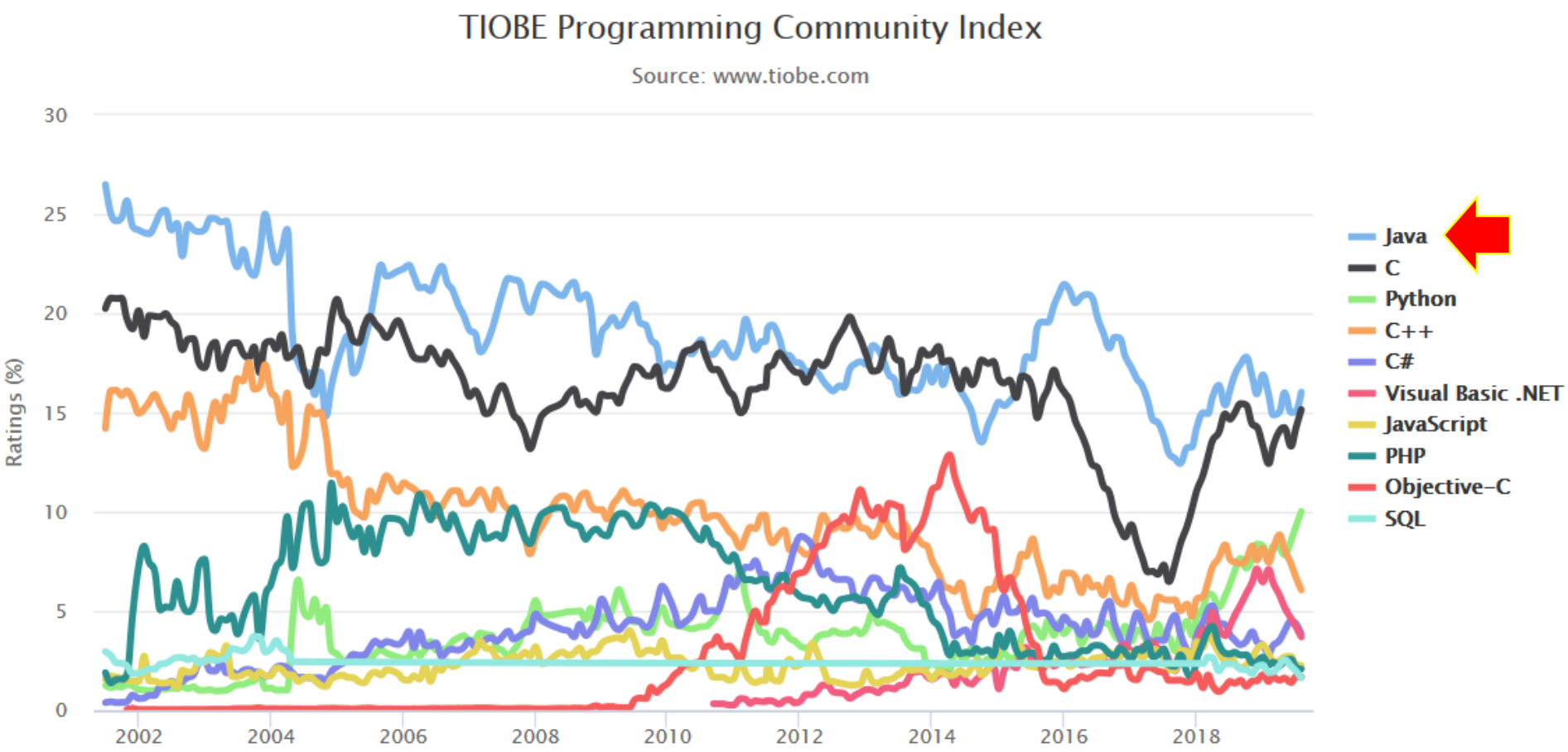
Background

Java is a general-purpose, object-oriented language developed in 1991 by a group led by James Gosling and Patrick Naughton of Sun Microsystems.

Major advantages of Java:

- Can **run** on almost **any type of machine**.
- **Popular** language for web and system development.
- Good teaching language because
  - a) **many issues** such as memory management are **hidden**.
  - b) strict – i.e. doesn't tolerate errors, and hence teaches you to follow the rules.

Most Popular Languages



TIOBE programming community index is a measure of popularity of programming languages

In demand languages for jobs

Based on job posting on Indeed as of January 2019:

Java – 65,986 jobs

Python – 61,818 jobs

Javascript – 38,018 jobs

C++ – 36,798 jobs

C# – 27,521 jobs

PHP – 16,890 jobs

PERL – 13, 727 jobs

Source: https://www.codingdojo.com/blog/the-7-most-in-demand-programming-languages-of-2019

Top Ranked Language for Jobs

IEEE Spectrum, 2017

Choose a Ranking (choose a weighting or make your own)

IEEE Spectrum | Trending | Jobs | Open | Custom

Edit Ranking | Add a Comparison |

Language Types (click to hide)

Web | Mobile | Enterprise | Embedded

Language Rank	Types	Jobs Ranking
1. Java	Web, Mobile, Enterprise	100.0
2. C	Mobile, Enterprise, Embedded	99.4
3. Python	Web, Enterprise	99.3
4. C++	Mobile, Enterprise, Embedded	92.2
5. JavaScript	Web, Mobile	89.9
6. C#	Web, Mobile, Enterprise	86.4
7. PHP	Web	80.5
8. HTML	Web	79.7
9. Ruby	Web, Enterprise	76.6
10. Swift	Mobile, Enterprise	76.4
11. Assembly	Embedded	75.6
12. Shell	Enterprise	73.7
13. Scala	Web, Mobile	70.9
14. R	Enterprise	68.9
15. Perl	Web, Enterprise	65.5
16. SQL	Enterprise	63.0
17. Matlab	Enterprise	61.7
18. Go	Web, Enterprise	[illegible]

<u>How this is computed?</u>

They track the popularity of the programming langagues on Google, Twitter, GitHub, StackOverflow, Reddit, Hacker News, CareerBuilder, IEEE Job Site.

Top Ranked Language for Jobs

IEEE Spectrum, 2019

https://spectrum.ieee.org/static/interactive-the-top-programming-languages-2019

Choose a Ranking

IEEE Spectrum | Trending | Jobs | Open | Custom

Create custom ranking

Language Types

Web | Enterprise | Mobile | Embedded

(Click to hide)

Language Ranking: Jobs

Rank	Language	Type	Score
1	Python		100.0
2	Java		97.5
3	C		96.0
4	C++		85.7
5	JavaScript		83.1
6	C#		77.2
7	HTML,CSS		75.6
8	Swift		69.4
9	Matlab		69.4
10	SQL		69.3

How this is computed?

They track the popularity of the programming langagues on Google, Twitter, GitHub, StackOverflow, Reddit, Hacker News, CareerBuilder, IEEE Job Site.

Java is Portable

Java program run on almost any type of machine

The Java Virtual Machine (JVM)

The ***Java Virtual Machine (JVM)*** is a program that executes a Java program on an individual machine.

After the Java compiler compiles your program:

- your program is in Java *bytecode* which is a set of instructions for the JVM to execute (not the same as machine code)

When you run your program:

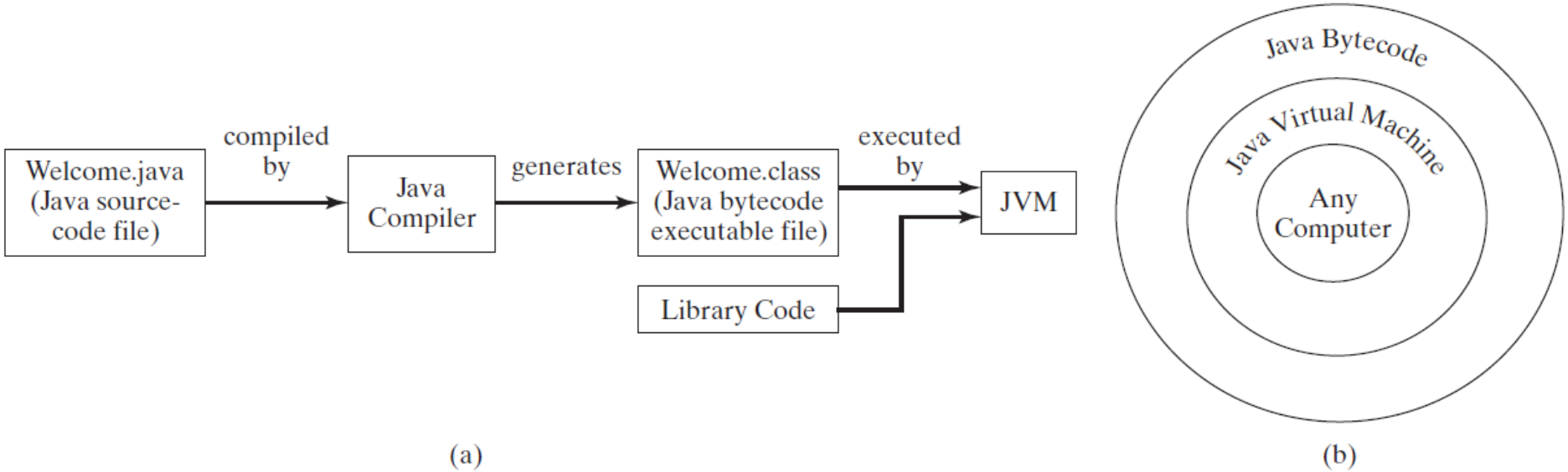
- the JVM is started by the operating system
- the JVM loads your program and begins executing it
- *each byte* in your compiled Java program is either an *instruction* or *data* used by the JVM
- the JVM translates instructions in your program to the appropriate machine code for the machine it is running on

The JVM is effectively a ***virtual machine*** in your computer.

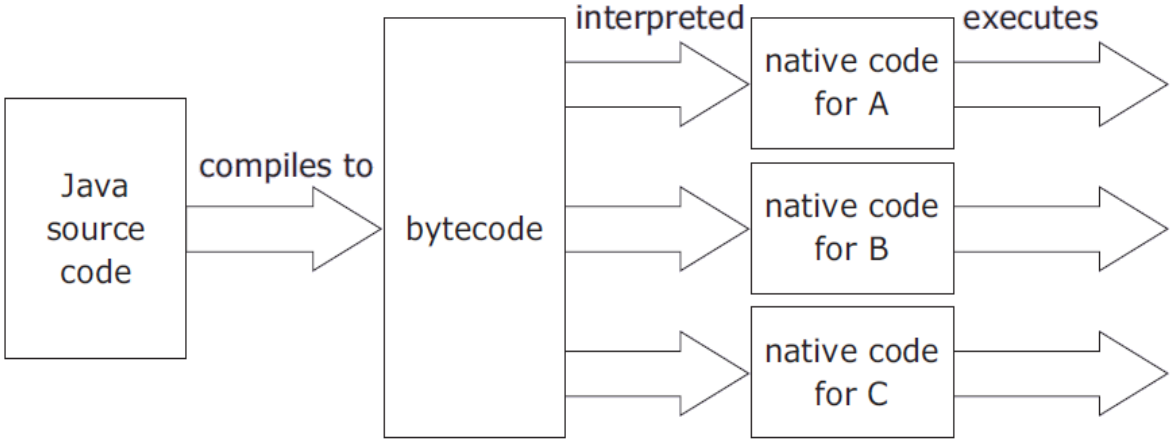
Java Is Portable!

Java was designed to run object programs **on any platform**. With Java,

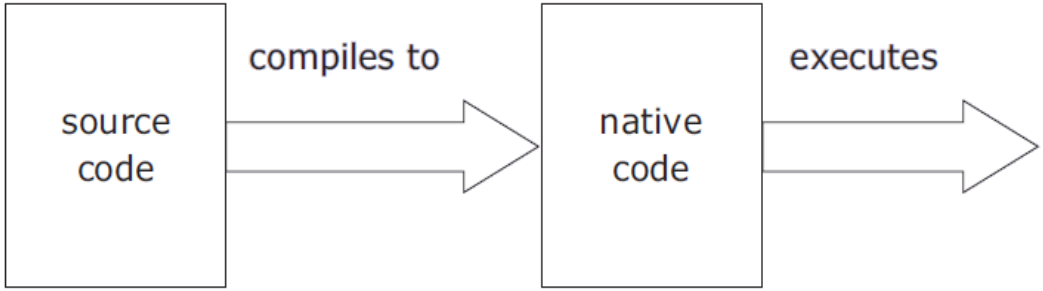
- you write the program once,
- and compile the source program into bytecode.
- The bytecode can then run on any computer with a JVM.



Java Is Portable, cont.



Developing and running ***Java*** programs for different platforms (A, B, and C)



Developing and running programs of some ***other languages*** for a **specific platform**

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

Source: www.cs.armstrong.edu/liang/JavaCharacteristics.pdf

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Java is partially modeled on C++, but greatly simplified and improved.
  - It is like C++ but with more functionality and fewer negative aspects.

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Java was designed from the start to be object-oriented.
  - Object-oriented programming (OOP) is a popular programming approach that is replacing traditional procedural programming techniques.

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Java is designed to make distributed computing easy.
  - Distributed computing involves several computers working together on a network.

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Java 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 Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Java **compilers can detect** many problems that would first show up at execution time in other languages.
- Java has eliminated certain types of error-prone programming constructs found in other languages.
- Java has a runtime **exception-handling** feature to provide programming support for robustness.

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Java implements several security mechanisms to **protect your system against harm** caused by stray programs.

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Write once, run anywhere. With a Java Virtual Machine (JVM), you can write one program that will run on any platform.

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Because Java is architecture neutral, Java programs are portable. They can be run on any platform without being recompiled.

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Java's performance is criticized. Because Java is interpreted, the bytecode is not directly executed by the system, but is run through the interpreter.
  - Techniques are continuously investigated to improve Java's performance

Java Aims

Java Is Simple

Java Is Object-Oriented

Java Is Distributed

Java Is Interpreted

Java Is Robust

Java Is Secure

Java Is Architecture-Neutral

Java Is Portable

Java's Performance

Java Is Multithreaded

- Multithread programming is smoothly integrated in Java.
  - Multithreading is a program's capability to perform several tasks simultaneously

JDK Versions

JDK 1.02 (1995)

JDK 1.1 (1996)

JDK 1.2 (1998) – (aka Java 2)

JDK 1.3 (2000)

JDK 1.4 (2002)

Java 5 (2004) -- (aka JDK 5)

Java 6 (2006) -- (aka JDK 6)

Java 7 (2011) -- (aka JDK 7)

Java 8 (2014) -- (aka JDK 8)

Java 9 (2017)

Java 10 (2018)

Java 11 (2018)

Java 12 (2019)

JDK Editions

Java Standard Edition (J2SE) → ***focus of the course***

- J2SE can be used to develop client-side standalone applications or applets.

Java Enterprise Edition (J2EE)

- J2EE can be used to develop server-side applications such as Java servlets, Java ServerPages, and Java ServerFaces.

Java Micro Edition (J2ME).

- J2ME can be used to develop applications for mobile devices such as cell phones.

Developing Java Programs

To write Java programs, you need to install:

1. JDK (Java Development Kit)
2. IDE (Integrated Development Environment) – we'll use eclipse

What is JDK?

- JDK contains the tools that allow you to develop and run Java programs.