INTRODUCTION

CSC111

DR. SULTAN ALFARHOOD

sultanf@ksu.edu.sa

SYLLABUS

- Textbook
  - Java: An Introduction to Problem Solving and Programming,7th Edition, W. Savitch, Pearson International (Textbook)
  - Java How to Program, 7ed, Deitel and Deitel, Pearson International (Reference)
  - Introduction to Java Programming, Comprehensive Version, 107th Edition, Y. Daniel Liang, Prentice Hall (Reference)

CONTENT

- Chapter 1: Introduction to Computers and Java
- Chapter 2: Basic Computation
- Chapter 3: Flow of Control: Branching
- Chapter 4: Flow of Control: Loops
- Chapter 5: Defining Classes and Methods
- Chapter 6: More About Objects and Methods
- Chapter 7: Arrays

ASSESSMENT METHODS & POLICY

Homework, Quizzes and Attendance	10% 5%	Homework Assignments Class Project
Lab	25%	2 Evaluation Exams in the Lab (10+15)
Written Midterm Exams	20%	10% Midterm exam 1 10% Midterm exam 2
Written final exam	40%	

HOMEWORK ASSIGNMENTS

- Homework will be assigned and graded.
- All homework assignments will be given with a strict deadline, and students are required to submit the assignments on or before the deadline.
- Cheating will not be tolerated.
- All homework assignments or project documents should be submitted using MS-Word and/or an appropriate computer software.
- No hand-written submission will be accepted.

FAQ

- How many lectures have I missed?
- Where can I find the slides?
- Can I prepare for the exam by studying only the slides?

TIPS

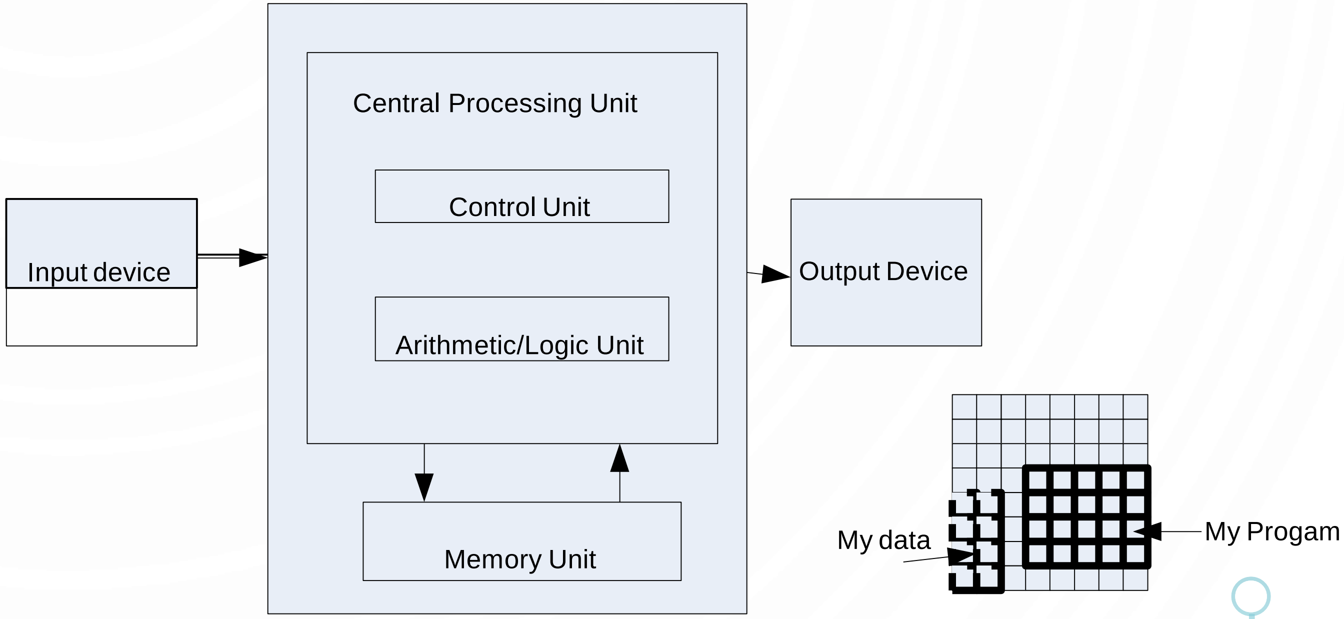
- Practice coding!
- Study missed topics

CHAPTER OUTLINE

- What a computer is
- What a computer program is
- The Programmer’s Algorithm
- How a program that you write in Java is changed into a form that your computer can understand
- Characteristics of Java

WHAT IS A COMPUTER?

- Computer
    - Executes statements (computations/logical decisions)
- Hardware includes the tangible parts of computer systems.
- Software includes programs - sets of instructions for the computer to follow.



COMPUTER ORGANIZATION

- Logical units of a computer system
    - Input unit (Mouse, keyboard)
    - Output unit (Printer, monitor, audio speakers)
    - Central processing unit (CPU) which consists of:
        - Control unit (Supervises operation of other devices)
        - Arithmetic and logic unit (ALU) (Performs calculations)
    - Memory unit (Retains input and processed information)
    - Secondary/auxiliary storage unit (Hard drives, floppy drives)

THE PROCESSOR

- Also called the *CPU* (central processing unit) or the *chip* (e.g. Pentium processor)
- The processor **processes** a program’s instructions.
- It can process only very simple instructions.
- The power of computing comes from speed and program intricacy.

MEMORY

- Memory holds
  - programs
  - data for the computer to process
  - the results of intermediate processing.
- Two kinds of memory
  - main memory
  - auxiliary memory

MAIN MEMORY

- Working memory used to store
  - The current program
  - The data the program is using
  - The results of intermediate calculations
- Usually measured in megabytes (e.g. 8 gigabytes of RAM)
  - RAM is short for *random access memory*
  - A *byte* is a quantity of memory

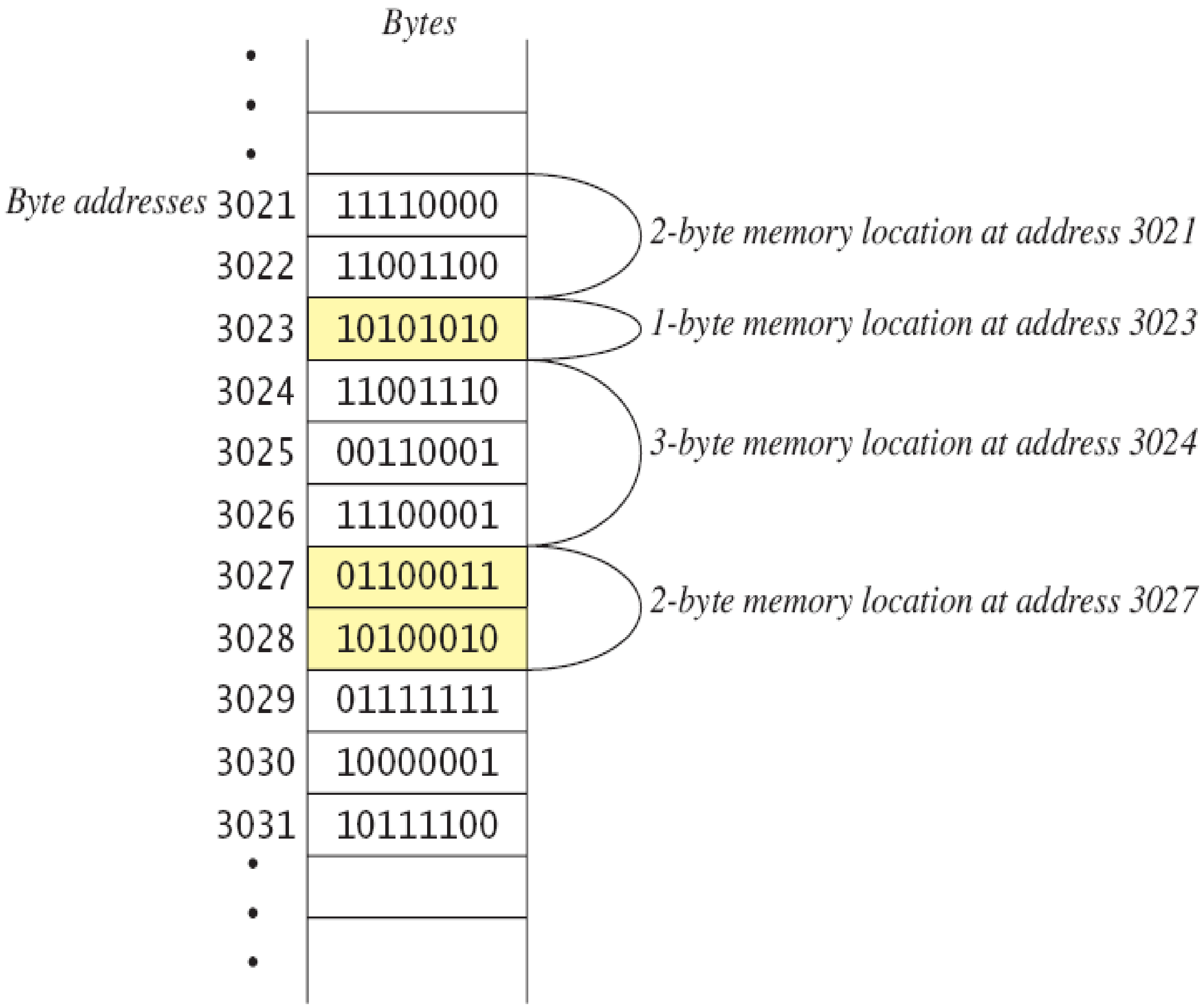
AUXILIARY MEMORY

- Also called *secondary memory*
- Disk drives, CDs, DVDs, flash drives, etc.
- More or less permanent (nonvolatile)
- Usually measured in gigabytes (e.g. 50 gigabyte hard drive)

BITS, BYTES, AND ADDRESSES

- A *bit* is a digit with a value of either 0 or 1.
- A *byte* consists of 8 bits.
- Each byte in main memory resides at a numbered location called its *address.*

MAIN MEMORY



Bytes

Byte addresses	Bytes
3021	11110000
3022	11001100
3023	10101010
3024	11001110
3025	00110001
3026	11100001
3027	01100011
3028	10100010
3029	01111111
3030	10000001
3031	10111100

2-byte memory location at address 3021

1-byte memory location at address 3023

3-byte memory location at address 3024

2-byte memory location at address 3027

0S AND 1S

- Machines with only 2 stable states are easy to make, but programming using only 0s and 1s is difficult.
- Fortunately, the conversion of numbers, letters, strings of characters, audio, video, and programs is done automatically.

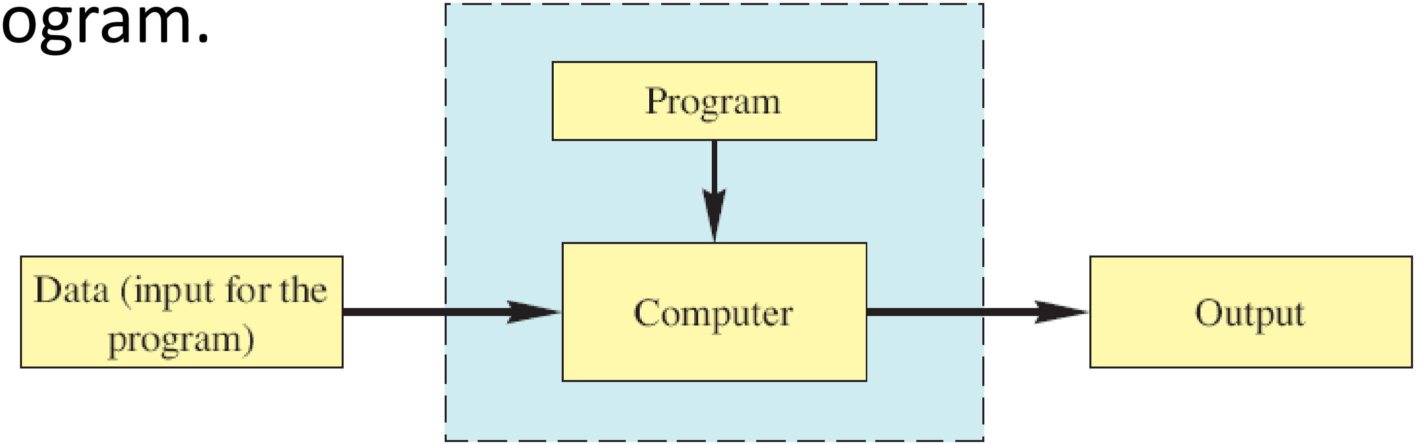
WHAT A COMPUTER PROGRAM IS?

- For a computer to be able to perform specific tasks (i.e. print what grade a student got on an exam), it must be given instructions to do the task

- The set of instructions that tells the computer to perform specific tasks is known as a computer program

INPUT AND OUTPUT

- Normally, a computer receives two kinds of input:
  - The program
  - The *data* needed by the program.
- The output is the result(s) produced by following the instructions in the program.



LEVELS OF ABSTRACTION

- Human thought
- Pseudo-Natural Language (English, Arabic)
- High Level Programming Language (C, C++, Java, …)
- Machine Code

THE PROGRAMMER’S ALGORITHM

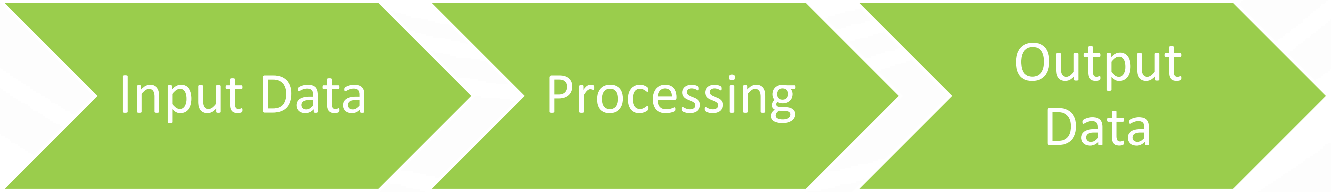
- An **algorithm** is a finite sequence of instructions that produces a solution to a problem.
- The programmer’s algorithm:
  - Define the problem.
  - Plan the problem solution.
  - Code the program.
  - Compile the program.
  - Run the program.
  - Test and debug the program.

THE PROGRAMMER’S ALGORITHM

- **Define the problem.**
- Plan the problem solution.
- Code the program.
- Compile the program.
- Run the program.
- Test and debug the program.

DEFINING THE PROBLEM

- The problem must be defined in terms of:
  - Input: Data to be processed.
  - Output: The expected result.
    - Look for nouns in the problem statement that suggest output and input.
  - Processing: The statements to achieve.
    - Look for verbs to suggest processing steps.



Input Data

Processing

Output Data

- Keyboard
- Screen

INPUT AND OUTPUT

- Inputs
  - Can come from many sources
    - such as users, files, and other programs
  - Can take on many forms
    - such as text, graphics, and sound
- Outputs
  - Can also take on many forms, such as numbers, text, graphics, sounds, or commands to other programs

EXAMPLE: AREA AND PERIMETER OF A RECTANGLE

- Input
  - Length
  - width
- Processing
  - Area = length × width
  - Perimeter = 2 × (length + width)
- Output
  - Area
  - Perimeter

EXAMPLE: SUM AND AVERAGE OF 5 NUMBERS

- Input
  - five number x1, x2, x3, x4, x5
- Processing
  - Sum = x1+x2+x3+x4+x5
  - Average = Sum/5
- Output
  - Sum
  - Average

EXAMPLE: AREA AND PERIMETER OF A CIRCLE

- Input
  - Radius
  - PI
- Processing
  - Area = PI × Radius × Radius
  - Perimeter = 2 × PI × Radius
- Output
  - Area
  - Perimeter

THE PROGRAMMER’S ALGORITHM

- Define the problem.
- **Plan the problem solution.**
- Code the program.
- Compile the program.
- Run the program.
- Test and debug the program.

PLANNING THE SOLUTION

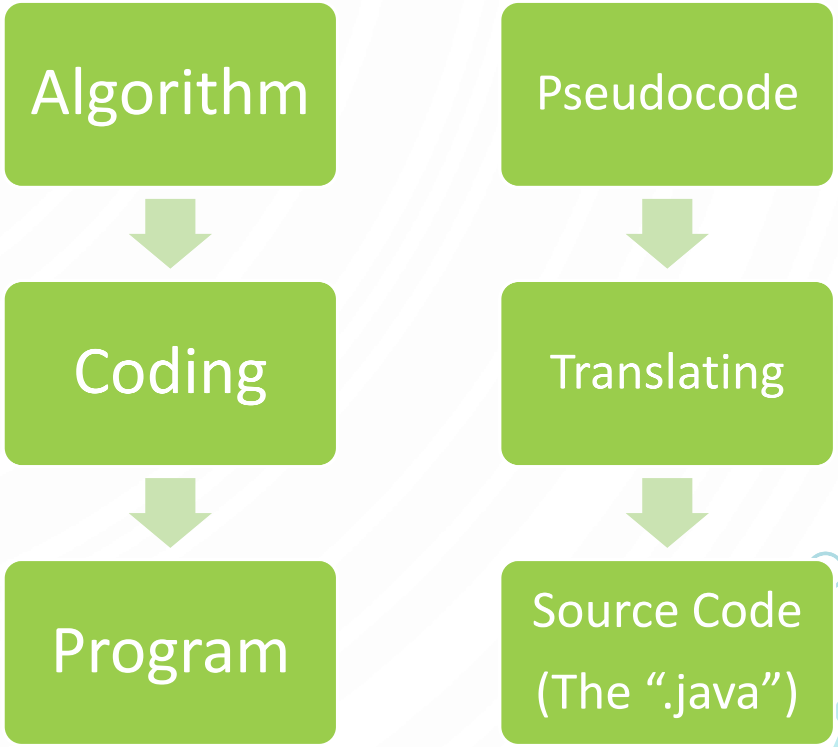
- When planning, algorithms are used to outline the solution steps using Englishlike statements, called **pseudocode**.

THE PROGRAMMER'S ALGORITHM

- Define the problem.
- Plan the problem solution.
- **Code the program.**
- Compile the program.
- Run the program.
- Test and debug the program.

CODING THE PROGRAM

- Coding is writing the program in a formal language called Programming Language.
    - **Programming Language**: A set of rules, symbols and special words used to write statements.
- The program is written by translating the algorithm steps into a programming language statements.
- The written program is called **Source code** and it is saved in a file with “.java” extension.



Algorithm

Coding

Program

Pseudocode

Translating

Source Code (The “.java”)

WHY CODING IN PROGRAMMING LANGUAGES

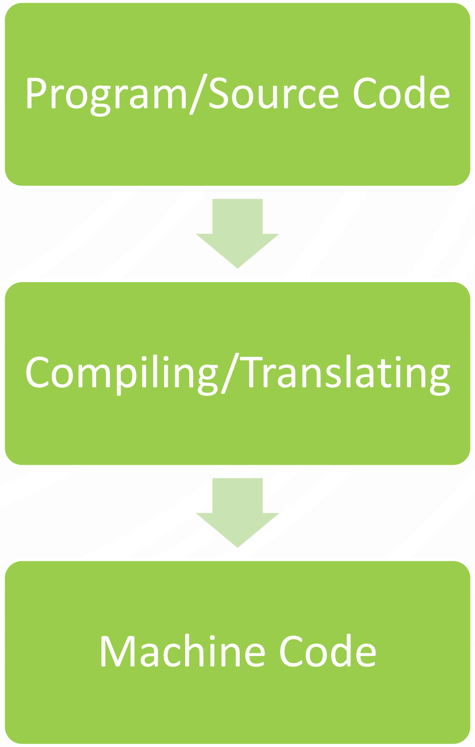
- We write computer programs (i.e. a set of instructions) in programming languages such as C, C++, and Java.
- We use these programming languages because they are easily understood by humans
- Unfortunately, computer hardware does not understand high-level languages.
  - Therefore, a high-level language program must be translated into a low-level language.

THE PROGRAMMER'S ALGORITHM

- Define the problem.
- Plan the problem solution.
- Code the program.
- **Compile the program.**
- Run the program.
- Test and debug the program.

COMPILING COMPUTER PROGRAMS

- Computers do not understand programs written in programming languages such as C++ and Java
- Programs must first be converted into machine code that the computer can run
- A Software that translates a programming language statements into machine code is called a **compiler**


Program/Source Code

Compiling/Translating

Machine Code

PROGRAMMING LANGUAGE COMPILER

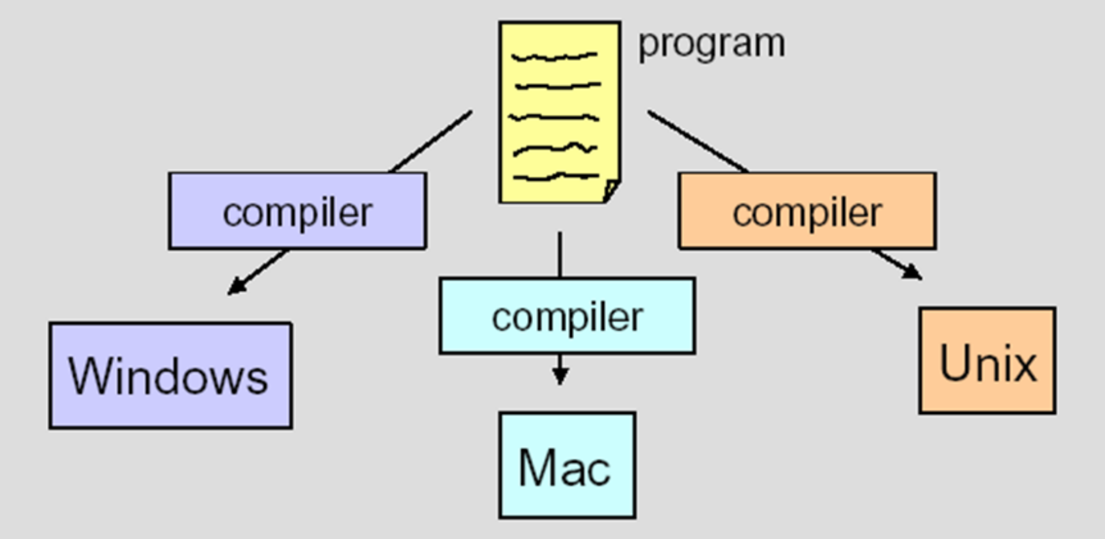
- A **compiler** is a software that:
  - **Checks** the correctness of the source code according to the language rules.
    - Syntax errors are raised if some rules were violated.
  - **Translates** the source code into a machine code if no errors were found.

COMPILERS

- Most high-level languages need a different compiler for each type of computer and for each operating system.
- Most compilers are very large programs that are expensive to produce.

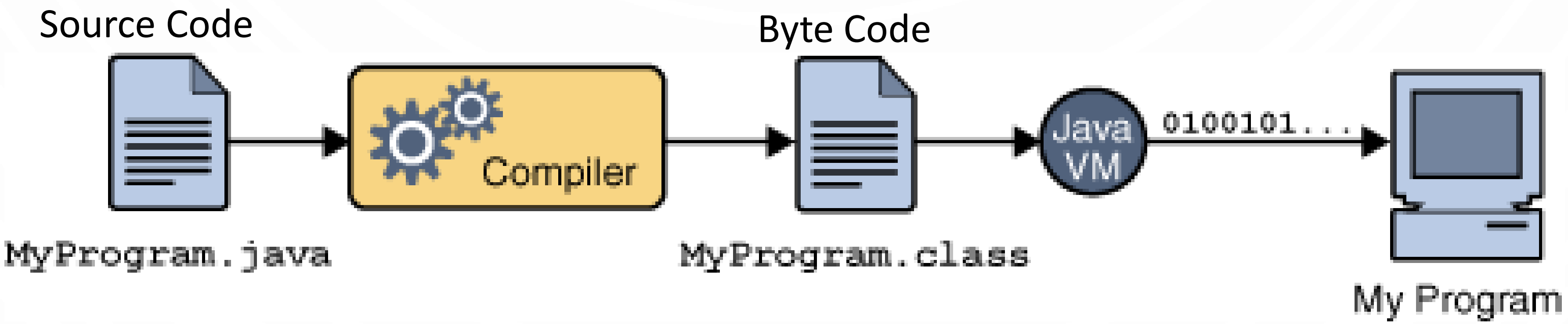
PLATFORM DEPENDENT COMPILING

- Because different platforms or hardware architectures along with the operating systems (Windows, Macs, Unix), require different machine code, you must compile most programs separately for each platform.



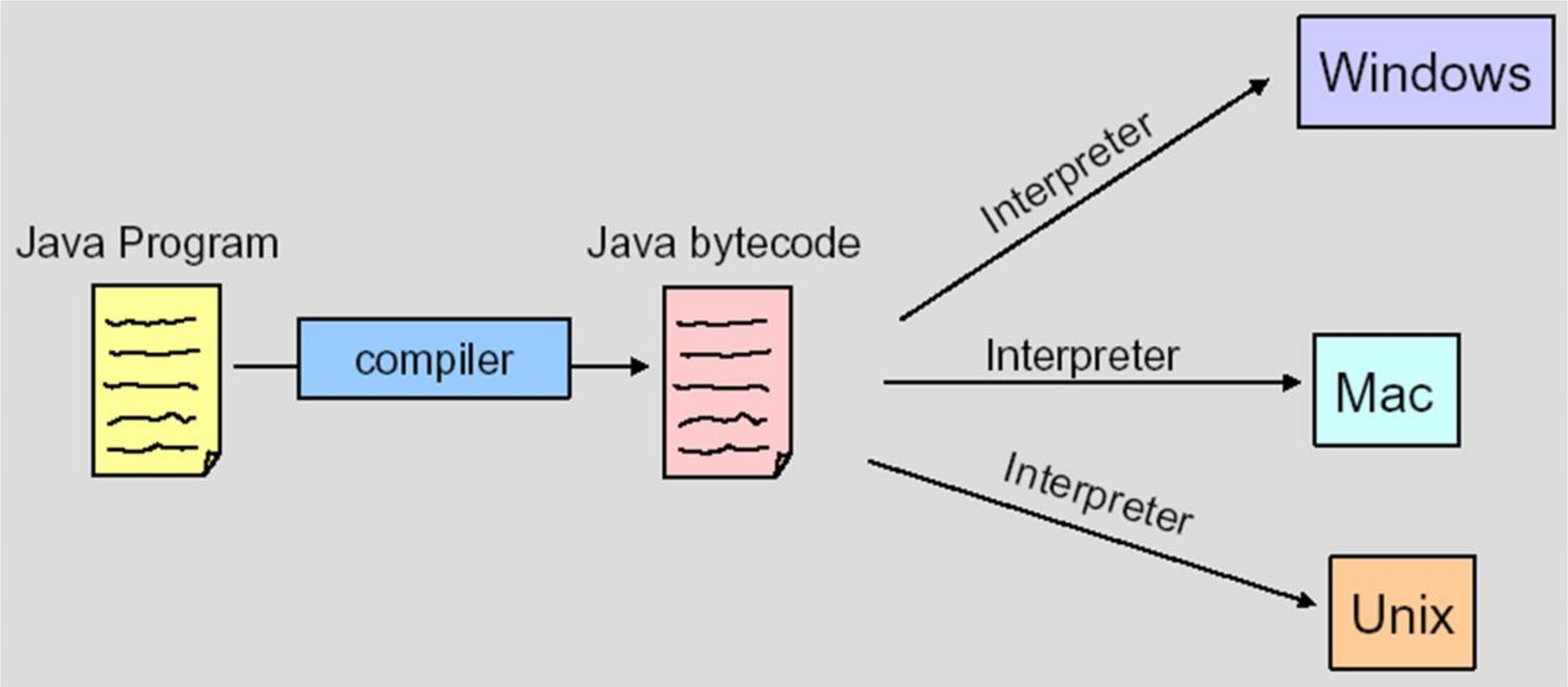
COMPILING JAVA PROGRAMS

- The Java compiler produces bytecode (a “.class” file) not machine code from the source code (the “.java” file).
- Bytecode is converted into machine code using a Java Interpreter

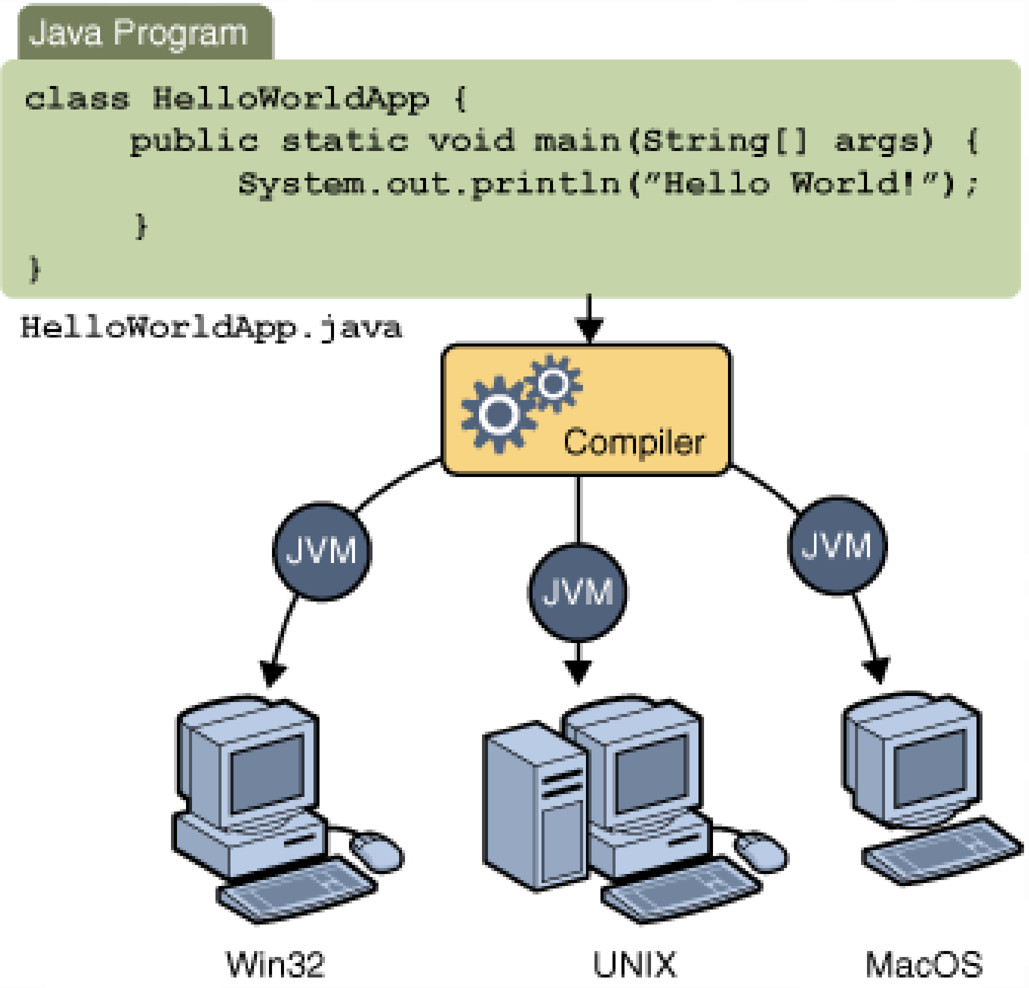


PLATFORM INDEPENDENT JAVA PROGRAMS COMPILING

- You can run bytecode on a computer that has a Java Interpreter installed.



MULTIPURPOSE JAVA COMPILING



Java Program

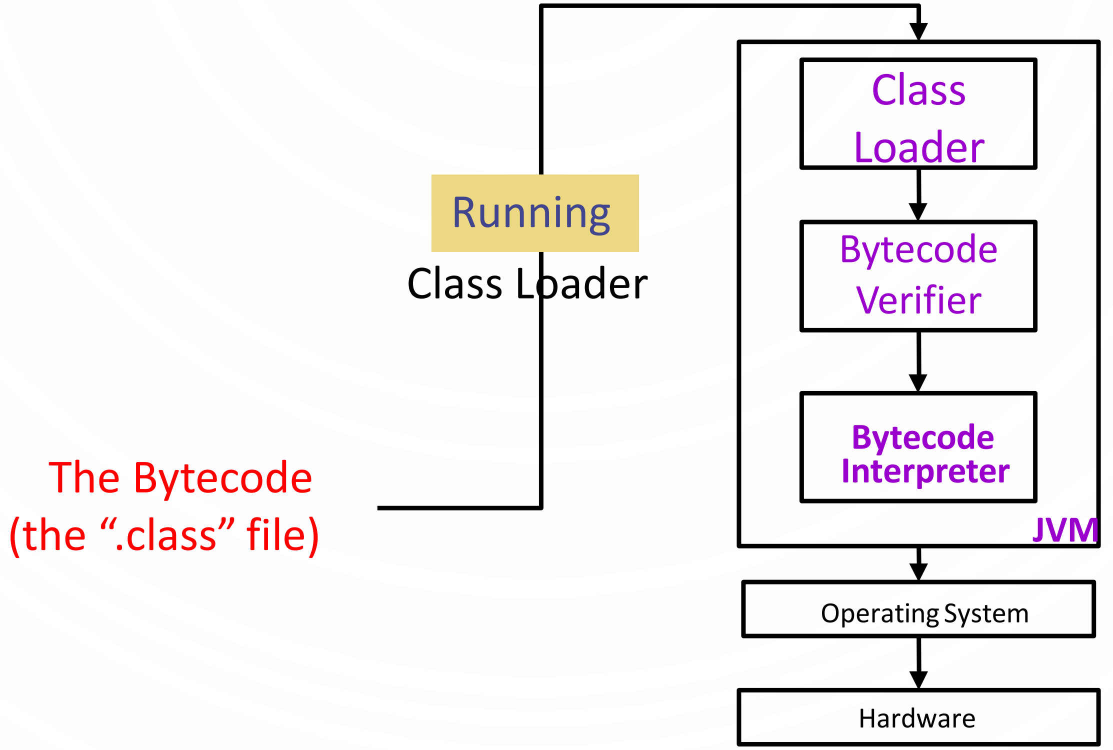
```
class HelloWorldApp {
        public static void main(String[] args) {
                System.out.println("Hello World!");
        }
}
```

HelloWorldApp.java

THE PROGRAMMER’S ALGORITHM

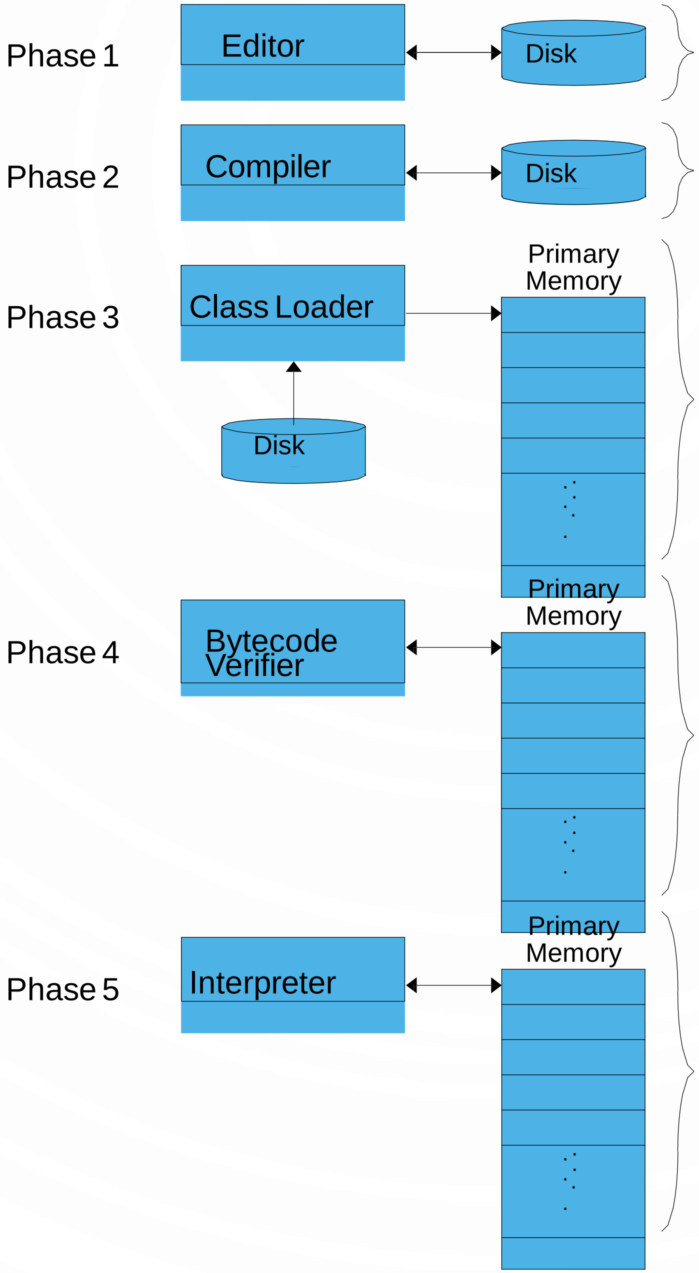
- Define the problem.
- Plan the problem solution.
- Code the program.
- Compile the program.
- **Run the program.**
- Test and debug the program.

RUNNING THE PROGRAM



THE JAVA VIRTUAL MACHINE COMPONENTS

- The Class Loader
  - stores bytecodes in memory
- Bytecode Verifier
  - ensures bytecodes do not violate security requirements
- Bytecode Interpreter
  - translates bytecodes into machine language



Program is created in an editor and stored on disk in a file ending with `.java`.

Compiler creates bytecodes and stores them on disk in a file ending with `.class`.

Class loader reads `.class` files containing bytecodes from disk and puts those bytecodes in memory.

Bytecode verifier confirms that all bytecodes are valid and do not violate Java’s security restrictions.

Interpreter reads bytecodes and translates them into a language that the computer can understand, possibly storing data values as the program executes.

THE JAVA VIRTUAL MACHINE

- The class Loader, the Bytecode Verifier and Interpreter constitute the Java Virtual Machine (JVM).
- JVM is platform specific.
- The interpreter translates the bytecodes into specific machine commands.

THE PROGRAMMER’S ALGORITHM

- Define the problem.
- Plan the problem solution.
- Code the program.
- Compile the program.
- Run the program.
- **Test and debug the program.**

TESTING AND DEBUGGING THE PROGRAM

- Testing
  - Be sure that the output of the program conforms with the input.
  - Types of errors:
    - Logical errors
      - The program run but provides wrong output.
    - Runtime errors
      - The program stop running suddenly when asking the OS executing a non accepted statement (divide by zero, etc).
- Debugging
  - Find, Understand and correct the error

IDE (INTEGRATED DEVELOPMENT ENVIRONMENT)

- *IDE* (integrated development environment) combines a text editor with commands for compiling and running Java programs.
- It helps to
  - write code
  - compile
  - test
  - debug

SOME CHARACTERISTICS OF JAVA

- Object-Oriented
  - Combines data and behavior into one unit -> **objects**
  - Provides data abstraction and encapsulation
  - Decompose program into objects
  - Programs are collections of interacting and cooperating objects
- Platform-independent
  - Portable
  - Architecture neutral
  - ”Write-once, run-anywhere”
- Secure
  - The bytecode verifier of the JVM:
    - checks untrusted bytecode
    - controls the permissions for high level actions