420-101-VA: Programming 1

WEEK 1: COURSE INTRODUCTION

Nagat Drawel - Email: MIO (Omnivox)

Don’t panic!

- You'll gain essential skills in just 2 weeks.
  - You will work hard
  - I'm here to provide support, ensuring your success.
- Ask questions!
  - this is the most effective learning path for you.
- I encourage you to:
  - Actively engage with the instructor's guidance.
  - Develop logical thinking to approach challenges confidently.
  - Solve puzzles independently to enhance your problem-solving abilities.
- Your responsibilities
  - Understand lectures and course material
  - Staying updated by regularly visiting Omnivox and checking your MIO.
  - Uphold Academic Honor Policy
- Get ready to embrace the exciting world of programming, where dedication and collaboration pave the way for your success!

What do we learn in this course?

- You will develop basic skills in solving problems and designing algorithms

- You will learn to work with fundamental programming concepts such as program structure, data types and variables, conditional and looping constructs, objects, methods, and classes.

- You will learn *object-oriented* programming
  —today's key programming methodology.

- You will write and test your programs using an Integrated Development Environment (IDE). Additionally, you will learn how to write clean code

Evaluation

Description	Grade Value	Completion Date
Test 1	**15%**	**Week 6**
Test 2	20%	Week 13
Lab exercises	10%	Week 2-12
Assignments (4 – 5 assignments)	25%	Week 3-12
Project (see, Learning Integration Assessment)	30%	Week 11-15

The exact date of each test will be announced at least one week in advance.

Academic Integrity - Please Don't Cheat!

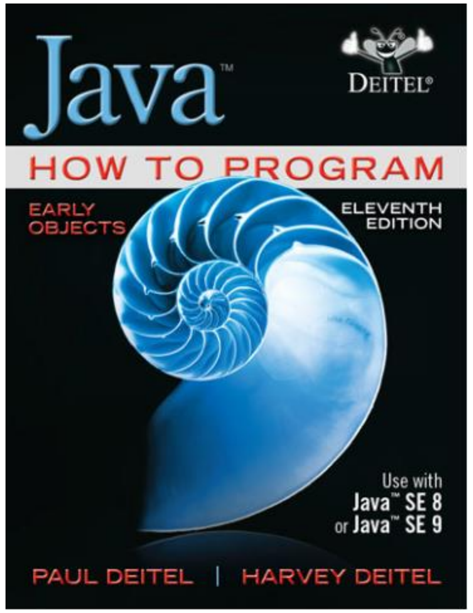
- Please don't help others by giving them solutions!
- Please don't put others in awkward positions by asking them for help on a piece of coursework
- What you can do to help
  - Discuss key concepts and examples from lecture or labs.

Course Materials

- Lecture slides will be published on Omonvox
- Code developed in class, also available on Omonvox
- The textbook is optional for this course. However, the following book is strongly recommended for this course. It may also be used for Programming 2 and Programming Patterns:
- **Title: Java How to Program, Early Objects (Deitel: How to Program) 11th Edition**

  by Paul Deitel (Author), Harvey Deitel (Author),

  Paperback : 1296 pages, ISBN-13: 978-0134743356


Java™
DEITEL®
HOW TO PROGRAM
EARLY OBJECTS
ELEVENTH EDITION
Use with Java™ SE 8 or Java™ SE 9
PAUL DEITEL | HARVEY DEITEL

Let's explore the Course Outlines

- Any Questions?

What is a Computer?

- A computer is basically an electronic device that manipulates information or data.
- It is a machine that can be programmed to carry out sequences of arithmetic or logical operations automatically.
- It has the ability to store, retrieve, and process data.

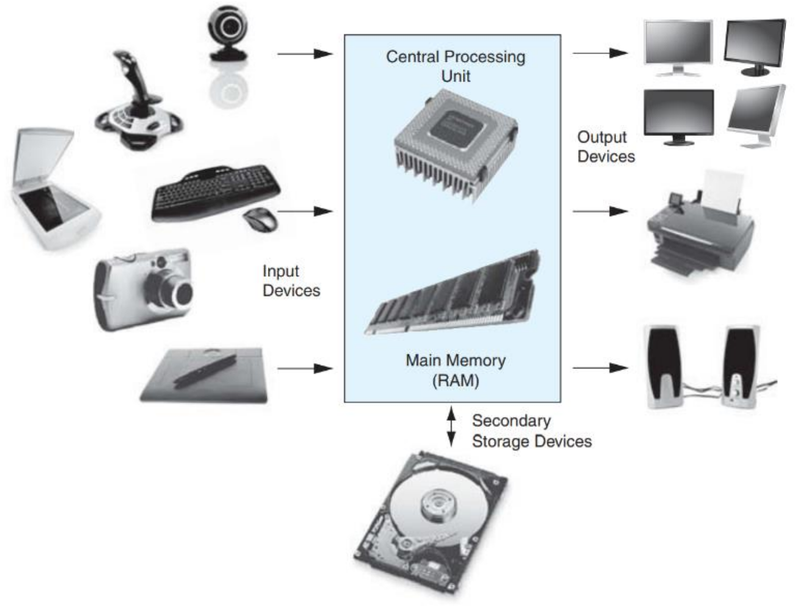
Hardware and Software

- Hardware
  - the physical, tangible parts of a computer
- Software
  - programs and data
  - a program is a series of instructions
- A computer requires both hardware and software
  - Each is essentially useless without the other

Main Components of a Computer

<u>A basic computer consists of:</u>

- CPU - Central Processing Unit: The "brain" of the computer
- ALU - Arithmetic & Logic Unit
  - responsible for performing arithmetic calculations, as well as logical operations.
- Main Memory (RAM - Random Access Memory)
  - stores executing programs and data being currently worked on
- Secondary Memory
  - hard disk, CD, DVD, etc.
- Input devices
  - mouse, keyboard, scanner, USB Flash Drive, etc.
- Output devices
  - screen/console, printer, etc.



Memory refers to the location of short-term data, while storage refers to the location of data stored on a long-term basis.

A Computer Program

- A **computer program** is a set of instructions that can be executed by a computer to perform a specific task.

- Programs usually accept inputs from a source, process them, and output results to a destination

- A computer program is usually written by a programmer in a programming language (special language used to write computer programs).

A Computer Algorithm

- To write a computer program, you have to tell the computer, step by step, exactly what you want it to do. The computer then "executes" the program, following each step mechanically, to accomplish the end goal.

- When you are telling the computer *what* to do, you also get to choose *how* it's going to do it.

- That's where **computer algorithms** come in. The algorithm is the basic technique used to get the job done.

A Computer Algorithm Example

- Let's follow an example to help get an understanding of the algorithm concept.

- **Algorithm to add two numbers entered by the user**

```
Step 1: Start
Step 2: Declare variables num1, num2 and result.
Step 3: Read values num1 and num2.
Step 4: Add num1 and num2. result←num1+num2
Step 5: Display the result
Step 6: Stop
```

A Pseudo Code

- Pseudocode, from pseudo, which means “appearing like”, is a method for writing algorithms informally in plain English which uses short phrases to write code for a program before it is implemented in a specific programming language.
- It is a simpler version of a programming code
- **Example:**

```
ask the user to input a number
read a number from user input
if the number is greater than 0
        display “the number is positive”
else if the number is equal to 0
        display “the number is 0”
else if the number is smaller to 0
        display “the number is negative”
```

Programming Languages

- A programming language employs a set of rules that dictate how the words and symbols can be put together to form valid program statements
- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate *translation steps.*
- These may be divided into three general types:
  - Machine languages
  - Assembly languages
  - High-level languages

Machine Language

- Any computer can directly understand only its own machine language
- Machine language is written using **binary numbers**.
- The binary numbering system (base 2) only has two digits (0 and 1).
- The code that is written with machine language is called machine code or binary code

    - `Example: 1110110101010110001101010` `Say what?`

Assembly Language

- English-like abbreviations that represent elementary operations formed the basis of assembly languages.
- *Translator programs* called assemblers convert early assembly-language programs to machine language.
- Assembly language uses easy-to-remember instructions
  - `Example: ADD $R1, $R2, $R3`
    - `Well, we know it has something to do with addition!`

High level programming languages

- Allow you to write instructions that look almost like English
- Single statements accomplish substantial tasks.
- **Compilers** convert high-level language programs into machine language.

- A payroll program written in a high-level language might contain a *single* statement such as

```
grossPay = basePay + overTimePay
```

- The programming language you will learn is **Java**
- Some other common programming languages:

Java, BASIC, Visual Basic, Python, JavaScript, C, C++, C#, …..

Comparing Among The Three Languages

Machine Code

```
100100 010001
100110 010010
100010 010011
```

Assembly Code

```
LOAD rate
MULT hours
STOR salary
```

Java Code

```
salary = rate * hours;
```

Interpreters and Compilers

- Two kinds of programs translate high-level languages into low-level languages: **interpreters and compilers.**
  - Compiler transforms code written in a high-level programming language into the machine code at once before the program runs, whereas an Interpreter converts each high-level program statement, one by one, into the machine code, during program run.
  - Compiled code runs faster, while interpreted code runs slower.
  - Compiler displays all errors after compilation, on the other hand, the Interpreter displays errors of each line one by one.

Building Programs

- The mechanics of developing a program include several activities:
    - writing the program in a specific programming language (such as Java)
    - translating the program into a form that the computer can execute
    - investigating and fixing various types of errors that can occur
- Software tools can be used to help with all parts of this process

- Any Questions?

WHAT DO YOU THINK ABOUT JAVA?

Java

- Java is one of the world’s most widely used computer programming languages.
- It was introduced by Sun Microsystems, Inc. in 1995 and it's popularity has grown quickly since
- For many organizations, the preferred language for meeting their enterprise programming needs is Java.
- Java is also widely used for implementing Internet-based applications and software for devices that communicate over a network.

Java Program Structure

- In the Java programming language:
    - A program is made up of one or more classes
    - A class contains one or more methods
    - A method contains program statements

- These terms will be explored in detail throughout the course

- A Java application always contains a method called `main`

Java Program Structure

```
  //  comments about the class
       public class MyProgram
       {

//  comments about the method

          public static void main (String[] args)
           {


           }


       }
```

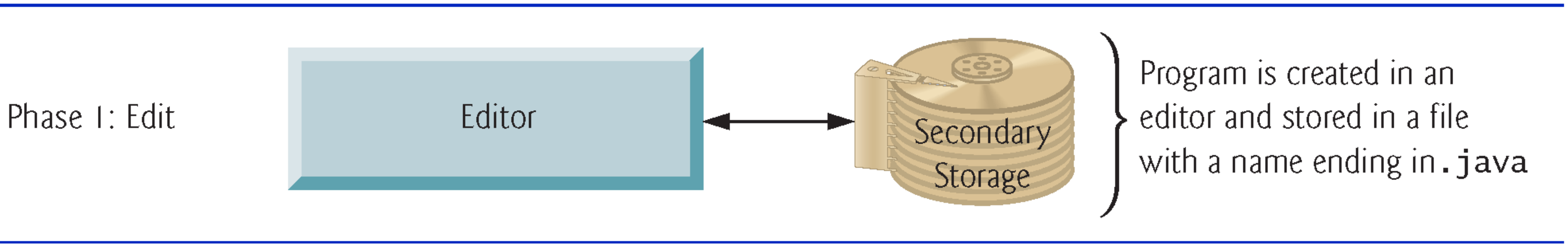
class header

method body

method header

A Typical Java Development Environment

- Phase 1 consists of editing a file with an *editor program*
  - Using the editor, you type a Java program (source code).
  - Make any necessary corrections.
  - Save the program.
  - Java source code files are given a name ending with the .java extension indicating that the file contains Java source code.


Phase 1: Edit
Editor
Secondary Storage
Program is created in an editor and stored in a file with a name ending in .java

Typical Java development environment—editing phase.

A Typical Java Development Environment (Cont.)

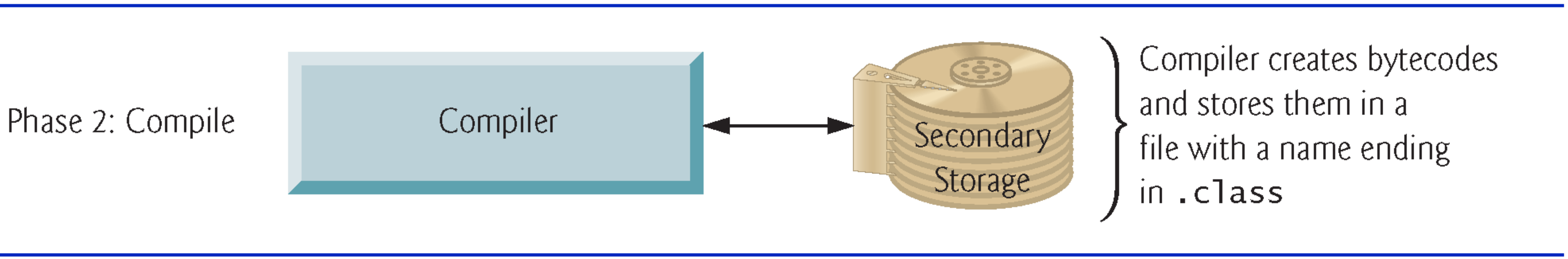
- Windows provides Notepad.
- MacOS provides TextEdit.
- Many freeware and shareware editors are also available online, including
  - Notepad++ (http://notepad-plus-plus.org)
  - EditPlus (http://www.editplus.com)
  - TextPad (http://www.textpad.com)
  - Sublime (https://www.sublimetext.com/download)
  - jEdit (http://www.jedit.org) and more.

A Typical Java Development Environment (Cont.)

- Phase 2: Compiling a Java Program into Bytecodes
  - Use the command javac (the Java compiler) to compile a program. For example, to compile a program called `Welcome.java`, you’d type

```
javac Welcome.java
```

  - If the program compiles, the compiler produces a .class file called `Welcome.class` that contains the compiled version.


Phase 2: Compile
Compiler
Secondary Storage
Compiler creates bytecodes and stores them in a file with a name ending in .class

Typical Java development environment—compilation phase.

A Typical Java Development Environment (Cont.)

- Java compiler translates Java source code into **bytecodes** that represent the tasks to execute, But Bytecode instructions are not machine language, and therefore cannot be directly executed by the CPU

- Instead, they are executed by the **Java Virtual Machine (JVM).** Bytecode is the machine language for the JVM.

- You can think of the JVM as a program that simulates a computer whose machine language is Java byte code.

A Typical Java Development Environment (Cont.)

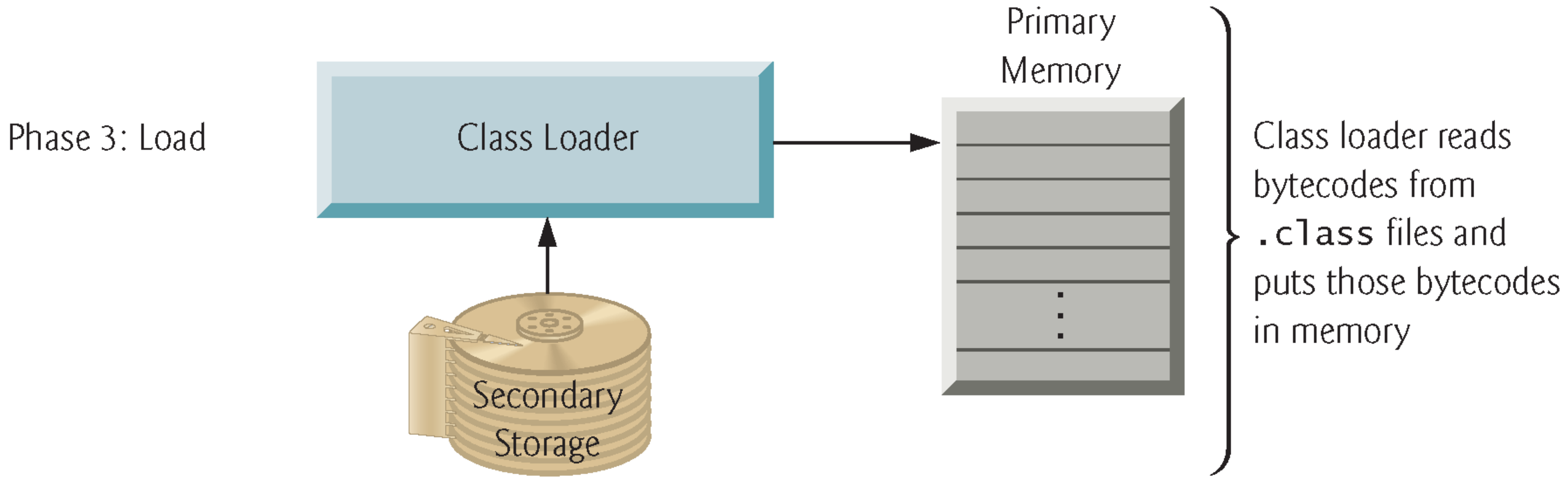
- Bytecode instructions are *platform independent*
- Bytecodes are portable
  - The same bytecode instructions can execute on any platform containing a JVM that understands the version of Java in which the bytecode instructions were compiled.
- The JVM is invoked by the java command. For example, to execute a Java application called `Welcome`, you'd type the command

```
java Welcome
```

The designers of the Java language introduced the (JVM) to allow Java programs to run on any device or operating system (known as the "Write once, run anywhere" principle),

A Typical Java Development Environment (Cont.)

- Phase 3: Loading a Program into Memory
  - The JVM places the program in memory to execute it—this is known as loading.
  - Class loader takes the `.class` files containing the program’s bytecodes and transfers them to primary memory.
  - Also loads any of the `.class` files provided by Java that your program uses.

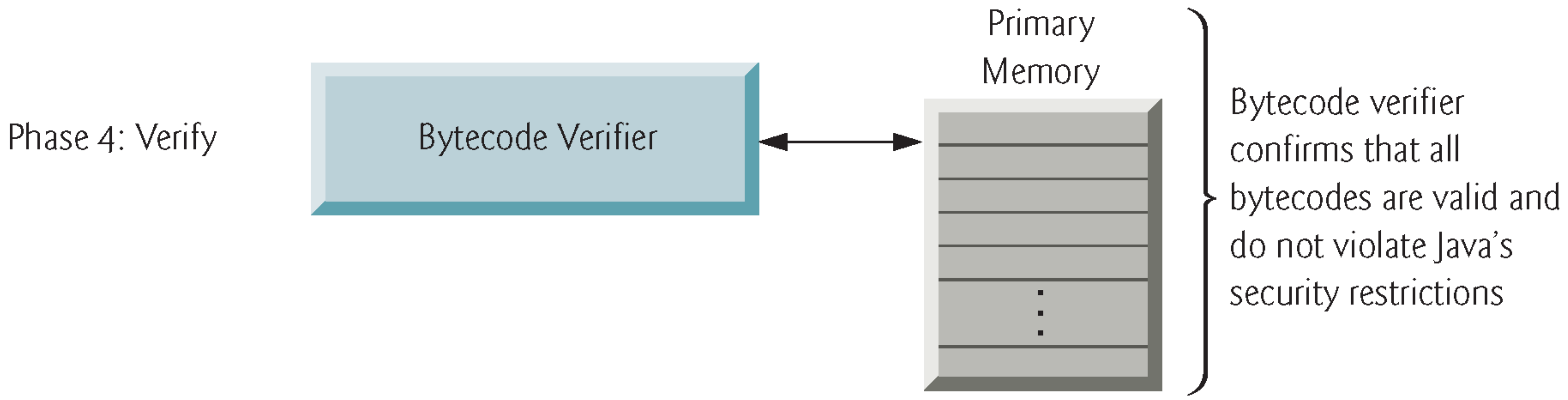

Phase 3: Load
Class Loader
Secondary Storage
Primary Memory
Class loader reads bytecodes from .class files and puts those bytecodes in memory

Typical Java development environment—loading phase.

A Typical Java Development Environment (Cont.)

- Phase 4: Bytecode Verification
  - As the classes are loaded, the bytecode verifier examines their bytecodes
  - Ensures that they’re valid and do not violate Java’s security restrictions.

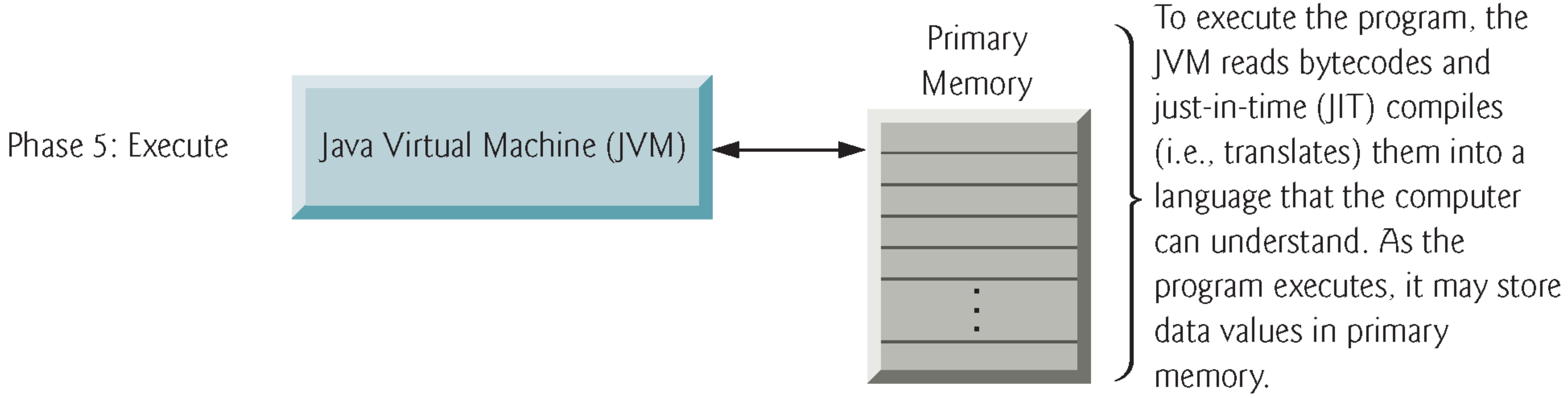
- Java enforces strong security to make sure that Java programs arriving over the network do not damage your files or your system (as computer viruses and worms might).


Phase 4: Verify
Bytecode Verifier
Primary Memory
Bytecode verifier confirms that all bytecodes are valid and do not violate Java's security restrictions

Typical Java development environment—verification phase.

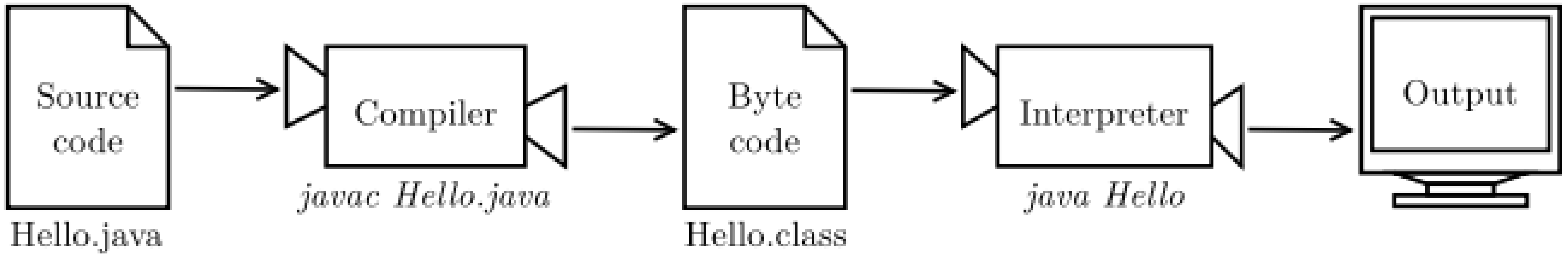
A Typical Java Development Environment (Cont.)

- Phase 5: Execution
    - The JVM executes the program's bytecodes.
    - When the JVM encounters these compiled parts again, the faster machine-language code executes.
    - Java programs go through *two* compilation phases
    - One in which source code is translated into bytecodes (for portability across JVMs on different computer platforms) and
    - A second in which, during execution, the bytecodes are translated into *machine language* for the actual computer on which the program executes.


Phase 5: Execute
Java Virtual Machine (JVM)
Primary Memory
To execute the program, the JVM reads bytecodes and just-in-time (JIT) compiles (i.e., translates) them into a language that the computer can understand. As the program executes, it may store data values in primary memory.

Typical Java development environment—execution phase.

The process of compiling and running a Java program



Java Software Edition

- The software that you use to create Java programs is referred to as JDK (Java Development Kit) or SDK (Software Development Kit).
- There are the following different editions of the JDK available from Oracle:
  - Java Standard Edition (SE) provides all the essential software tools necessary for writing Java applications.
  - Java Enterprise Edition (Java EE) provides tools for creating large business applications that employ servers and provide services over the Web.
  - Java Micro Edition (Java ME): provides a small, highly optimized runtime environment for consumer products such as cell phones, pagers, and appliances.
- We will use Java Standard Edition (SE) for this course

Integrated Development Environments (IDEs)

- **Integrated Development Environments (IDEs)** provide tools that support the software development process
- The most popular Java IDEs are:
    - Eclipse (http://www.eclipse.org)
    - IntelliJ IDEA (http://www.jetbrains.com)
    - NetBeans (http://www.netbeans.org)
- An IDE normally consists of a source code editor (text editor), build automation tools and a debugger.
    - Source code editor is the tool for us to write the source code
    - Build automation tools will “translate” the source code to the machine code for computer to understand and execute.
    - Debugger is a tool to help us to analysis our code and find errors.

What is a Java Development Kit (JDK)

- The JDK includes tools for developing and testing programs written in the Java programming language and running on the Java platform

- The software you'll need for this course is available free for download from the web.
- The most recent JDK version is available from:

https://www.oracle.com/ca-en/java/technologies/downloads/

- The JDK also contains the Java Runtime Environment (or JRE) which is the core of a Java program.

What is a Java Development Kit (JDK)

NetBeans Downloads:

- You can download the JDK/NetBeans bundle from: http://www.oracle.com/technetwork/java/javase/downloads/index.html
- The NetBeans version that's bundled with the JDK is for Java SE development, which you can download from: https://netbeans.org/downloads/

Demo: Your first program

(1) Create the source file:

- file and class name are case sensitive and must be matched exactly (except the .java part)

Example Code: HelloWorld.java

```
/**
 * The HelloWorld class implements an application
 * that displays "Hello World!" to the standard output
*/
public class HelloWorld {
  public static void main(String[] args) {
    // Display "Hello World!"
    System.out.println("Hello World!");
  }
}
```

Compiling and Running a Java Program

- The Java compiler is a **command line** utility.
- The command to compile a program is: (`javac` is the Java compiler)

```
javac filename.java
```

- To run a compiled Java program, use the `java` command:

```
java ClassFilename
```

- *`ClassFilename`* is the name of the .class file; however, you do not type the `.class` extension.

Summary so far

- If you want to program in Java, you should have:
  - Java JDK SE 8 (or higher)
  - A text Editor (e.g.: Sublime 3) Or an IDE (e.g.: NetBeans)
- Source code is just a plain text file. In Java, we give the filename an extension of .java to identify it as a source code file
- Compilation -- The compiler does syntax checking, translation to bytecode in files with the .class extension
- bytecode is a translation of the source code to an intermediate level of code
- The loader is part of the Java Virtual Machine
  - It loads the bytecode into memory and executes the instructions via an interpreter for the given platform (Windows, Mac, Linux, etc)
  - Each platform needs its own JVM, but the same bytecode runs on any JVM on any platform (i.e. the compiled version is portable)