Object-Oriented Programming with Java

Dr. Marenglen Biba

General Info

- Course : Object-Oriented Programming with Java (3 credit hours)
- Instructor : Dr. Marenglen Biba
- Office : Faculty building 1st floor
- Office Hours : Wednesday 16-17 PM or by appointment
- Phone : 42273056 / ext. 112
- E-mail : marenglenbiba@unyt.edu.al
- Course page : http://www.marenglenbiba.net/java/

- Course Location and Time: Laboratory Room 2B, Tuesday 15-18.
- Prerequisite: Introduction to Programming

Course Description

- This course introduces Java language and architecture.
- Students will learn how to program in Java and use some of its most important APIs.
- Special importance will be assigned to the Object-Oriented nature of Java and its use of polymorphism.

Course Outcomes

- use Java programming language in object-oriented program design
- understand the Java architecture
- understand and use inheritance and polymorphism as implemented in Java
- understand and use the exception handling mechanism of Java
- perform standard input-output operations
- understand and use multithreading in simple situations
- understand and use GUI components
- (tentative) understand and use JDBC to access databases from Java

Required Readings

- Java: How to Program. 8th ed. by Deitel & Deitel, (**required**)
- Thinking in Java. 4th ed. by Bruce Eckel, Pearson Education. (**recommended**)

Course Content

- Introduction
- Classes and Objects
- Control Statements
- Arrays and Enumerations
- Inheritance
- Polymorphism
- Collections
- Exceptions
- Standard IO
- GUI components in Java
- Threading
- JDBC

Grading Policy

Assignments	15%
Project	30%
Midterm	25%
Final	30%

Technology Expectations

- Internet use is necessary since students should regularly check the course home page
- Continued and regular use of e-mail is expected
- Students must keep copies of all assignments and projects sent by e-mail.

Before we start: why some students fail?

- Reasons
  - Lack of concentration?
  - Lack of continuity?
  - Lack of determination?
  - Lack of target?
  - Lack of fortune?
  - Lack of work
  - …
  - …
- Response
  - Hard work will help!!!

Recommendations

- Start studying now
- Do not be shy! Ask any questions that you might have. Every questions makes you a good candidate.
- The professor is a container of knowledge and the goal is to get most of him, thus come and talk.
- Respect the deadlines
- Respect the appointments
- Try to study from more than one source, Internet is great!
- If you have any problems come and talk with me in advance so that we can find an appropriate solution

GOOD LUCK!

Oracle Academy

- Sun Microsystems has become part of Oracle
- UNYT Computer Science Department is Oracle Academy.
- Check the link for further info
  - UNYT CSD is an Oracle Academy

Chapter 1 Introduction to Computers, the Internet and the Web

Java™ How to Program, 8/e

OBJECTIVES

In this Chapter you'll learn:

- Basic computer hardware, software and networking concepts.
- Basic object-technology concepts, such as classes, objects, attributes, behaviors, encapsulation and inheritance.
- The different types of programming languages and which languages are most widely used.
- A typical Java program-development environment.
- Java's role in developing distributed client/server applications for the Internet and the web.
- The history of the UML—the industry-standard object-oriented design language.
- The history of the Internet and the World Wide Web through Web 2.0.
- To test-drive Java applications.
- Some key recent software technologies.

1.1 Introduction
1.2 Computers: Hardware and Software
1.3 Computer Organization
1.4 Early Operating Systems
1.5 Personal, Distributed and Client/Server Computing
1.6 The Internet and the World Wide Web
1.7 Machine Languages, Assembly Languages and High-Level Languages
1.8 History of C and C++
1.9 History of Java
1.10 Java Class Libraries
1.11 Fortran, COBOL, Pascal and Ada
1.12 BASIC, Visual Basic, Visual C++, C# and .NET
1.13 Typical Java Development Environment
1.14 Notes about Java and *Java How to Program, Eighth Edition*
1.15 Test-Driving a Java Application

1.16 Software Engineering Case Study: Introduction to Object Technology and the UML
1.17 Web 2.0
1.18 Software Technologies
1.19 Wrap-Up
1.20 Web Resources

1.1 Introduction

- The core of the course emphasizes achieving program clarity through the proven techniques of object-oriented programming.

- We will follow a live-code approach — Java features presented in complete working Java programs.

1.1 Introduction (Cont.)

- You'll learn to write instructions commanding computers to perform tasks.
- Software (i.e., the instructions you write) controls hardware (i.e., computers).
- Java is one of today's most popular languages for developing software.

1.1 Introduction (Cont.)

- Over the years, many programmers learned structured programming.
- You'll learn structured programming and object-oriented programming — the key programming methodology used by programmers today.
- You'll create and work with many software objects.
  - Their internal structure is often built using structured-programming techniques.

1.1 Introduction (Cont.)

- Java has become the language of choice for implementing Internet-based applications and software for devices that communicate over a network.
- There are now billions of Java-enabled mobile phones and handheld devices.
- Java is the preferred language for meeting many organizations' enterprise-wide programming needs.

1.1 Introduction (Cont.)

- This course is based on Java Standard Edition (Java SE) 6, Update 11

- Java Enterprise Edition (Java EE)
  - geared toward developing large-scale, distributed networking applications and web-based applications.

- Java Micro Edition (Java ME)
  - geared toward developing applications for small, memory-constrained devices, such as cell phones, pagers and PDAs.

1.2 Computers: Hardware and Software

- Computer
  - Today's personal computers can perform billions of calculations in one second.
- Supercomputers are already performing *thousands of trillions (quadrillions) of instructions per second.*
- Computers process data *under the control of sets of instructions called computer programs*.
- These programs guide the computer through orderly sets of actions specified by people called computer *programmers*.

1.2 Computers: Hardware and Software (Cont.)

- A computer consists of various devices referred to as hardware
  - e.g., the keyboard, screen, mouse, disks, memory, DVD, CD-ROM and processing units

- The programs that run on a computer are referred to as software.

1.3 Computer Organization

- Logical units or sections of a computer
- Input unit. This "receiving" section obtains information (data and computer programs) from input devices *and places it at the disposal of the other units so that it can be processed.*
- Input devices
  - Keyboard, mouse, microphone, scanner, hard drives, CD drives, DVD drives, USB drives and more

1.3 Computer Organization (Cont.)

- Output unit. This “shipping” section takes information that the computer has processed and places it on various output devices to make it available for use outside the computer.
- Output devices
  - Screen, printer, speakers and more

1.3 Computer Organization (Cont.)

- Memory unit
  - Rapid-access, relatively low-capacity "warehouse" section retains information that has been entered through the input unit, making it immediately available for processing when needed.
  - Retains processed information until it can be placed on output devices by the output unit.
  - Information in the memory unit is volatile — it's typically lost when the computer's power is turned off.
  - Often called either memory *or primary memory*.

1.3 Computer Organization (Cont.)

- Arithmetic and logic unit (ALU)
  - "Manufacturing" section performs calculations, such as addition, subtraction, multiplication and division.
  - Contains the mechanisms that allow the computer to make decisions.
  - In today's systems, the ALU is usually implemented as part of the next logical unit, the CPU.

1.3 Computer Organization (Cont.)

- Central processing unit (CPU)
  - "Administrative" section coordinates and supervises the operation of the other sections.
  - Tells the input unit when information should be read into the memory unit.
  - Tells the ALU when information from the memory unit should be used in calculations and tells the output unit when to send information from the memory unit to certain output devices.
  - Many of today's computers have multiple CPUs—such computers are called multiprocessors.
  - A multi-core processor implements multiprocessing on a single integrated circuit chip.
  - Multithreading can benefit from multiprocessors. We will see how to implement Multithreading in Java.

1.3 Computer Organization (Cont.)

- Secondary storage unit
  - Long-term, high-capacity “warehousing” section.
  - Programs or data not actively being used by the other units normally are placed on secondary storage devices (e.g., your hard drive) until they are again needed, possibly hours, days, months or even years later.
  - Information on secondary storage devices is persistent—it is preserved even when the computer’s power is turned off.
  - Examples of secondary storage: CDs, DVDs and flash drives

1.4 Early Operating Systems

- Early computers performed one job *or task at a time*.
  - Single-user batch processing
  - The computer ran a single program while processing data in groups or batches.
- Operating systems
  - Developed to make using computers more convenient.
  - Smoothed and speeded up the transition between jobs, increasing the amount of work, or throughput, computers could process in a given time.
- Multiprogramming
  - Simultaneous operation of many jobs that are competing to share the computer's resources.

1.5 Personal, Distributed and Client/Server Computing

- 1977: Apple Computer popularized personal computing.
- 1981: IBM introduced the IBM Personal Computer.
  - Quickly legitimized personal computing in business, industry and government organizations.
- Early personal computers could be linked together in computer networks, sometimes over telephone lines and sometimes in local area networks (LANs) within an organization.
  - Led to distributed computing — computing distributed over networks to the sites where the organization's work is performed.

1.5 Personal, Distributed and Client/Server Computing (Cont.)

- Personal computers now are as powerful as the million-dollar machines of just a few decades ago.
- Servers store data that may be used by client computers distributed throughout the network—hence the term client/server computing.
- Java is widely used for writing software for computer networking and for distributed client/server applications.

1.6 The Internet and the World Wide Web

- The Internet
  - Global network of computers
  - Has its roots in the 1960s, when funding was supplied by the U.S. Department of Defense.
  - Now accessible by billions of computers and computer-controlled devices worldwide.
- World Wide Web
  - Allows computer users to locate and view multimedia-based documents on almost any subject over the Internet.
- Java is widely used to build programs for WWW

1.7 Machine Languages, Assembly Languages and High-Level Languages

- Programmers write instructions in various programming languages, some directly understandable by computers and others requiring intermediate translation steps.
- Three general language types:
  - Machine languages
  - Assembly languages
  - High-level languages

1.7 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

- Any computer can directly understand only its own machine language.
  - This is the computer's "natural language," defined by its hardware design.
  - Generally consist of strings of numbers (ultimately reduced to 1s and 0s) that instruct computers to perform their most elementary operations one at a time.
  - Machine dependent — a particular machine language can be used on only one type of computer.

1.7 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

- English-like abbreviations that represent elementary operations formed the basis of assembly languages.

- Translator programs called assemblers convert assembly-language programs to machine language.

1.7 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

- High-level languages
  - Single statements accomplish substantial tasks.
  - Compilers convert high-level language programs into machine language.
  - Allow you to write instructions that look almost like everyday English and contain commonly used mathematical notations.
- C, C++, Microsoft's .NET languages (e.g., Visual Basic, Visual C++ and C#) are among the most widely used high-level programming languages; Java is by far the most widely used.

1.7 Machine Languages, Assembly Languages and High-Level Languages (Cont.)

- Compiling a high-level language program into machine language can take a considerable amount of computer time.
- Interpreter programs execute high-level language programs directly, although slower than compiled programs run.
- Java uses a clever mixture of compilation and interpretation to run programs.

1.8 History of C and C++

- Java evolved from C++, which evolved from C, which evolved from BCPL and B.
- C
  - Originally implemented in 1972
  - Evolved from B by Dennis Ritchie at Bell Laboratories
  - Became widely known as the UNIX operating system's development language
  - Today, most of the code for general-purpose operating systems is written in C or C++.

1.8 History of C and C++ (Cont.)

- C++
  - An extension of C
  - Developed by Bjarne Stroustrup in the early 1980s at Bell Laboratories
  - Provides capabilities for *object-oriented programming*.
  - Hybrid language — it's possible to program in either a C-like style, an object-oriented style or both.

1.9 History of Java

- Microprocessors are having a profound impact in intelligent consumer-electronic devices.
- 1991
  - Recognizing this, Sun Microsystems funded an internal corporate research project, which resulted in a C++-based language named Java
  - Created by James Gosling.
- 1993
  - The web exploded in popularity
  - Sun saw the potential of using Java to add dynamic content to web pages.
- Java garnered the attention of the business community because of the phenomenal interest in the web.

1.10 Java Class Libraries

- Java programs consist of pieces called classes.
- Classes include methods that perform tasks and return information when the tasks complete.
- Java class libraries
  - Rich collections of existing classes
  - Also known as the Java APIs (Application Programming Interfaces)
- Two aspects to learning the Java "world."
  - The Java language itself
  - The classes in the extensive Java class libraries
- Download the Java API documentation
  - http://www.oracle.com/technetwork/java/javase/documentation/index.html

Software Engineering Observation 1.1

Use a building-block approach to creating your programs. Avoid reinventing the wheel—use existing pieces wherever possible. This software reuse *is a key benefit of object-oriented programming.*

1.10 Java Class Libraries (Cont.)

- Programming tips
  - Software Engineering Observations — explain concepts that affect and improve the overall architecture and quality of software systems.
  - Good Programming Practices — help you write programs that are clearer, more understandable, more maintainable and easier to test and debug — i.e., remove programming errors.
  - Common Programming Errors — discuss problems to watch out for and avoid.

1.10 Java Class Libraries (Cont.)

- Programming tips (cont.):
  - Performance Tips—techniques for writing programs that run faster and use less memory
  - Portability Tips—techniques to help you write programs that can run, with little or no modification, on a variety of computers
  - Error-Prevention Tips—techniques for removing bugs from your programs
  - Look-and-Feel Observations—techniques to help you design the “look” and “feel” of your applications’ user interfaces

Software Engineering Observation 1.2

When programming in Java, you'll typically use the following building blocks: classes and methods from class libraries, classes and methods you create yourself and classes and methods that others create and make available to you.

Performance Tip 1.1

Using Java API classes and methods instead of writing your own versions can improve program performance, because they are carefully written to perform efficiently. This technique also shortens program development time.

Portability Tip 1.1

Using classes and methods from the Java API instead of writing your own improves program portability, because they are included in every Java implementation.

1.11 Fortran, COBOL, Pascal and Ada

- Fortran *(FORmula TRANslator)*
  - Developed by IBM Corporation in the mid-1950s
  - Used for scientific and engineering applications that require complex mathematical computations.
  - Still widely used in engineering applications.
- COBOL (COmmon Business Oriented Language)
  - Developed in the late 1950s by computer manufacturers, the U.S. government and industrial computer users
  - Used for commercial applications that require precise and efficient manipulation of large amounts of data.
  - Much business software is still programmed in COBOL.

1.11 Fortran, COBOL, Pascal and Ada (Cont.)

- Research in the 1960s resulted in the evolution of structured programming
  - A disciplined approach to writing programs that are clearer, easier to test and debug and easier to modify than large programs produced with previous techniques.
- Pascal
  - Developed by Professor Niklaus Wirth in 1971
  - Designed for teaching structured programming in academic environments.

1.11 Fortran, COBOL, Pascal and Ada (Cont.)

- Ada programming
  - Developed under the sponsorship of the U.S. Department of Defense (DOD) during the 1970s and early 1980s.
  - The DOD wanted a single language to fill most of its needs.
  - Named after Lady Ada Lovelace, daughter of poet Lord Byron.
    - She's credited with writing the world's first computer program in the early 1800s.
  - Supports **multitasking** — allows programmers to specify that many activities in a program are to occur in parallel.
  - Java, through a technique called **multithreading**, also enables programmers to write programs with parallel activities.

1.12 BASIC, Visual Basic, Visual C++, C# and .NET

- BASIC (Beginner's All-Purpose Symbolic Instruction Code)
  - Developed in the mid-1960s at Dartmouth College as a means of writing simple programs.
  - Used to familiarize novices with programming techniques.
- Microsoft's Visual Basic
  - Introduced in the early 1990s to simplify the development of Microsoft Windows applications.
- Microsoft's latest development tools
  - Corporatewide strategy for integrating the Internet and the web into computer applications.
  - Implemented in Microsoft's .NET platform
  - Three primary programming languages: Visual Basic (based on the original BASIC), Visual C++ (based on C++) and C# (based on C++ and Java, and developed expressly for the .NET platform).

1.13 Typical Java Development Environment

- Java program development and execution cycle (illustrated in Fig. 1.1).

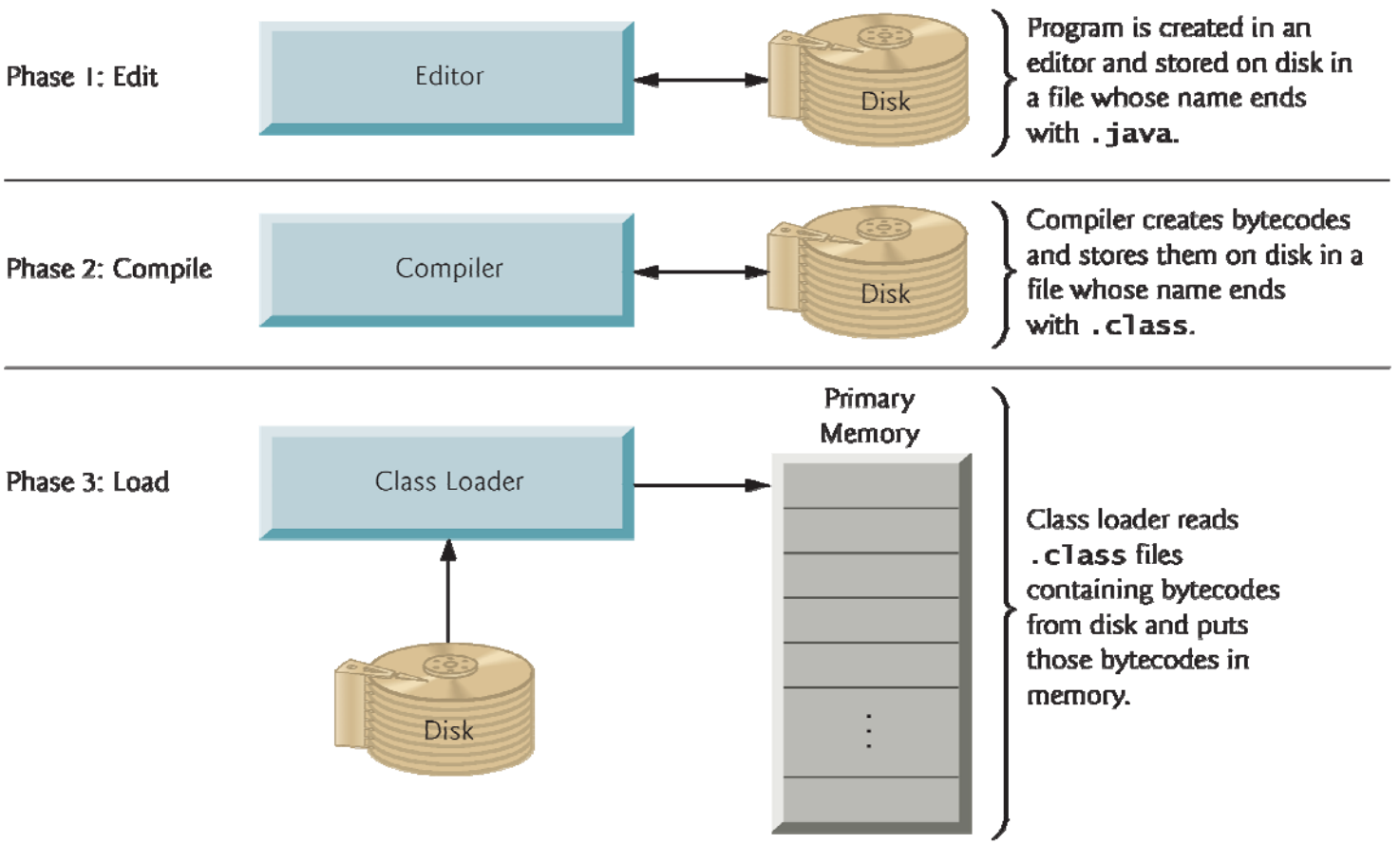


Fig. 1.1 | Typical Java development environment. (Part 1 of 2.)

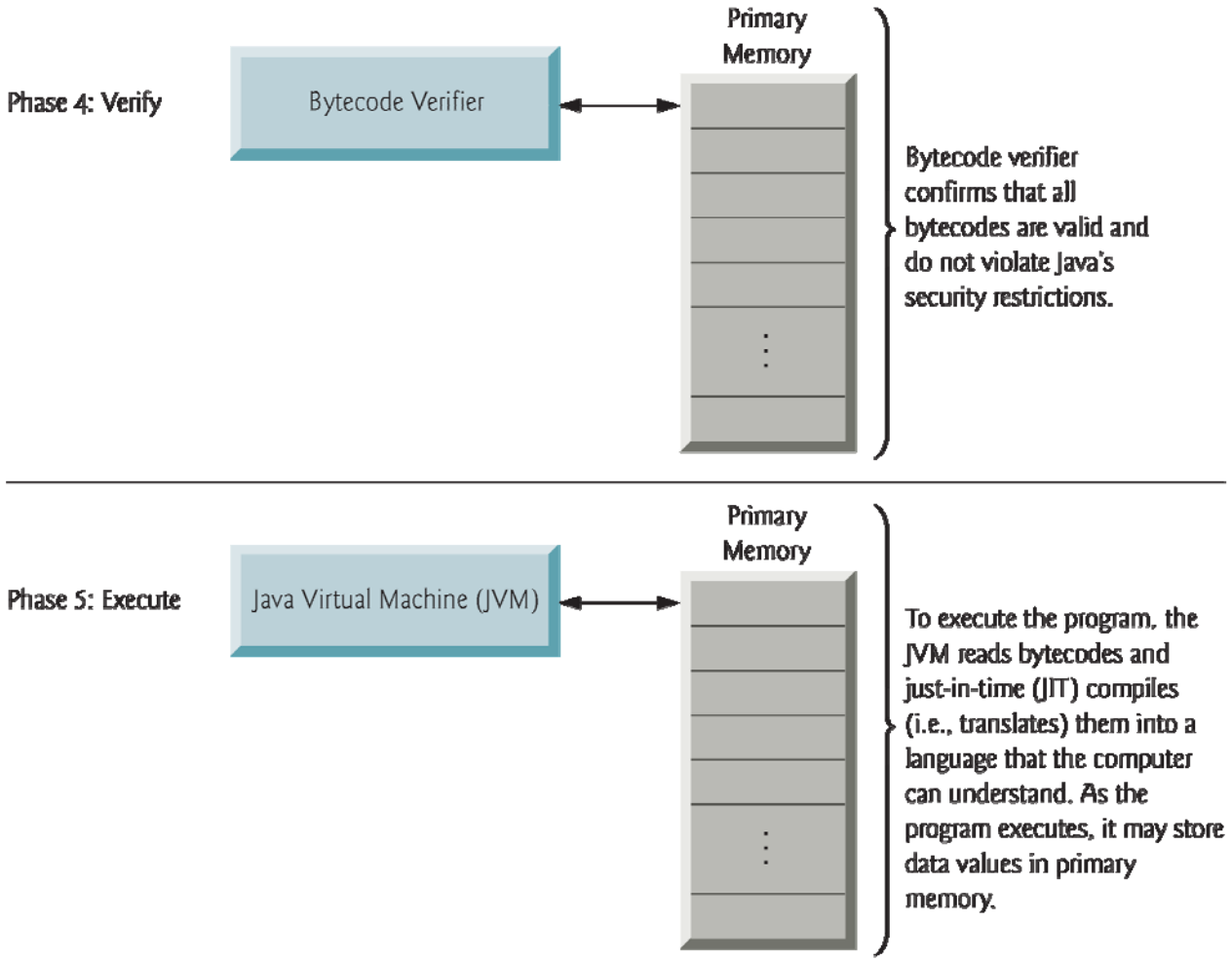


Fig. 1.1 | Typical Java development environment. (Part 2 of 2.)

1.13 Typical Java Development Environment (Cont.)

- Java programs normally go through five phases
  - edit
  - compile
  - load
  - verify
  - execute

1.13 Typical Java Development Environment (Cont.)

- We discuss these phases in the context of the Java SE Development Kit 6 (JDK6) from Sun Microsystems, Inc.
- Download the most up-to-date JDK and its documentation from:
  - http://www.oracle.com/technetwork/java/javase/downloads/index.html.
- Carefully follow the installation instructions for the JDK provided in the Before You Begin section of the book to ensure that you set up your computer properly to compile and execute Java programs.
- Sun's New to Java Center at:
  - http://www.oracle.com/technetwork/java/javase/overview/index.html

1.13 Typical Java Development Environment (Cont.)

- Phase 1 consists of editing a file with an editor program *(normally known simply as an editor).*
  - Type a Java program (source code) using the editor
  - Make any necessary corrections
  - Save the program
    - A file name ending with the `.java` extension indicates that the file contains Java source code.
  - Linux editors: `vi` and `emacs`.
  - Windows editors: Notepad, EditPlus (`www.editplus.com`), TextPad (`www.textpad.com`) and jEdit (`www.jedit.org`).

1.13 Typical Java Development Environment (Cont.)

- Integrated development environments (IDEs)
  - Provide tools that support the software-development process, including editors for writing and editing programs and debuggers for locating logic errors — errors that cause programs to execute incorrectly.
- Popular IDEs
  - Eclipse (`www.eclipse.org`)
  - NetBeans (`www.netbeans.org`)
  - JBuilder (`www.codegear.com`)
  - JCreator (`www.jcreator.com`)
  - BlueJ (`www.blueJ.org`)
  - jGRASP (`www.jgrasp.org`)

1.13 Typical Java Development Environment (Cont.)

- Phase 2
  - 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 of the program.

1.13 Typical Java Development Environment (Cont.)

- Java compiler translates Java source code into bytecodes that represent the tasks to execute.
- Bytecodes are executed by the Java Virtual Machine (JVM)—a part of the JDK and the foundation of the Java platform.
- JRE
  - You can download this from the same page of JDK
- Virtual machine (VM) — a software application that simulates a computer
  - Hides the underlying operating system and hardware from the programs that interact with it.
- If the same VM is implemented on many computer platforms, applications that it executes can be used on all those platforms.

1.13 Typical Java Development Environment (Cont.)

- Bytecodes are platform independent
  - They do not depend on a particular hardware platform.
- Bytecodes are portable
  - The same bytecodes can execute on any platform containing a JVM that understands the version of Java in which the bytecodes 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
```

1.13 Typical Java Development Environment (Cont.)

- Phase 3
  - 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.
    - The `.class` files can be loaded from a disk on your system or over a network.

1.13 Typical Java Development Environment (Cont.)

- Phase 4
  - As the classes are loaded, the bytecode verifier examines their bytecodes
    - Ensures that they are 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).

1.13 Typical Java Development Environment (Cont.)

- Phase 5
  - The JVM executes the program's bytecodes.
  - JVM typically uses a combination of interpretation and just-in-time (JIT) compilation.
  - Analyzes the bytecodes as they are interpreted, searching for hot spots — parts of the bytecodes that execute frequently.
  - A just-in-time (JIT) compiler (the Java HotSpot compiler) translates the bytecodes into the underlying computer's machine language.
  - When the JVM encounters these compiled parts again, the faster machine-language code executes.
  - Java programs actually go through two compilation phases
    - One in which source code is translated into bytecodes (for portability across computer platforms)
    - A second in which, during execution, the bytecodes are translated into machine language for the actual computer on which the program executes.

Common Programming Error 1.1

Errors such as division by zero occur as a program runs, so they are called ***runtime errors*** *or* ***execution-time errors****.* ***Fatal runtime errors*** *cause programs to terminate immediately without having successfully performed their jobs.* ***Nonfatal runtime errors*** *allow programs to run to completion, often producing incorrect results.*

1.14 Notes about Java and *Java How to Program, Eighth Edition*

- Web-based version of the Java API documentation
  - java.sun.com/javase/6/docs/api/index.html
- Download this documentation to your own computer
  - java.sun.com/javase/downloads/
- Additional technical details on many aspects of Java development
  - java.sun.com/reference/docs/index.html
- JDK Programmers Guide
  - http://java.sun.com/javase/6/docs

Good Programming Practice 1.1

Write your Java programs in a simple and straightforward manner. This is sometimes referred to as KIS ("keep it simple"). Do not "stretch" the language by trying bizarre usages.

Portability Tip 1.2

Although it's easier to write portable programs in Java than in most other programming languages, differences between compilers, JVMs and computers can make portability difficult to achieve. Simply writing programs in Java does not guarantee portability.

Error-Prevention Tip 1.1

Always test your Java programs on all systems on which you intend to run them, to ensure that they'll work correctly for their intended audiences.

Good Programming Practice 1.2

Read the documentation for the version of Java you're using. Refer to it frequently to be sure you're aware of the rich collection of Java features and are using them correctly.

1.15 Test-Driving a Java Application

- Checking your setup. Read the *Before You Begin section of the book to confirm that you've set up Java properly on your computer.*

- *Writing a simple Java Program.*
  - *Welcome*
  - *Perform simple operations*
  - *Print to the console*

- *Compiling the program*
  - *Javac command or*
  - *Netbeans*

- *Executing*
  - *The java command or*
  - *Run in Netbeans*

First Program in Java

```
public class Operator {

public static void main( String args[ ] ) {
   System.out.println("**************************************");
   System.out.println( "Welcome, this is your first program in Java" );
   int x = 30;
   int y = 2;
   int result = x * y + 9 / 3;
   System.out.print("The result is: ");
   System.out.println(result);
   System.out.println("**************************************");
}
}
```

1.16 Software Engineering Case Study: Introduction to Object Technology and the UML

- Unified Modeling Language™ (UML™)
  - A graphical language that allows people who design software systems to use an industry-standard notation to represent them.
- Official site
  - http://www.uml.org/
- Many software to use UML
  - StarUML
  - ArgoUML
  - …

1.16 Software Engineering Case Study (Cont.)

- Objects
  - Reusable software components that model real world items.
  - Humans think in terms of objects. People, animals, plants, cars, etc.
  - Have attributes (e.g., size, shape, color and weight)
  - Exhibit behaviors (e.g., a ball rolls, bounces, inflates and deflates; a baby cries, sleeps, crawls, walks and blinks; a car accelerates, brakes and turns; a towel absorbs water).

1.16 Software Engineering Case Study (Cont.)

- Object-oriented design (OOD)
  - Models software in terms similar to those that people use to describe real-world objects.
  - Class relationships
  - Inheritance relationships
  - Models communication between objects (via messages).
  - Encapsulates attributes and operations (behaviors) into objects.
  - Information hiding
    - Objects may know how to communicate with one another across well-defined interfaces, but normally they are not allowed to know how other objects are implemented.

1.16 Software Engineering Case Study (Cont.)

- Object oriented languages
  - Programming in such a language, called object-oriented programming (OOP)
  - Allows you to implement an object-oriented design as a working system.
  - Java is object oriented.
- Focus on creating classes.
  - Each contains fields and the set of methods that manipulate the fields and provide services to clients (i.e., other classes that use the class).
  - Programmers use existing classes as the building blocks for constructing new classes.
- Classes are to objects as blueprints are to houses.
- Associations—relationships between classes.
- Packaging software as classes facilitates reuse.

1.16 Software Engineering Case Study (Cont.)

- Object-oriented analysis and design (OOAD)
  - Analyzing your project's requirements (i.e., determining *what the system is supposed to do*) and developing a design that satisfies them (i.e., *deciding how the system should do it*).
- Unified Modeling Language (UML)—A single graphical language for communicating the results of *any OOAD process* has come into wide use.
- The UML is the most widely used graphical representation scheme for modeling object-oriented systems.

Example of OOAD

- You are probably wearing on your wrist one of the world's most common types of objects — a watch.
- Let's discuss how each of the following terms and concepts applies to the notion of a watch:
  - object,
  - attributes,
  - behaviors,
  - class,
  - inheritance (consider, for example, an alarm clock),
  - abstraction,
  - modeling,
  - messages,
  - encapsulation,
  - interface
  - information hiding.

OOAD example

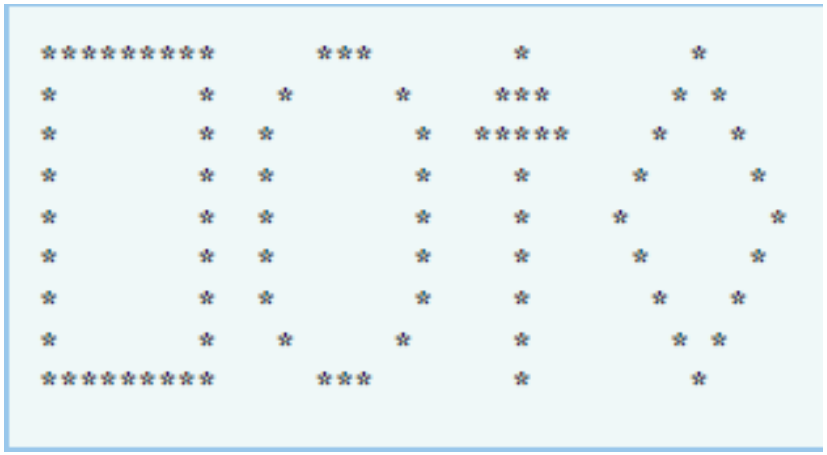
- The entire watch is an object that is composed of many other objects (such as the moving parts, the band, the face, etc.)
- Watch attributes are time, color, band, style (digital or analog), etc.
- The behaviors of the watch include setting the time and getting the time.
- A watch can be considered a specific type of clock (as can an alarm clock).
- With that in mind, it is possible that a class called Clock could exist from which other classes such as watch and alarm clock could inherit the basic features in the clock.

OOAD example

- The watch is an abstraction of the mechanics needed to keep track of the time.
- The user of the watch does not need to know the mechanics of the watch in order to use it; the user only needs to know that the watch keeps the proper time.
  - In this sense, the mechanics of the watch are encapsulated (hidden) inside the watch.
- The interface to the watch (its face and controls for setting the time) allows the user to set and get the time. The user is not allowed to directly touch the internal mechanics of the watch.
- All interaction with the internal mechanics is controlled by the interface to the watch.
- The data members stored in the watch are hidden inside the watch and the member functions (looking at the face to get the time and setting the time) provide the interface to the data.

End of Class

- Assignment 1
  - Write a program in Java called Shapes that prints the following:



  - Zip the code into a .zip document and submit it by email to marenglenbiba@unyt.edu.al.
    - Mail Subject: Java Assignment 1 – Name Surname
  - Deadline 18/10/2010 23:59.