Problem Solving and Software Engineering

Chapter 1

Chapter Contents

Chapter Objectives

- Indicate variety of uses for computers
- Cover historical development of Java
- Note the platform-independent scheme of Java
- Take first looks at Java applications and applets, comparing and contrasting
- Illustrate software development life cycle
- Note ethical issues and principles for computing professionals

Importance of Computers in our World

- Areas of human endeavor where computers are used:
  - Business and Finance
  - Industry
  - Government
  - Medicine
  - Entertainment
  - Science
  - Information Technology

A Brief History of OOP and Java

- Early high level languages
  - FORTRAN, COBOL, LISP
- More recent languages
  - BASIC, Pascal, C, Ada, Smalltalk, C++, Java
- UNIX operating system upgrades prompted the development of the C language
- Powerful language … but not the best intro to programming
  - Difficult for beginning programmers
  - Does not fit modern programming strategies

A Brief History of OOP and Java

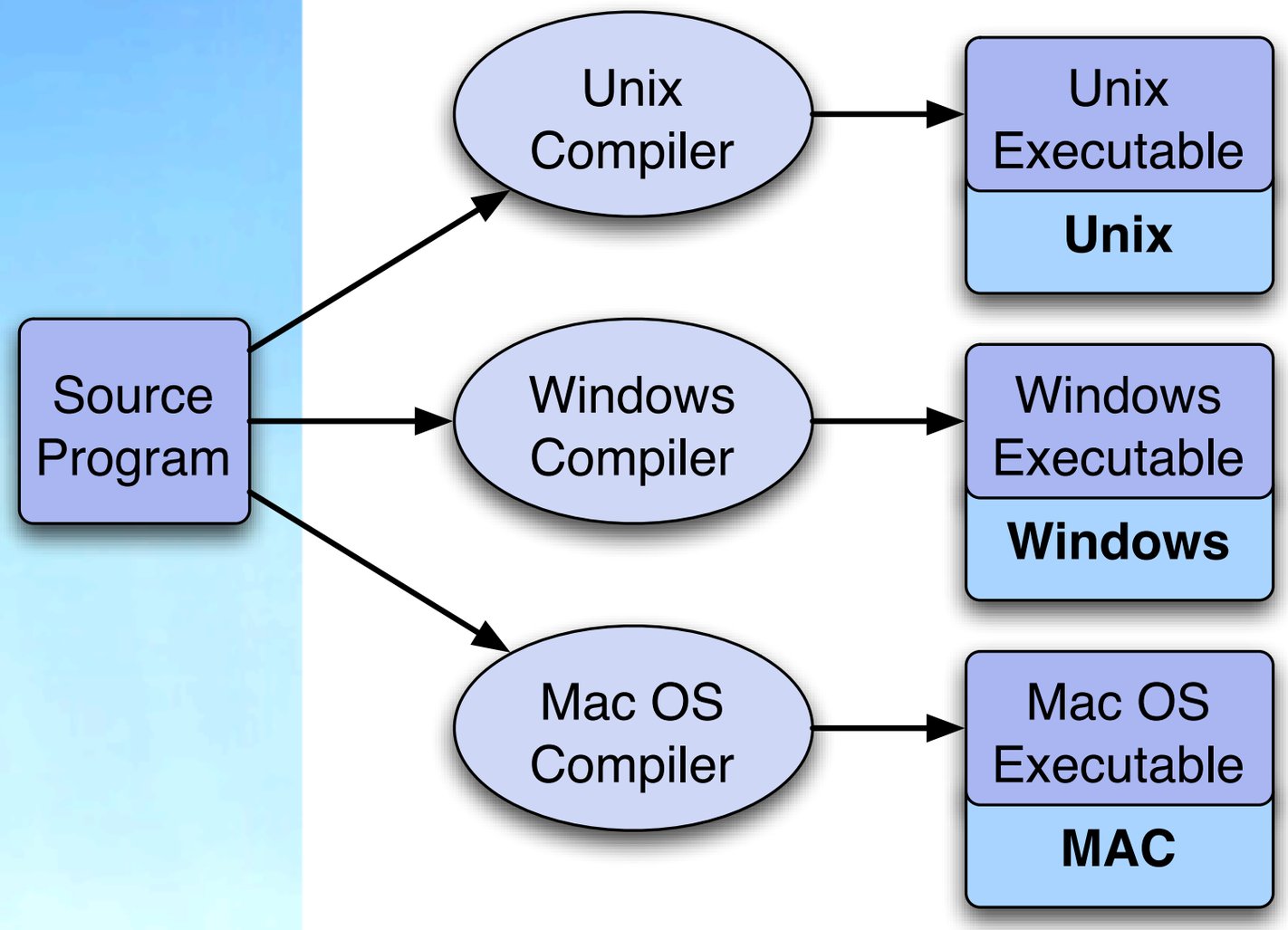
- Simula-67 was a language which facilitated the modeling of real-world objects
  - New language feature called the “class”
  - A class was extendable through “inheritance”
- This was the foundation needed for Object Oriented Programming, OOP
- Smalltalk-80 used this concept
  - This is the first truly object-oriented language

A Brief History of OOP and Java

- C was extended to include “classes” and in 1983 became C++
- Java, originally conceived to control household devices using the OOP concept
  - Meant to be platform (or device) independent
- Soon this was seen as well suited for running small programs (applets) on Internet web pages
  - 1995, Netscape browsers support Java applets
  - This did much to establish Java as a major language

A Brief History of OOP and Java

- Note typical implementation of a program on different platforms

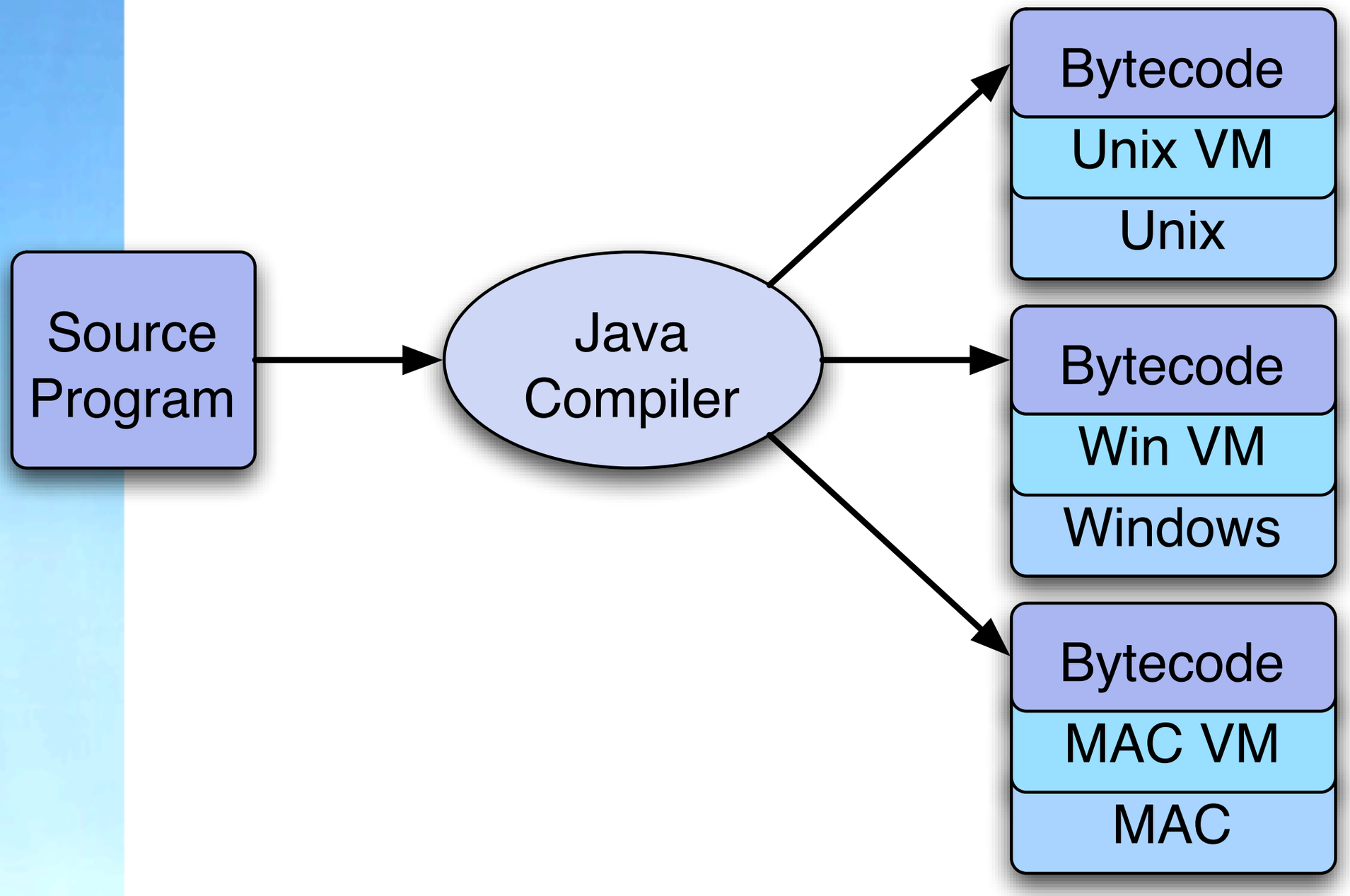


We need Platform Independence

A Brief History of OOP and Java

- Contrast compiler with interpreter
  - Compiler runs once, translates to machine code for commands of whole program
  - Interpreter translates one command at a time
- Java combines these two
  - Java compiler generates intermediate "bytecode"
  - An interpreter is developed for each platform to interpret the bytecode
  - Interpreter called the Java Virtual Machine or JVM

Java Platform Independence



Introduction to Java Application Programs

- What is a program?
  - A collection of statements
  - Written in a programming language
  - Specify the steps taken to solve a problem
- Programs have grammar rules specify:
  - How the statements are formed
  - How the statements are combined

Introduction to Java Application Programs

- Java is an object oriented programming language
  - Uses objects to carry out the tasks
  - Sends messages to the objects to perform the tasks
  - Objects interact with each other to do the tasks
  - An actual object is called an instance of a class
    - The class is the declaration of or blueprint for the object

Introduction to Java Application Programs

- Object oriented programs:
  - A collection of object interactions that solve a problem

- Note the similarity of this definition to the definition for a program

Display a Greeting Figure 1.2

```
import ann.easyio.*;
import java.util.*;
class Greeter1 extends Object
{
  public static void main (String [] args)
  {
    Date currentDate = new Date();
   String today = currentDate.toString();
   Screen theScreen = new Screen();
   theScreen.println ("Welcome! today, " +
        today + ", you begin to study Java!");
    }
}
```

Program Components

- Comments
  - Anything between /* and */ or following // on a single line
  - Documentation for humans to read
  - Compiler ignores
- Objects for this program
  - The screen
  - Current time and date
  - A string to store the time and date information

Classes and Packages

- Objects are constructed from classes
- Classes used for our objects
- Classes accessed by use of import statements

```
import java.util.*
```

Class Name	Package Name
Screen	`ann.easyio`
Date	`java.util`
String	`java.lang`

Class Declaration

- Syntax:

```
class   ClassName   extends Object
{
   Declarations of class members
}
```

- Note the **`extends`** clause
  - specifies that the **`ClassName`** inherits attributes and behaviors of **`Object`**
  - also it will add new attributes and behaviors
  - **`ClassName`** is the subclass or derived class
  - **`Object`** is the superclass or base class

Class Members

- Specified between outermost set of curly braces

  { … }

- Members can be
  - variables that store values
  - methods which perform operations on the variables
- The main method's declaration

```
public static void main (String [ ] args)
{
   a list of Java statements
}
```

First program statement is first of these
Last program statement is last of these

Declarations

- Example

```
Date currentDate = new Date()
```

Class specification

Variable name for a `Date` object

Creates a new object

Calling an Object Method

- Example

Object name

Method name

Parameters

```
theScreen.println( "Welcome ! Today" +
   today + “you begin your study of Java"
);
```

Introduction to Java Applet Programs

- Applications are stand alone programs
  - executed with Java interpreter
- Applet is a small program
  - can be placed on a web page
  - will be executed by the web browser
  - give web pages “dynamic content”

Applet to Display a Greeting Figure 1.3

```
import javax.swint.*;
import java.util.*;
public class Greeter2 extends JApplet
{
   public void init()
   { Date currentDate = new Date();
     String today = currentDate.toString();
     JLabel greeting = new JLabel
        ("Welcome ! today, " + today +
         ", you begin your study of Java!");
   }
}
```

Java Applets

- Built using one of general definitions of applets
  - `Applet` class
  - `JAapplet` class
- Java applets are usually graphical
  - Draw graphics in a defined screen area
  - Enable user interaction with GUI elements

Java Applet Classes

- Abstract Windowing Toolkit AWT
  - Earlier versions of Java
  - `Applet` class is one of the AWT components
- Java Foundation Classes JFC
  - Extension to Java in 1997
  - Has a collection of Swing components for enhanced GUIs
  - Swing component classes begin with J
  - Note the import `javax.swing.*;` line in Figure 1.3

Applet Declaration

- Syntax (note difference from application declaration)

```
public class ClassName extends JAapplet
```

ClassName is an object that is a subclass of JApplet

Body of an Applet

- Note there is no `main()` method in an applet
  - `JApplet` class provides other methods instead of a `main` method
- First method executed is the `init()` method

Applet Statements

- Declaration statements for `Date` are same as in previous example
- Labels declared

```
JLlabel greeting = new JLabel ( … )
```

  - freestanding strings of text
  - not part of a button or menu
- Content pane
  - portion of window where label is added

```
getContentPane().add(greeting)
// add() method  called
// greeting is the parameter
```

Applets and Web Pages – HTML

- Applets embedded in a web page
  - Executed when web page loaded by browser
- Web pages structured with HTML codes
  - HyperText Mark-up Language
- Syntax

```
<command>   <-------- Turns format on
  .  .  .
</command>  <-------- Turns format off
```

Applets and Web Pages – HTML

- Embedding Java applets
  - Insert applet tags

```
<APPLET>
</APPLET>
```

- Call the specific applet by its file name
  -
```
<APPLET CODE = "Whatever.class"
     WIDTH = nnn HEIGHT = mmmm>
<\APPLET>
```
  - Where `nnn` and `mmm` are specific pixel sizes

Applets and Web Pages – HTML

- Create the web page code using a text editor
- Save it with an .html suffix
- Open this file with appletviewer or with a web browser that supports Java

```
<HTML>
<HEAD>
</HEAD>
<BODY>
<APPLET CODE = ... >
</APPLET>
</BODY>
</HTML>
```

Applets and Web Pages – HTML

- Client Web browser anywhere can access this web page from its host server
- Embedded Java applet runs on client browser (of any type platform)
- This means a client anywhere on any type of platform can run a piece of software developed on any other type of platform

Platform Independence

Problem Solving through Software Engineering

- Common steps or phases in software development
  - Design
    - create an algorithm to solve the problem
  - Coding
    - use the high-level language according to its syntax rules
  - Testing – Execution – Debugging
    - try out variety of possibilities, correct problems
  - Maintenance
    - update, modify for changing needs

Object Centered Design

1. Behavior: state precisely what program should do
2. Objects: identify real-world objects
   - some will be primitive types
   - some will need a new class to represent
3. Operations: what actions do the objects do or have done to them
4. Algorithm: arrange the objects and operations in an order that solves the problem

Sample Problem

- Write a program to compute a sprinter's average speed in kilometers per hour, given distance (meters) and time elapsed (seconds)

Behaviors

- Display prompt for distance in meters
- Receive input from keyboard
- Display prompt for time in seconds
- Receive input from keyboard
- Compute kph
- Display titled results

Objects

- Program
- Prompt for distance
- Distance
- Keyboard
- Screen
- Prompt for seconds
- Time
- Calculated speed

Types of Objects

Description of Object	Type	Kind	Name
Program	??	??	??
screen	Screen	variable	`thescreen`
prompt for dist	String	constant	
distance	double	variable	`meters`
keyboard	Keyboard	variable	`thekeyboard`
prompt for time	String	constant	
time	double	variable	`seconds`
speed	double	variable	`kmPerHour`

Operations

- Display on `theScreen` prompt for distance
- Read a value from `theKeyboard`
- Store it in `meters`
- Display on `theScreen` prompt for time
- Read a value from `theKeyboard`
- Store it in `seconds`
- Compute `kilometersPerHour`
- Display `kilometersPerHour` on `theScreen`

Algorithm

1. Construct `theKeyboard` and `theScreen`
2. Ask `theScreen` to display prompt for dist
3. Ask `theKeyboard` to read a double value, store in `meters`
4. Ask `theScreen` to display prompt for time
5. Ask `theKeyboard` to read a double value, store it in `seconds`
6. Compute the conversion of mps to kph
7. Ask `theKeyboard` to display `kilometersPerHour` along with descriptive text

Coding and Testing

- Translate algorithm into syntax of Java
  - Note Figure 1.4
- Compile the program – compiler checks for syntax errors
- Run the program with sample values
  - Observe whether the results are reasonable
  - Calculate results by hand to see if there is agreement
  - Try a variety of times
- Run-time errors can be tracked down with a debugger
  - Executes the program one line at a time

Maintenance

- Requirements of program may change
  - Client wants different kind of output
  - Client sees ways of expanding program to do more
  - Some other entity (government) requires a different format or additional results

Part of the Picture: Computer Ethics

- Computers permeate every aspect of our lives
- They perform life-critical tasks
- Yet computer science is not regulated to the extent of medicine, air travel, or construction zoning
- Much thought should be given to issues of ethics

Computer Crime & Security

- Some crimes are high tech versions of low tech problems (theft, fraud, child porno)
- Viruses and “trojan horses”
- Hackers try to get into restricted system
- Some solutions
  - effective use of passwords
  - antiviral software
  - firewalls
  - physical security

Health Concerns & the Environment

- People who spend too long at a computer and get too little exercise
- Ergonomic issues
  - radiation, eye strain, repetitive motion damage

- Internet addiction
- Disposal of old computer parts

Information Ownership

- Illegal software copying (pirating)
- Infringement copyright by copying of pictures or text from web pages
- Plagiarism by copying text from other sources when original work is expected

“Netiquette” and Hoaxes

- Inflammatory interchange of messages via internet (email, chat rooms, etc.)
- Chain mail
- Virus warning hoaxes
- “Spam” – unsolicited, bulk email

Internet Content & Free Speech

- Information on internet includes hate, violence, harmful information for children
- How much of this should be regulated
- Do filters solve problems or create more
- How reliable are web sites used for course work and research

Privacy

- U.S. Constitution, Amendments, and laws specify certain levels of privacy
- Databases containing personal information are easily stored, transmitted, and often available
- Does an employer have a right to monitor email messages
- Procedures and policies should be put in place and used by computer professionals

Quality Control & Risk Reduction

- Good software is difficult to produce
- It must be carefully designed, developed, tested
- Mistakes generated by computers can be far reaching
- Commenting and documenting software is required for effective maintenance throughout the life of the program
- Y2K issues highlighted the need for thinking ahead

The Future

- Telecommuting
- Distance learning
- E-commerce
- Information availability

- Also … hazards