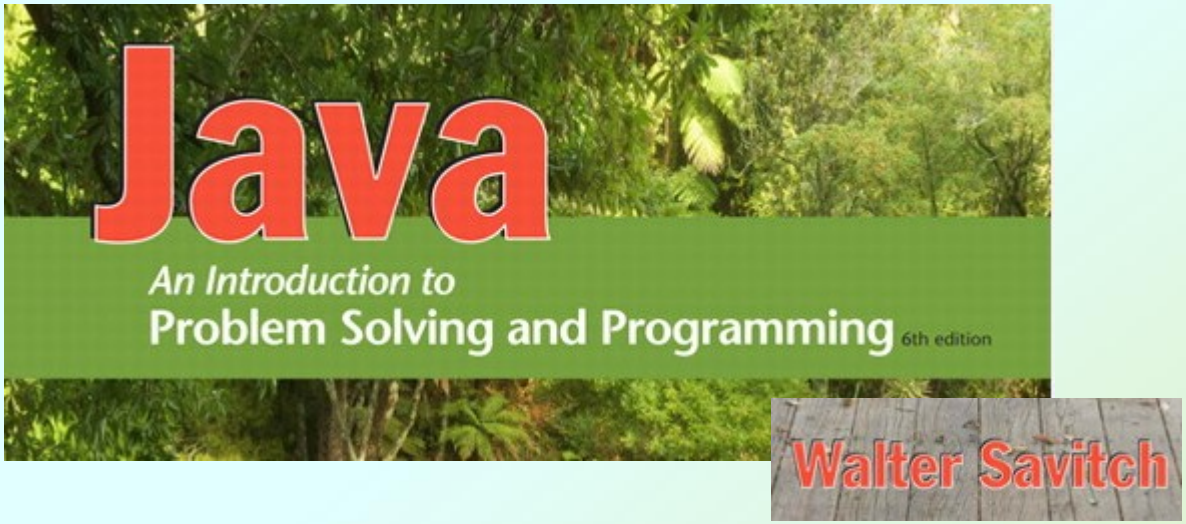


Introduction to Computers and Java

Chapter 1

*JAVA: An Introduction to Problem Solving & Programming, 6*th Ed. By Walter Savitch
ISBN 0132162709

A First Java Application

- View sample program Listing 1.1
  - **`class FirstProgram`**

```
Hello out there.
I will add two numbers for you.
Enter two whole numbers on a line:
12 30
The sum of those two numbers is
42
```

Sample screen output

Some Terminology

- A *package* is a library of classes that were previously defined.
  - `import java.util.Scanner;`
- The item(s) inside parentheses are called *argument(s)* and provide the information needed by methods.
- A *variable* is something that can store data.
- An instruction to the computer is called a *statement*; it ends with a semicolon.
- The grammar rules for a programming language are called the *syntax* of the language.

Printing to the Screen

```
System.out.println ("Whatever you want to print");
```

- `System.out` is an object for sending output to the screen.
- `println` is a method to print whatever is in parentheses to the screen.

Printing to the Screen

- The object performs an action when you *invoke* or *call* one of its methods

```
objectName.methodName(argumentsTheMethodNeeds);
```

Compiling a Java Program or Class

- A Java program consists of one or more classes, which must be compiled before running the program.
- You need not compile classes that accompany Java (e.g. `System` and `Scanner`).
- Each class should be in a separate file.
- The name of the file should be the same as the name of the class.

Compiling and Running

- Use an *IDE* (integrated development environment) which combines a text editor with commands for compiling and running Java programs.
- When a Java program is compiled, the byte-code version of the program has the same name, but the ending is changed from `.java` to `.class`.

Compiling and Running

- A Java program can involve any number of classes.
- The class to run will contain the words

```
public static void main(String[] args)
```

somewhere in the file

Programming Basics: Outline

- Object-Oriented Programming
- Algorithms
- Testing and Debugging
- Software Reuse

Programming

- Programming is a creative process.
- Programming can be learned by discovering the techniques used by experienced programmers.
- These techniques are applicable to almost every programming language, including Java.

Object-Oriented Programming

- Our world consists of *objects* (people, trees, cars, cities, airline reservations, etc.).
- Objects can perform *actions* that affect themselves and other objects in the world.
- Object-oriented programming (*OOP*) treats a program as a collection of objects that interact by means of actions.

OOP Terminology

- Objects, appropriately, are called *objects.*
- Actions are called *methods.*
- Objects of the same kind have the same *type* and belong to the same *class.*
  - Objects within a class have a common set of methods and the same kinds of data
  - but each object can have it's own data values.

OOP Design Principles

- OOP adheres to three primary design principles:
  - Encapsulation
  - Polymorphism
  - Inheritance

Introduction to Encapsulation

- The data and methods associated with any particular class are encapsulated ("put together in a capsule"), but only part of the contents is made accessible.
  - Encapsulation provides a means of using the class, but it omits the details of how the class works.
  - Encapsulation often is called *information hiding.*

Accessibility Example

- An automobile consists of several parts and pieces and is capable of doing many useful things.
  - Awareness of the accelerator pedal, the brake pedal, and the steering wheel is important to the driver.
  - Awareness of the fuel injectors, the automatic braking control system, and the power steering pump is not important to the driver.

Introduction to Polymorphism

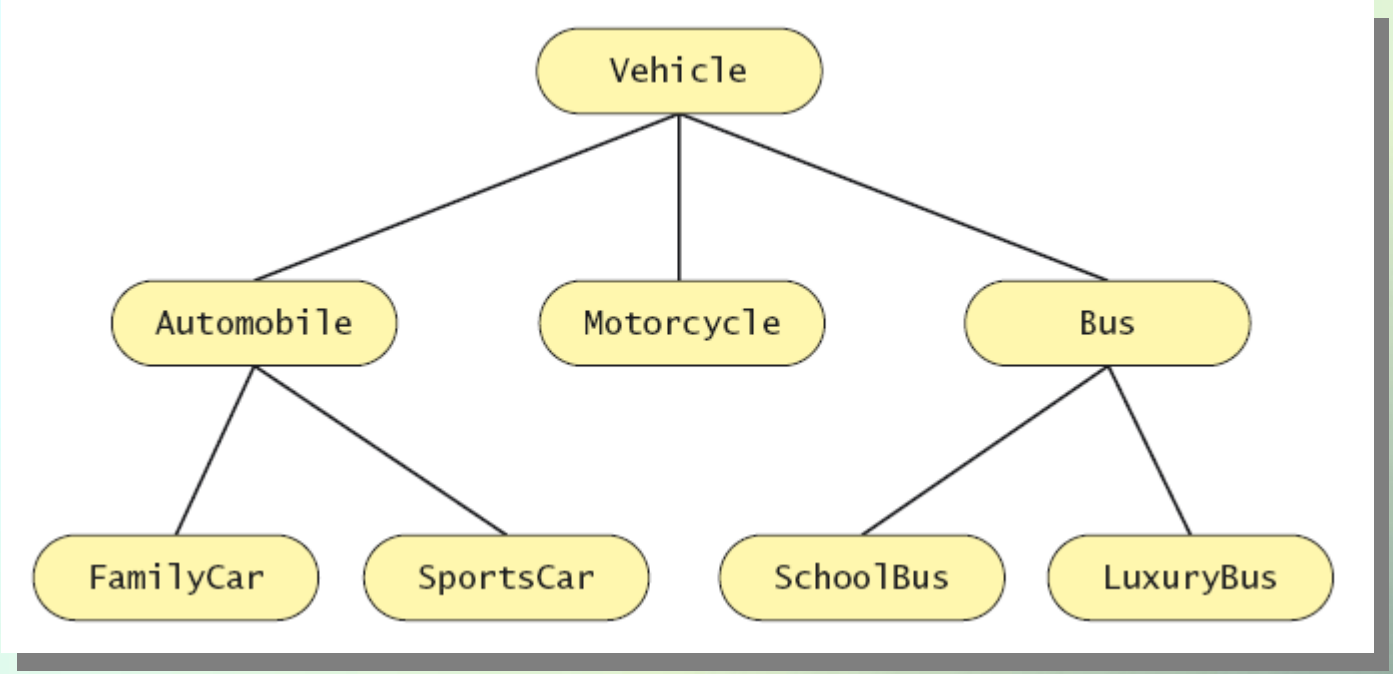
- From the Greek meaning “many forms”
- The same program instruction adapts to mean different things in different contexts.
  - A method name, used as an instruction, produces results that depend on the class of the object that used the method.
- More about polymorphism in Chapter 8

Introduction to Inheritance

- Classes can be organized using *inheritance.*
- A class at lower levels inherits all the characteristics of classes above it in the hierarchy.
- At each level, classifications become more specialized by adding other characteristics.
- Higher classes are more inclusive; lower classes are less inclusive.

Introduction to Inheritance

- Figure 1.4



Inheritance in Java

- Used to organize classes
- “Inherited” characteristics do not need to be repeated.
- New characteristics are added.
- More about inheritance in chapter 8

Algorithms

- By designing methods, programmers provide actions for objects to perform.
- An *algorithm* describes a means of performing an action.
- Once an algorithm is defined, expressing it in Java (or in another programming language) is usually is easy.

Algorithms

- An algorithm is a set of instructions for solving a problem.
- Algorithms usually are expressed in English or in *pseudocode.*
- *An algorithm must be expressed completely and precisely.*

Example: Hair Washing

- Lather
- Rinse
- Repeat

Imprecise!

Infinite loop

Errors

- An error in a program is called a *bug.*
- Eliminating errors is called *debugging.*
- Three kinds or errors
  - Syntax errors
  - Runtime errors
  - Logic errors

Syntax Errors

- Grammatical mistakes in a program
  - The grammatical rules for writing a program are very strict
- The compiler catches syntax errors and prints an error message.
- Example: using a period where a program expects a comma

Runtime Errors

- Errors that are detected when your program is running, but not during compilation
- When the computer detects an error, it terminates the program and prints an error message.
- Example: attempting to divide by 0

Logic Errors

- Errors that are not detected during compilation or while running, but which cause the program to produce incorrect results
- Example: an attempt to calculate a Fahrenheit temperature from a Celsius temperature by multiplying by 9/5 and adding 23 instead of 32

Summary

- You have completed an overview of computer hardware and software.
- You have been introduced to program design and object-oriented programming.
- You have completed an overview of the Java programming language.