Proyecto Access

Chicago PREP

NASA - HACU

Richard J. Daley College

Chicago PREP Curriculum

COMPUTER SCIENCE WITH JAVA

Writing Faculty
Dr. Vali Siadat, Co-Principal Investigator
Dr. Du Zhang
Dr. Kecheng Zhou
Dr. Younker Chyu

Produced and Distributed by:
CHICAGO PREFRESHMAN ENGINEERING PROGRAM (PREP)

This project was funded by:
The Hispanic Association of Colleges and Universities (HACU) and the National Aeronautics and Space Administration (NASA)

Foreword

This note introduces you to the field of computer science. It is intended for those who need to get a preliminary exposure to personal computers and their usage. Upon completing the introduction to computer science curriculum, you should be able to understand what a computer is, how it operates and also know about different programming languages.

In this note, we introduce you to a modern programming language called Java. This is an objective-oriented programming language for writing programs on various computers connected to the Internet.

In studying the computer science with Java, we hope you are challenged to use your logical and analytical thinking skills to write programs and solve problems using the state of the art Pentium computers available to you.

Table of Content

UNIT 1. INTRODUCTION

Objective: The student will learn how the course is to be structured and how grades will be determined.

Prerequisites: None.

Materials:

Contents:

Give instructor's name, office hours and location.

Topics to be covered in the course:

- What is a computer? What does a computer do?
- Why is a computer so powerful?
- Computer concepts and terminologies.
- Computer hardware and software.
- What are Windows systems? What do they do?
- Internet and E-mail.
- Computer languages.
- Writing programs in Java.
- Solving problems on a computer.

Grading scheme:

Lab (Programs)	30%	A – 93 ~ 100
Notebook	10%	B – 85 ~ 93
Work sheet and quizzes	10%	C – 75 ~ 84
Final exam	50%	D – 70 ~ 74
		F – Below 70

Requirements:

- Notebook
  - Contents:
  - Class notes
  - Work Sheets and Quizzes
  - Programs
  - Handouts

- Programs to be developed in class and entered on the computer.
- Programs to be written by students on their own.
- Supported materials to be turned in with programs.
- Homework.
- Quizzes and tests.
- Special projects for advanced students.

Logistics:

- When and how often students will use lab.
- Location of lab.
- Will they work in partners or alone?
- Importance of using the lab for work, not play.
- Pass out rules for lab.
- Instructions for using the computer. (Windows Systems)

Homework assignment:

Lab Assignment: (given by the instructor)

UNIT 2. AN OVERVIEW OF COMPUTER CONCEPTS

Objective: The student will learn the definition of a computer and its parts and peripherals.

Prerequisites: None.

Materials:

Content:

2.1. What is a computer?

A computer is an electronic device operating under the control of instructions stored in its own memory unit that can accept data (input), process data arithmetically and logically, produce output from the processing, and store the result for further use, without intervention of a human being.

In short, a computer is a collection of electronic devices that function together to process data. An example of such a collection that makes up a computer consists of system unit, monitor, keyboard, and mouse.

2.2. What does a computer do?

Whether small or large, computers can perform four general operations. These operations comprise the information processing circle and are: input, process, output, and storage.

All computer processing requires data. Data refers to the raw facts, including numbers, words, images, and sounds, given to a computer during the input operation. In the processing phase, the computer manipulates the data to create meaningful and useful information. During the output operation the information created is put into some form, such as a printed report or a visible format that people can use. The information can also be stored electronically for future use.

In order for a computer to process data, it must contain two parts: the physical parts (i.e., electronic devices) of the computer and the programs (i.e., instructions) that enable the computer processing. The physical parts of a computer are called hardware and the programs are called software.

2.3. What are the components of hardware?

There are four types of computer hardware, classified according to their functions.

Central processing unit (CPU)

The CPU is the unit that processes the data. It is made up of two units:

- The control unit, which controls the flow of data through the computer.
- The Arithmetic Logic Unit (ALU), which performs the arithmetic calculations and logic operations.

Main Memory (often simply called memory).

This is where the data and programs are stored. It is made up of two parts:

- Read-Only Memory (ROM). ROM is used to store a `simpler' program that is needed to boot up the computer. This program cannot be changed and are not lost when the computer's power goes off.
- Random Access Memory (RAM). RAM is used to store data being processed. These data are being changed rapidly from time to time during the computer processing. The data and programs stored in RAM are lost if the computer's power goes off.

Auxiliary Memory (also called ``Auxiliary Storage'')

This is where data and programs are stored on a more permanent basis. Loss of power does not usually destroy programs and data that are stored in auxiliary memory. The most common auxiliary devices are floppy disks, hard disks, and tapes.

Input/Output devices (I/O Devices)

Input devices are those used to enter data into a computer. Two common input devices are keyboard and mouse. As the data is entered on the keyboard, it is temporarily stored in the computer's memory and displayed on the screen of the monitor. A mouse is a type of pointing device used to select processing options or information displayed on the screen. Other input devices are microphone, scanner, light pen, track ball, etc.

Output devices are those used to receive results from the computer. The two most commonly used output devices are the printer and the monitor. Others are speaker, plotter, robot, etc.

Assignment

1. What CPU is in your computer?
2. What is the RAM memory capacity in your computer?
3. What is the storage capacity of your hard-drive? How much storage space does a standard double-side high-density floppy disk have?
4. Please name two input devices and two output devices connected to your computer.

UNIT 3. WINDOWS ENVIROMENT AND WIMP INTERFACE

Objective: The student will be exposed to the basic components of the Windows operating system and the window/icon/menu/pointing interface.

Prerequisites: Knowledge of a computer system.

Materials:

Contents:

3.1. Desktop

There are a number of icons on your desktop:

- **My Computer**: double-click this icon to see your computer's contents and manage your files.
- **Network Neighborhood**: double-click this icon to see available resources on the network, if your computer is or can be connected to one.
- **Recycle Bin**: this is a temporary storage place for deleted files. You can restore files deleted in error or empty the bin to actually discard the files.
- **Start button**: you can click the Start button on the taskbar to start a program, open a document, change system settings, get help, find items on your computer, shut down your computer, and more.
- **Taskbar**: you can use the taskbar to switch between open windows

3.2. Basics of Windows Systems

Windows systems (i.e., Windows 95/98/2000) offer the window/icon/menu/pointing interface through which various user tasks can be carried out.

When you click the Start button, a menu pops up that contains everything you need to begin using Windows.

Every time you start a program or open a window, a button representing that window appears on the taskbar. To switch between windows, just click the button for the window you want. When you close a window, its button disappears from the taskbar.

After you start a task, it runs in a task window on the desktop. Each window has three buttons in the upper-right corner: minimize, maximize and close. You can reduce or enlarge an open task window through minimize or maximize buttons, and quit the program through close button.

There are a number of ways to open documents in Windows. You can:

- Open your document from within the program you used to create it.
- Use the Documents command on the Start menu.
- Use the Find command on the Start menu.

Online help is available for a specific procedure and for information about what you see on your screen.
General help information is available in Help item on the Start menu, or the Help menu in My Computer or Windows Explorer
The Help menu in a program (such as WordPad, Paint, Word, etc.) provides the help information for that particular program.

By selecting the Settings item on the Start menu, you can change the way Windows looks and works. You can use Control Panel under the Settings to change your screen colors, install or change settings for hardware and software.

You need to shut down Windows before you turn off or restart your computer. Use the Shut Down command on the Start menu.

3.3. Additional Windows Features

My Computer

Everything you have on your computer (your programs, documents and data files) is accessible from one place called My Computer, an icon from the desktop.

Windows Explorer

Offers another way of seeing what is on your computer. You can open it by clicking the Start button, pointing to Programs and clicking Windows Explorer, or by using the right mouse button to click My Computer icon and click Explore.

Files and Folders Organization

Using either My Computer or Windows Explorer, you can organize your files and file folders. You can perform the following:

- To move or copy a file/folder
- To delete a file/folder
- To create a new folder
- To copy a file to a floppy disk.

Working with Documents

You can use certain programs (WordPad, Word, etc.) to work with documents. After editing your document (copying, moving, deleting), you can save changes to an existing document.

Printing

You can set up a printer for use with Windows by clicking the Start button, pointing to Settings, and then clicking Printers. In Printers folder, double-click Add Printer and follow the instructions on the screen.

After a printer is set up, documents can be easily printed out (if the document is open, click File and then click Print).

Installing Software and Hardware

Windows systems offer a convenient way to install/remove software and hardware. By selecting Add/Remove Programs or Add New Hardware icons in the Control Panel of Settings under Start menu, you can easily add a new software or hardware to your Windows environment.

Assignments

1. When you open a file or activate a program in the Windows environment, they will be presented to you as an individual window. They may also cover the other windows. How can you tell the number of windows that are currently open?

2. What is the difference between "close" and "minimize" a window?

3. What are the differences between "My Computer" and "Windows Explorer" in their appearance and usages?

4. How can the Windows system distinguish different types of files in your computer?

5. When you right-click the mouse, a menu will usually come up at the tip of the mouse cursor. Is it always the same?

6. How can you select multiple files? How can you select multiple folders?

UNIT 4. INTERNET AND WORLD-WIDE WEB

Objective: The student is expected to learn the components of Internet/WWW, use of Internet/WWW for communication and information retrieval.

Prerequisites: Knowledge of a window-base operating system.

Materials:

Contents:

4.1. Internet

The Internet is a worldwide network of computer networks. It is very difficult to give numbers for the size of the Internet because of its rapid growth. Some of the services and resources available on the Internet include the following:

- **Electronic mail:** E-mail can be sent/received anywhere in the world between users with an Internet address.
- **File transfer**: files can be transferred between host computers anywhere in the world.
- **Remote login**: you can connect to and use a remote computer anywhere on the Internet.
- **Database search**: there are hundreds of databases online and available for review. Internet offers several tools for finding information on a particular subject.
- **Electronic magazines**: a variety of journals and magazines are published electronically on the Internet that are accessible to users.
- **Special interest groups**: there are numerous discussion/news groups, mailing lists, talk/chat facilities and bulletin board systems on the Internet that are organized by users sharing a common interest in an area.
- **White page directories**: services are provided to look for an Internet user's address.
- **Entertainment:** there are a variety of games available on the Internet.

Communication through E-mail

- Connection to the Internet (telephone line, modem)
- Internet addressing (domain, sub-domain, userid)
- How to compose an E-mail
- How to send an E-mail
- How to receive an E-mail

- How to save a received E-mail to a folder

4.2. World Wide Web (WWW)

The World Wide Web (WWW) is defined as the universe of global network-accessible information. It is an abstract space within which people can interact, and it is largely populated by inter-linked pages of text, images, and animations, with occasional sounds, videos, and three-dimensional worlds.

The WWW technology consists of the following elements:

- **Universal resource locator (URL):** a convention for information naming and linking. If you have your home page on the WWW, there is a URL associated with your home page. People can find your home page according to the URL.

- **Hypertext markup language (HTML):** a text-based language for information rendering (hypertext: data that contain links to other data). You can use HTML to write your home pages for review by other people on the WWW.

- **Hypertext Transfer Protocol (HTTP):** a client-server protocol to transport information associated with a URL.

- **Web browser:** a program that displays HTML documents, provides a storage space for URLs, and supports a URL directory. Example of browsers include: Netscape and Internet Explorer.

- **Web server:** a computer system that responds to requests for information from the Web browsers.

Surfing the WWW

- Open a page according to its URL
- Place a bookmark at a web-site
- Search for web sites on a subject topic using some search engine (e.g., Yahoo)
- Print information from web site
- Take a look at the resource file of a home page
- Edit your bookmark file

Assignments

1. What is the difference or relationship between a computer network system and the Internet?

2. "Windows" is an operating system used on many computers connecting to the Internet. Please name two more computer operating systems.

3. What is the name of your Web browser? Which version is it?

4. How can you search information over the Internet? Please give a URL where we may do so.

5. Do you have an e-mail account? What is your e-mail address? Please send an e-mail to your teacher.

UNIT 5. PROGRAM DEVELOPMENT

Objective: The student is expected to learn the steps of how to write a computer program for a given problem.

Prerequisites: Knowledge of a computer system.

Materials:

Contents:

5.1. What Is A Computer Program?

A computer program is a detailed set of instructions that directs a computer to perform the tasks necessary for the solution of a given problem.

A computer program is stored in a computer's memory. Instructions of a program are fetched from memory to the CPU in order for the specified tasks to be carried out. A program is written by a computer programmer who codes the instructions using a programming language. To create programs that are correct (produce accurate results for a given problem) and easy to modify, programmers follow a process called program development.

5.2. What Is Program Development?

Program development is the process of producing quality programs to be run on a computer. The process of program development has five steps:

1. **Review specification:** the programmer reviews the specifications (what the problem is, what the program is going to do to solve the problem, what is the user's requirement for problem solution, etc.).

2. **Design:** the programmer decides and documents the course of actions the computer will take to accomplish the desired tasks.

3. **Code**: the programmer writes the actual program instructions in terms of some programming language.

4. **Test:** the written programs are tested to make sure they perform as intended.

5. **Finalize documentation:** the programmer documents explanatory information about the program throughout the development process. All the documentation produced during the first four steps is brought together and organized.

High-level languages

High-level languages contain nouns, verbs, and mathematical, relational and logical operators that can be used to define program statements, thus closely resembling a natural language. It is easier for the programmer to learn and use high-level languages than machine or assembly languages. In addition, these languages are machine independent in the sense that programs written in high-level languages can run on different sorts of computers.

Compiler and interpreter

Programs written in a high-level language must be translated into their machine language equivalents before they can be executed. Such a translation can be done in one of two ways: through a *compiler* or an *interpreter*.

- A compiler translates an entire program into a set of machine language instructions that is stored on a disk for later execution. The program to be translated is called the *source program* and the set of machine language instructions is called the *object program*. Compilers check programming errors and report to the programmer a listing of all program statements and definitions not consistent with syntactic rules and semantics of the programming language used.

- An interpreter translates one program statement at a time and then executes the resulting machine language instruction(s) before translating the next program statement. No object program will be generated by the interpreter.

6.3. Types of Programming Languages

Depending on the problem solving strategies used in the program development, programming languages can be classified into the following:

- Functional programming languages (Lisp).
- Logical (relational) programming languages (Prolog).
- Object-oriented programming languages (C++, Java).
- Parallel and concurrent programming languages (Concurrent Pascal).

Assignments

1. Where is a computer program stored in your computer when it is not used? In what form is it stored?

2. How does a computer system handle a program when you want to use it?

3. When we are learning the rules of a programming language, we are learning the "_____" of that language.

4. How can the computer understand a program written in a high-level programming language?

This process is iterative in the sense that at any particular step, program development usually requires returning to previous step(s) to correct errors made earlier.

5.3. Tools and Methodologies

There are tools and methodologies available for program development.

Structured program design:

Structured program design is a methodology consisting of three main program design concepts: modules, control structures and single entry/single exit. Use of these concepts helps create programs that are easy to write, read, understand, check for errors and modify.

Modules:

With structured design, programming problems are decomposed into smaller parts called modules. Each module performs a given task within the program. The major benefit of this technique is that it simplifies program development because each module of a program can be developed individually. When the modules are combined, they form a complete program that accomplishes the desired result.

Control structures:

Three basic control structures are used to form the logic of a program:

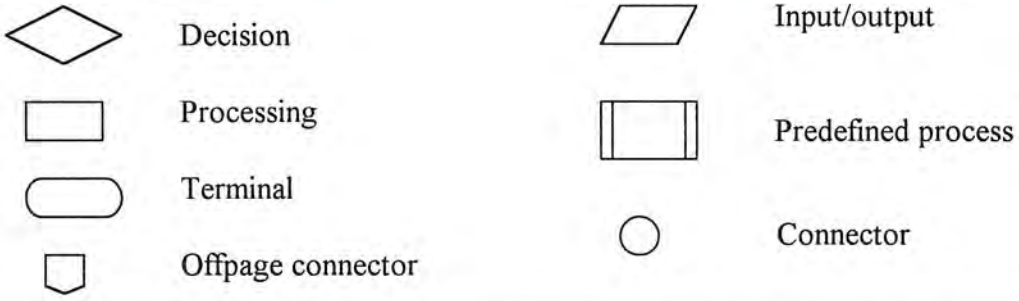
- **Sequence structure:** one action occurs immediately after another.
- **Selection structure:** offers a way to represent conditional program logic (if-then-else, case).
- **Iteration structure:** provides a means in which one or more actions are repeated until a given condition is satisfied (for-loop, while-loop, do-until).

Single entry/single exit

This concept means that there is only one entry point and one exit point for each of the three control structures. An entry point is one where a control structure is entered. An exit point is one where the control structure is exited.

Flowcharts

Program flowcharts were one of the first program design tools. In a program flowchart, all the logical steps of a program are represented by a combination of symbols and text.



Assignments

1. Where is a computer program stored in your computer when it is not used? In what form is it stored?

2. How does a computer system handle a program when you want to use it?

3. What are the steps to develop a computer program?

4. What are the three basic control structures?

5. Assume that "x", "y", and "z" represent three numbers. Draw a flow chart to show the followings:
   - If ($x \geq y$) then (z is equal to x)
   - If ($x < y$) then (z is equal to y)

6. Based on Exercise 5, what will z become if:
   - $x = 10$ and $y = 5$?
   - $x = 4$ and $y = 8$?

5. What is the difference between a compiler and an interpreter?

UNIT 6. PROGRAMMING LANGUAGE

Objective: The student is expected to learn the basic concepts and components of programming languages.

Prerequisites: Knowledge of program development process.

Materials:

Contents:

6.1. What Is A Programming Language?

A programming language is a set of written words and symbols that allow the programmer or user to communicate with the computer.

- **Syntax:** Like any spoken language, programming languages have rules called *syntax* that governs their use.
- **Semantics:** The meanings of language constructs in a programming language are referred to as *semantics*.

6.2. Levels of Programming Languages

There are hundreds of programming languages available, each having its own syntax and semantics. In general, programming languages can be classified into the following categories:

Machine language

Machine language is the fundamental language understood by a computer's processing unit (CPU). Programs written in all other languages are eventually translated into machine language before they can be executed. Individual machine language instructions correspond directly to the commands in the computer's instruction set. Because the instruction set is unique for a particular processing unit of a computer, machine languages are different for computers that have different processing units.

The advantage of writing a program in machine language is that the programmer has direct control over the computer and accomplishes exactly what needs to be done. The disadvantages of machine language programs are (1) it takes a long time to write; (2) it is difficult to review in order to locate an error; (3) the programs are machine dependent.

Assembly language

An assembly language uses *mnemonics* (abbreviations) or *symbolic operation codes* to represent the machine instructions and memory locations. Assembly language programs are converted into machine language instructions by a special program called an assembler.

UNIT 7. JAVA: GETTING STARTED

Objective: The student will learn how to get started with Java programming.

Prerequisites: Knowledge of basic programming concepts.

Materials:

Contents:

7.1. Brief Background

The Java programming language was developed by Sun Microsystems for writing programs to run on different computers connected to the Internet. Programs written in Java (called **applets**) can be downloaded through a web browser as part of a web page, and run on a user's computer. There are a number of salient features of the language:

In addition to applets, the Java language can be used to develop stand-alone applications that do not involve the use of web pages.

Java is an **object-oriented (OO) programming** language. From a programmer's perspective, OO programming implies that the focus in programming activities is on building classes to represent the data and their operations in an application, rather than thinking of a solution to a problem in terms of a set of procedures that must be executed in certain order. As a result, Java programs consist of class definitions.

The portability of Java (programs runnable on a variety of hardware platforms) comes from the fact that the Java compiler translates a program into its **byte code** that is neutral to the instruction set of any particular computer. The byte code representation of a Java program has to be translated by a **Java virtual machine** into a machine specific executable for execution.

The Java virtual machine has features that allow for secure and robust program behaviors. It also supports networking applications.

The Java language is simple to learn and to use. Its syntax is largely based on that of popular C and C++ programming languages. It has an extensive library of software components that a programmer can use

7.2. Java Development Kit

The **Java Development Kit (JDK)** by Sun Microsystems is a collection of programs to help developers compile, run and debug Java programs. JDK can be downloaded from Sun Microsystems free of charge.

There are two tools in JDK that will be used extensively throughout the remaining units:

- **javac**: the Java language compiler to be used to produce the byte code for a Java program.
- **java**: the Java interpreter to be used to run Java programs.

7.3. How to Compile and Run A Simple Java Program

Create a Java program

To create a Java program, an editor is needed. The files containing Java source programs must be text files and have a file extension of ".java". For example, the Notepad program in the Windows Environment may be used in creating such files.

Example

Use Notepad to type the following lines

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

Save the file with the name "HelloWorld.java". Make sure the file type must be "*.java*". Open a DOS command window and use "type" command to verify that the file contents are exactly the same as above.

Compile the program

Now compile the example program in the DOS window by using the following command:

javac HelloWorld.java

The result of the compilation is the creation of a new file called "HelloWorld.class". This is the byte code file for the source code file "HelloWorld.java". The Java virtual machine system will use "HelloWorld.class" to run the program.

Run the program

To run the program, use the following command

java HelloWorld

The execution of the program will produce the following message on your screen.

"Hello, World!"

Document the program

Program documenting refers to adding comments to your programs. There are two ways to write comments in Java

- A C style comment begins with the symbol pair /* and ends with */
- A C++ style comment begins with // and ends with the end of the text line.

7.4. Analysis of the Example Program

Java is a case-sensitive language. Therefore, HelloWorld is different from helloworld.

The first line defines a class called HelloWorld.

- The first keyword **public** indicates the accessibility of the HelloWorld class.
- The second keyword **class** indicates that a class is about to be defined.

The second line defines a method called main in the HelloWorld class

- The first keyword **public** indicates the accessibility of the method
- The second keyword **static** indicates that the **main** method is a **class** method rather than an **instance** method
- The third keyword **void** indicates that the main method returns no value
- All Java programs contain one or more class definitions, each of which may contain various method definitions. In general, a method's body contains a sequence of one or more statements.
- Every standalone Java program must contain a class definition that defines a method called main. When the program execution starts, it starts with the computations specified in main.

The third line is the body of the main method and contains a single statement that invokes a method called **println** of an object called **System.out** that belongs to the class where println is defined. The purpose of invoking println is to print the string "HelloWorld".

Assignments

1. JAVA is an object-oriented (OO) programming language. What does "object-orient programming" mean? Please explain it in your own words.

2. For portability, what is the process in terms of "byte code" and "Java virtual machine" so that we can use the same Java programs on different computer systems?

3. What is JDK? Why do we need to use "javac" and "java" commands?

4. Four steps were given as an example to successfully make a Java program. What are those steps?

5. In accord to the example program in this chapter:

    - How can we have a class accessible to everyone?
    - How do you indicate the starting of a class definition?
    - How do you identify a method belongs to a class?
    - How do we indicate a method that is not returning any value?
    - From where does a Java program start when it is executed?

6. Please make a hardcopy of the result generated by the example program in this chapter.

UNIT 8. PROGRAMS WITH INPUT

Objective: The student is expected to learn how to write Java programs that handle input.

Prerequisites: Knowledge of basic program input/output.

Materials:

Contents:

8.1. Interactive Input

Input is more error-prone than output. A program will fail abruptly if it receives the incorrect input. Such a run-time error is called an **exception**. Java offers special mechanisms for handling exceptions. The simplest exception handling mechanism is to append the clause "**throws IOException**" to the declaration of the main method.

Example:
Use Notepad to create the following program source code

```
import java.io.*;
public class HiThere
        {
        public static void main(string[] args) throws IOException
                {
                InputStreamReader reader = new InputStreamReader(System.in);
                BufferedReader input = new BufferedReader(reader);
                System.out.print("Enter your name: ");
                String name = input.readLine();
                System.out.println("Hi there, " + name + "!");
                }
        }
```

Save the file with name "HiThere.java", compile the program, and then run it. The program prints the following prompt:

Enter your name:

And then waits for input. After you type

John Doe

and press the Enter key, the program prints the message below:

Hi there, John Do!

Analysis:

There are five objects in the program through which the input from the keyboard is used as part of the message to be displayed on your monitor. These five objects are: **System.in**, **reader**, **input**, **name**, and **System.out**. The System.in and System.out objects are defined in the System class. The other three objects are defined in the program.

Every object used in a Java program must be declared before it is used. The declaration may optionally include an initialization.

The first line tells javac compiler to look in the java.io library for the definitions of the three I/O classes that are used in the program: IOException, InputStreamReader, and BufferedReader.

The fourth line defines the object reader to be an instance of the InputStreamReader class, binding it to the system input stream System.in. This means that the object reader will serve as a conduit, conveying data from the keyboard into the program.

The fifth line defines the object input to be an instance of the BufferedReader class, binding it to the reader object. This indicates that the object input can be used to extract input in a desirable way.

In the seventh line, the String object name is defined and initialized with the string that is returned by the input.readLine() method. The result is that the name object contains whatever you typed at the keyboard.

In the eighth line, the expression *"Hi there, " + name + "!"* means to concatenate the three strings "Hi there, ", name, and "!" to form a single string to be sent to the display.

8.2. Numeric Input

When the input is numeric, rather than alphabetic, different storage and processing will be needed. We will examine the following program to see how numeric input is handled.

Example:

```
import java.io.*;
public class YearOfBirth
        {
        public static void main(string[] args) throws IOException
                {
                InputStreamReader reader = new InputStreamReader(System.in);
                BufferedReader input = new BufferedReader(reader);
                System.out.print("Enter your age: ");
                String x = input.readLine( );
                int age = new Integer(x).intValue( );
                System.out.println("You are " + age + " years old now,");
```

```
        int year = 1999 – age;
        System.out.println("so you were born in" + year);
        }
}
```

Analysis:

Like the previous program, this one also imports class files from the java.io library.

Variables containing integer values are used in the program. "int" is used to declare an integer variable. After the program reads the input as a string, it converts x into an integer by the expression:

new Integer(x).intValue( )

The integer value is used to initialize the int variable age. The program prints that value, computes the user's year of birth from it, and then prints that year.

Assignments

1. What is the most basic way to handle an input/output error?

2. What are the methods/commands to:

3. Print an output?
4. Input one line of information?

5. How do you specify a string variables and a constant string?

6. Write a small program to input your full name and student ID.

UNIT 9. VARIABLES, OBJECTS AND OPERATORS

Objective: The student is expected to learn the variables, objects and operators in this unit.

Prerequisites: Knowledge of basic data types.

Materials:

Contents:

9.1. Variables and Objects

There are two types of entities that hold data in Java: variables and objects. A variable has a **type** and holds a single **value**. There are nine possible types that variables can have. Every variable has a unique **name**, which is designated when it is declared. A variable is created when it is declared, and it remains alive until the method in which it is declared terminates.

On the other hand, an object is an instance of a class and may contain many variables, the composite of whose values is called a state of the object. Because programmers can define their own classes, an unlimited number of objects may be instantiated out of classes. Objects have **references** instead of names. An object is created by using the **new** operator to invoke a constructor, it terminates when it has no references.

Example

```
import java.io.*;
public class Area
        {
        public static void main(string[] args) throws Exception
                {
                InputStreamReader reader = new InputStreamReader(System.in);
                BufferedReader input = new BufferedReader(reader);
                System.out.print("Enter the radius: ");
                String text = input.readLine( );
                Double x = new Double(text);
                double r = x.doubleValue( );
                System.out.println("The area of a circle of radius " + r);
                double area = Math.PI*r*r;
                System.out.println(" is " + area);
                }
        }
```

This program uses two variables (r and area) and five objects (reader, input, text, x and System.out). Both variables have the type of *double*. The objects are instances of the classes

UNIT 10. SELECTION

Objective: The student is expected to learn how to use if, if-else and switch statements to write programs that select alternative computations based on the value of some logical expression. In addition, the conditional operator is also explained.

Prerequisites: Knowledge of logical expressions.

Materials:

Contents:

10.1. *if* Statement

The if statement allows for conditional execution. Depending on the outcome of the condition, actions included in the if statement may or may not be executed. The syntax of the statement is as follows:

if (condition) statement

If condition is true, then statement is executed.

Example:

```
import java.util.Random;
public class NegativityTest
        {
        public static void main(string[] args)
                {
                Random random = new Random( );
                int n = random.nextInt( );
                System.out.println("n = " + n);
                if (n < 0) System.out.println("***** n is negative.");
                System.out.println("So long.");
                }
        }
```

This program uses a random number generator to randomly create an integer and then tests if the number is negative or not. The Random class is imported in the first line and is used to instantiate an instance of the class called random in line four. By calling its method random.nextInt( ), a random integer is created. The message "***** n is negative." Will be printed out only when the condition of n < 0 is satisfied.

InputStreamReader, *BufferedReader*, *String*, *Double*, and *PrintStream*, respectively. Technically, *reader* and *input* are the names of **references** to the objects.

9.2. Primitive Data Types

The nine data types are as follows:

- **reference**: a variable of reference type is "reference to *Xxxx* class," where *Xxxx* is the name of some class.
- **boolean**: a variable of this type has a value of either true or false.
- **char**: a variable of this type has as its values 16-bit Unicode characters.
- **byte**: a variable of this type has 8-bit integer values.
- **short**: a variable of this type has 16-bit integer values.
- **int**: a variable of this type has 32-bit integer values.
- **long**: a variable of this type has 64-bit integer values.
- **float**: a variable of this type has 32-bit floating-point numbers.
- **double**: a variable of this type has 64-bit floating-point numbers.

Syntactically, a variable must be declared before it can be used. A variable declaration is as follows:

type_name ***variable_name;***

where **type_name** is the name of the type and **variable_name** is the name of the variable having the type.

9.3. Operators

An **operator** is like a function that has a symbolic name, returns a value and is used in an expression. Following are some operators used in different types of expressions

- Operators in arithmetic expressions
  6 + 3, 6 – 3, 6 * 3, 6 / 3 (quotient), x / y, 5 % 3 (remainder), 6 + 5 * 2
- Assignment operators
  x = 9, x = (y = 4), x += 7, y –= 12, ++k, –-j, k *= j + 3

10.2. *if-else* Statement

The if-else statement is like the if statement, except that there is a second embedded statement that follows else:

```
if(condition)
        statement_i
else
        statement_j
```

We can have a statement like the following:

```
if (n < 0)
        System.out.println("***** n is negative.");
else
        System.out.println("***** n is positive.");
```

10.3. *switch* Statement

When you have to deal with more than two alternatives in your programs, you can use a multiway conditional statements switch.

```
switch (integer-producing expression)
        {
        case integer_constant_1 :   statements for integer 1;
                                    break;
        case integer_constant_2 :   statements for integer 2;
                                    break;
        .....
        default: default statements
        }
```

When a switch statement is encountered, the expression is evaluated, yielding an integer. The integer value is then compared with the integer constants found following the case keywords. Once there is a match, the statements in that case are executed until the first break statement that results in the control is transferred out of the switch statement.

The line beginning with the keyword default is optional. If the expression generates an integer that matches none of the case integer constants, the statements following default are executed (so as to act like catching all the other remaining alternatives).

For example, we can have the following:

```
switch (n)
        {
        case 0: System.out.println("***** n is zero.");      break;
```

- Operators in relational expressions (value returned is either true or false)
  k == y , j != 5, x > 10, y < 8, j >= k, j <= k

- Operators in logical expressions

and operator (&&):	10 < x && x < 20	(the value of x is between 10 and 20)
or operator (||):	10 < x || y > 20	(either x is greater than 10, or y is greater than 20)
not operator (!):	!(5 == 10),	not converts true to false and false to true

Assignments

1. How do you create an object? How do you specify a variable?

2. A numerical value has been stored as a string in *text*. How can you convert and store it into a floating-point variable *mynumber*?

3. What are the nine primitive data types?

4. What are the results of the following calculations if $x = 7$ and $y = 4$?

Expressions:	Results:
*z = ((x+y) * (x – y)) % 10;*	z = ?
*z = (x+y * x – y) % 10;*	z = ?
(x >= y)	?
(x > 5) && (y > 5)	?
(x < 5) || (y < 5)	?
++x; *-- y;*	x = ? y = ?
*z *= x + y;*	z = ?

```
case 2: System.out.println("***** n is one.");        break;
default: System.out.println("***** n is an integer other than zero or one.");
}
```

When there is no break or return statement to terminate the execution of a sequence of statements in a case, execution is said to **fall through** to the next sequence of statements in the subsequent case.

10.4. Conditional Operator

There is a special operator called conditional operator that can be used to evaluate one of the two alternative expressions

logical expression ? expression1 : expression2;

if logical expression returns a true, then expression1 is evaluated and its value returned, otherwise, expression2 is evaluated and its value returned.

Assignments

1. Draw a flow chart for the *if-else* example in section 10.2.

2. Write a program to input an integer and use the *if-else* statement to show that it is an even or odd number.

3. Change the *if*-else statement in your program of Exercise 2 with the *conditional operator*.

4. Write a program with the *switch* statement that:

5. User may input 3 choices for purchasing:
   a. A chair ($75.00)
   b. A table ($140.00)
   c. A book shelf ($99.00)
   d. Any other input will result in an error message

UNIT 11. ITERATION

Objective: The student is expected to learn how to use for and while statements to write programs that repeat some computation until a condition is satisfied.

Prerequisites: Knowledge of logical expressions.

Materials:

Contents:

11.1. *while* Statement

Java's while statement has the following syntax:

```
while (logical expression)
        {
        ...
        Embedded statement(s)
        ...
        }
```

Its semantics is as follows: the logical expression is evaluated. If its value is true, the embedded statements are executed; otherwise, Java skips the embedded statements. The **evaluate-execute loop** continues as long as the logical expression evaluates to true.

Example:

```
public class FibNumbers
    {
    public static void main(string[] args)
        {
        System.out.print(0);
        int fib0 = 0;
        int fib1 = 1;
        int fib2 = fib1 + fib0;
        while (fib2 < 1000)
            {
            fib0 = fib1;
            fib1 = fib2;
            fib2 = fib1 + fib0;
            System.out.print(", " + fib1);
            }
        }
```

```
}
```

This program will print out the sequence of *Fibonacci* numbers that are less than 1000. *Fibonacci* numbers are defined as follows:

F0 = 0
F1 = 1
Fn = Fn-1 + Fn-2

Its output is: *0, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987*

11.2. *for* Statement

The for statement offers another way to iterate a sequence of actions in a Java program. Its syntax is as follows:

```
for (expression1; expression2; expression3)
        {
        Embedded statement(s)
        }
```

The expression1 is evaluated only once, when the for statement is entered. Once expresion1 is evaluated, expression2 is evaluated, and if the result of expression2 is true, the embedded statements are executed. Afterwards expression3 is evaluated. The **evaluate-execute loop** continues until expression2 eventually evaluates to false.

For the same problem of generating Fibonacci numbers, we can use for loop to accomplish the same result as follows:

```
public class FibNumbers
        {
        public static void main(string[] args)
                {
                System.out.print(0);
                int fib0 = 0;
                int fib1 = 1;
                int fib2;
                for(fib2 = fib1 + fib0; fib2 < 1000; fib2 = fib1 + fib0)
                        {
                        fib0 = fib1;
                        fib1 = fib2;
                        System.out.print(", " + fib1);
                        }
                }
        }
```

The for loop can become a counting loop in the following manner:

```
for (   counter initialization expression;
        counter testing expression;
        counter updating expression)
            {
            ...
            Embedded statement(s)
            ...
            }
```

For example, variable x in the for loop below is such a counter:

```
for (int x = 0; x < 100; x++)
        {
        ... ...
        }
```

Assignments

1. Change your Exercise 4 program in unit 10 with the *while* statement so that the user may repeat the choosing process for additional purchases. The program shall display the purchased items and the total price after each input.

2. Draw a flow chart to show the *for* statement of the example in section 11.2.

UNIT 12. Methods

Objective: The student will learn the concept of subprograms.

Prerequisites: Knowledge of Java control statements

Materials:

Contents:

12.1. Method Definition

In Java, every executable statement must be within some method. Consequently, the *methods* are where the action is.

Example 12.1
The cube( ) Method: This program tests a method named cube ( ) that returns the cube of the integer passed to it.

```
public class TestCube
        {
        public static void main (String[ ] args)
                {
                for (int i = 0; i < 6; i++)      System.out.println (i + "\t" + cube (i));
                }

        static int cube (int n)
                {
                return n*n*n;
                }
        }
```

Here is its output:

```
0       0
1       1
2       8
3       27
4       64
5       125
```

The main( ) method contains a for loop which invokes the println( ) method 6 times. That method invokes the cube ( ) method, passing the value of its argument i to its parameter n. So, for example, on the third iteration, i = 2, the variable n is initialized to 2 inside the cube ( )

method. It then computes the value 8 from the expression n*n*n and returns it to the println ( ) method which prints it.

A ***local variable*** is a variable that is declared in a method. They can be used only within that method, and they cease to exist when the method finishes its execution. In the above example, the variable i is local to main ( ) and parameter n is local to cube ( ).

A **void method** is a method whose return type is void. That means that the method does not return a value. Note that the main ( ) method itself is a void method.

12.2. Methods Invocation

We have already seen an example of method that invokes other methods. In Example 12.1, the main( ) method invokes the println( ), which then invokes the cube( ) method.

A method that invokes itself is called **recursive** and the resulting process is called **recursion**.

Example 12.2.
The factorial function n! can be defined recursively as n! = 1 for n = 0, and n! = n*(n-1)! for n > 0. The function can be implemented as the following method.

```
static long f (int n)
        {
        if (0 <= n && n < 2) return 1;
        return n*f(n-1);
        }
```

12.3. Overloading

You can use the same name for different methods as long as they have different parameter type lists. This practice is called **overloading**.

Example 12.3
The following two methods, both named max( ). They have the distinct parameter type lists (int, int) and (int, int, int):

```
static int max(int m, int n)
        {
        if (m > n) return m;
        return n;
        }

static int max (int n1, int n2, int n3)
        {
        return max( max(n1, n2), n3);
```

```
}
```

A method's name and parameter type list is called its **signature**. For example, the signatures of the two methods in Example 12.3 are max(int, int) and max(int, int, int). It is the method's signature that the compiler uses to locate its definition when it encounters its invocation. That is why overloaded methods must have different signatures.

Assignments

1. Write a method that expects an integer parameter. The method shall return the string "*even*" or "*odd*" according to the integer input.

2. Write a program with the *main* method and three others so that the user may repeatedly make selections and have the purchase information updated. The three user-defined methods are listed as the followings:
   A. Method *itemChoice* returns the user's choice as
      a. A chair ($75.00)
      b. A table ($140.00)
      c. A book shelf ($99.00)
      d. Any other input shall end the program
   B. Method *totalPrice* returns to total purchasing price.
   C. Method *totalDisplay* shall display the number of each item being purchased and the total price.

3. Use the recursive methods to calculate the probability function: ${}_nC_m$
   A. m and n are integers
   B. $m < n$
   C. ${}_nC_m = (m!)\times((n-m)!)/(n!)$
   D. where $(n!) = n\times((n-1)!)$

UNIT 13. Classes

Objective: The student will learn the concept of Object-Oriented Programming (OOP).

Prerequisites: Knowledge of Java methods.

Materials:

Contents:

13.1. Basic principles of OOP

OOP is based on four key principles: **abstraction**, **encapsulation**, **inheritance**, and **polymorphism** – A PIE.

- ***Abstraction*** is the process of refining away the unimportant details of an object, so that only the appropriate characteristics that describe the object remain.

- ***Encapsulation*** is a way to associate the data and its operations closely together and treat them as a single unit of organization -- *class*.

- ***Inheritance*** means being able to declare a type which builds on the fields (data and methods) of a previously declared type. As well as inheriting all the operations and data, you get the chance to declare your own versions and new versions of the methods, to refine, specialize, replace or extend the ones in the parent class.

- ***Polymorphism*** means name sharing. There are two types of polymorphism in Java: **overloading** and **overriding**. Overloading allows several methods to have the same name. Overriding occurs when one class extends another, and the *subclass* has a method with same signature as a method in the *superclass*.

13.2. Classes and Objects

Java treats variables in two different ways, depending on their type:

1. Variables of built-in types: boolean, char, int, etc (primitive types)
2. Objects of user-defined types

The difference is this: what you get depends on the type. If it is a primitive type, you actually get the variable and you can read, write and process it immediately. If it is a class type, you can't get all the above immediately. What you get is a reference variable -- a location that can hold a pointer to the desired object when you fill it in. One way to visualize classes and objects is to think of a class as like a rubber stamp. The rubber stamp (class) can create many imprints (objects).

Classes can have three kinds of members: **fields**, **methods**, and **constructors**. A field is a variable that is declared as a member of o class. Its type may be any of the eight primitive types or a reference to an object. A method is a function that is used to perform some action for instances of the class. A Constructor is a special kind of function whose only purpose is to create the class's objects. This is called *instantiating* the class, and the objects are called *instance* of the class.

Class constructors are different from class methods in three distinct ways:

- Constructors have the same name as the class itself.
- Constructors have no return type.
- The ***new*** operator invokes constructors.
- If your class has no explicitly declared constructors, then objects can be created only by means of the class's (implicit) default constructor.

The syntax for a simple class declaration is

modifiers class class-name
{
body
}

where the optional ***modifiers*** may be one or more of the three keywords {**public**, **abstract**, **final**}, class-name is any valid identifier and body is a sequence of declarations of variables, constructors, and methods.

Example 13.1
A Class to Represent Coin Purses

```
public class Purse
    { // An object represents a coin purse
    private int pennies;
    private int nickels;
    private int dimes;
    private int quarters;

    public float dollars ( )
        {
        int p = pennies + 5*nickels + 10*dimes + 25*quarters;
        return (float)p/100;
        }

    public void insert (int p, int n, int d, int q)
        {
        pennies += p;
        nickels += n;
        dimes += d;
```

```
my_array[2] = 42;
my_array[3] = 99;
```

Or alternatively, you can use the following to combine declaration, creation and initialization:

```
int[ ] my_array = {55, 36, 42, 99};
```

Every array has an instance variable called length, which indicates the number of elements in the array. For example the length variable for my_array can be referenced in the following manner:

```
my_array.length
```

It will return the value of four.

14.2. Array Processing

Arrays are almost always processed using the for loops. The index of the for loop matches the array index (from 0 to length - 1). So you can use the following during array processing:

```
for (int i=0; i<my_array.length; i++)
//...
```

Example

Compute the average: The following program computes the sum of the integers in my_array and displays the average.

```
public class DisplayAverage
        {
        public static void main (String argv[ ])
                {
                int counter, sum = 0;
                int[ ] my_array = {55, 36, 42, 99};
                for (counter = 0; counter < my_array.length; counter++)
                        sum = sum + my_array[counter];
                System.out.println ("The average of the " + my_array.length);
                System .out.println ("integers in my_array is " + sum/ my_array.length);
                }
        }
```

Here is its output:

```
The average of the 4 integers in my_array is 58
```

In Java, you can create an array whose element type is any of the eight primitive types or any reference type.

```
        quarters += q;
        }

    public void remove (int p, int n, int d, int q)
        {
        pennies -= p;
        nickels -= n;
        dimes -= d;
        quarters -= q;
        }

    public String toString ( )
        {
        return new String (quarters + "quarters +" + dimes + "dimes + "
        + nickels + " nickels +" + pennies + "pennies = $"
        + dollars ( ) );
        }

    public static void main (String[ ] args)
        {
        Purse purse = new Purse ( ); // invokes the default constructor
        System.out.println (purse);
        purse.insert (3, 0, 2, 1);
        System.out.println (purse);
        purse.insert (3, 1, 1, 3);
        System.out.println (purse);
        Purse.remove (3, 1, 0, 2);
        System.out.println (purse);
        }
}
```

The output is

```
0 quarters + 0 dimes + 0 nickels + 0 pennies = $0.0
1 quarters + 2 dimes + 0 nickels + 3 pennies = $0.48
4 quarters + 3 dimes + 1 nickels + 6 pennies = $1.41
2 quarters + 3 dimes + 0 nickels + 3 pennies = $0.83
```

The first line declares the reference purse and then calls the default constructor to create an empty Purse object to which it refers. The output from the second line confirms that the object has been initialized to the zero state: all fields are zero. The third line invokes the insert ( ) method to insert 48 cents into the purse as confirmed by the next println ( ) statement. Another test is made to the insert ( ) method, and the remove ( ) method is tested once.

Assignments

1. Create a class to record your inventories as a computer hardware company.
   A. Variables:
      a. harddisks (Hard Disk)
      b. cdrom (CD-ROMs)
      c. zipdrive (Zip Drives)
      d. ram (RAM)
      e. cpu (CPU)
   B. Methods
      a. A constructor for initializing all variables to be zero
      b. A method for inputs
      c. A method for displaying the records

Assignments

1. Modify the variables of the exercise in unit 13 to be arrays so that:
   A. Each array represents the category of those items.
   B. Elements of each array represent different models under that category.
   C. Correspondingly, the methods of the class shall also need to be changed.
   D. Documentation is an important component in this program to show the models and their corresponding array elements.
   E. *Your instructor may provide the device specifications for each model.*
   F. ***Remember***: Copy your original file before making any change.

UNIT 14. Arrays

Objective: The student will learn the concept of arrays.

Prerequisites: Knowledge of Java basic data types and iteration statements.

Materials:

Contents:

14.1. Definition, Declaration, Creation and Initialization

In Java, an array is an object that has a sequence of numbered elements of the same type. Elements in an array are stored and retrieved in terms of an integer index (used with a pair of square brackets [ ]), with the first element being indexed by 0, the second by 1, and so forth.

Declaration

To declare an array, simply follow the type name with a pair of empty square brackets:

```
int[ ] my_array;
```

You can also put the square brackets after the array identifier to produce exactly the same meaning:

```
int my_array[ ];
```

Creation

To create an array instance, you need to deploy the new operator.

```
int[ ] my_array;
my_array = new int[4];
```

The second line above indicates that an array of four integers (my_array) is created. Alternatively, you can combine the declaration and creation in the following manner:

```
int[ ] my_array = new int[4];
```

Initialization

When an array of integers is created, all the elements in the array are initialized automatically to 0. If you want to initialize a created array with specific values, for example, to initialize four integers in my_array to 55, 36, 42, and 99, respectively, you can do the following:

```
my_array[0] = 55;
my_array[1] = 36;
```

UNIT 15. Strings

Objective: The student will learn the way Java handles strings of characters.

Prerequisites: Knowledge of basic concepts regarding strings.

Materials:

Contents:

15.1. The String Class

A string is a sequence of characters. Words, sentences, and names are strings. Therefore, strings play a very important role in application software development.

Example 15.1
A simple string object.

```
public class Alphabet
        {
        public static void main (String[ ] args)
                {
                String abc = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
                System.out.println (abc);

System.out.println("The string has " + abc.length( ) + " characters.");
System.out.println("The character at index 5 is " + abc.charAt(5));
System.out.println("The index of character X is " + abc.indexOf('X');
System.out.println("The substring from index 2 to index 6 is " + abc.substring(2, 6));
System.out.println("The substring from index 6 to index 7 is " + abc.substring(6, 7));
System.out.println("The substring from index 10 to the end is " + abc.substring(10));
System.out.println("The substring from index 9 to index 9 is " + abc.substring(9, 9));

                }
        }
```

The output is as follows:

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
The string has 26 characters.
The character at index 5 is F
The index of character X is 23
The substring from index 2 to index 6 is CDEF
The substring from index 6 to index 7 is G
The substring from index 10 to the end is KLMNOPQRSTUVWXYZ
```

The substring from index 9 to index 9 is

15.2. String Operations

Change Case

Java uses toLowerCase() and toUpperCase() methods to convert a string to its lowercase and uppercase, respectively.

Example 15.2
Case conversion

```
public class CaseChange
        {
        public static void main (String[ ] args)
                {
                String s = "ThisIsAStringHavingBothCases";
                System.out.println (s);
                String slc = s.toLowerCase( );
                System.out.println(slc);
                String suc = s.toUpperCase( );
                System.out.println(suc);
                }
        }
```

The output is

```
ThisIsAStringHavingBothCases
thisisastringhavingbothcases
THISISASTRINGHAVINGBOTHCASES
```

Concatenation

The concatenation operator "+" can be used to construct larger strings from smaller ones.

Example 15.3
String concatenation

```
public class Concatenating
        {
        public static void main (String[ ] args)
                {
                String first = "Java";
                String second = "language";
                System.out.println(first + second);
                System.out.println(first + " " + second);
                System.out.println(second + ", " + first);
                String s2 = first + " " + second;
```

```
        System.out.println(s2);
        }
    }
```

The output is

Javalanguage
Java language
Language, Java
Java language

Replace Characters

The method replace() can be used to replace every occurrence of a character with another.

Example 15.4

Character replacement

```
public class ReplaceChar
    {
    public static void main (String[ ] args)
        {
        String s3 = "Java Language";
        System.out.println(s3);
        System.out.println(s3.replace('a', 'o'));
        System.out.println(s3.replace('L', 'C'));
        System.out.println(s3);
        }
    }
```

The output is

Java Language
Jovo Longuoge
Java Canguage
Java Language

Represent A Primitive Value in A String

The valueOf() method can be used to convert primitive values to strings.

Example 15.5

Primitive values converting

```
public class PrimitiveValConverting
    {
    public static void main (String[ ] args)
        {
```

```
        boolean b = true;
        char c = '$';
        int n = 55;
        double x = 3.1415926;
        String strb = String.valueOf(b);
        String strc = String.valueOf(c);
        String strn = String.valueOf(n);
        String strx = String.valueOf(x);
        System.out.println("b = " + strb);
        System.out.println("c = " + strc);
        System.out.println("n = " + strn);
        System.out.println("x = " + strx);
        }
}
```

The output is

```
b = true
c = $
n = 44
x = 3.1415926
```

Assignments

1. Continuing the exercises from unit 13 and 14, add arrays of strings to store the names of the models in each device category.
   A. String Arrays:
      a. modelharddisks (Hard Disk)
      b. modelcdrom (CD-ROMs)
      c. modelzipdrive (Zip Drives)
      d. modelram (RAM)
      e. modelcpu (CPU)
   B. Hint:
      - Strings are actually a form of single-dimensional character arrays.
      - Strings may also be collected and declared as an array of strings.
      - You may initialize an array of strings.
      - For example: a 5-string array

        String[ ] itemname = {"CD_ROM", "2x", "4x", "8x", "12x"};
   C. ***Remember:*** Copy your original file before making any change.

UNIT 16. File Streams for Input and Output

Objective: The student will learn how to read input from and write output to files.

Prerequisites: Knowledge of basic input and output, and files.

Materials:

Contents:
A stream is a sequence of data. Your Java program deals with two types of streams: input and output. The standard input stream to your program is from the keyboard and the standard output stream is from your program to your display. However, you can also have your program read from or write to files, thus the file input and output streams.

16.1. File Input Stream

To create an input stream connecting a specific file to your program, you need to create an instance of the FileInputStream class (in java.io package):

```
FileInputStream inFile = new FileInputStream("input.data");
```

where "input.data" specifies the file from which your program reads input data.

To be able to read complete numbers and strings from an input file, you must then convert a FileInputStream instance into a StreamTokenizer instance:

```
StreamTokenizer token = new StreamTokenizer(inFile);
```

Tokenizers treat whitespace characters as delimiters that divide character sequences into tokens. Therefore, a file containing the following data is viewed as a stream of 6 tokens separated by spaces and line-terminating characters:

```
5 8 12 7
3 2
```

The value of the current token is stored in the nval instance variable of the tokenizer, you can obtain the number using the following expression:

```
token.nval
```

Because the number stored in the nval instance variable is always a floating-point double value, you can use the following casting to convert it to integer:

```
(int) token.nval
```

When reaching the end of the token stream, the nextToken method returns a special value that is the same as the value of the **TT_EOF** instance variable of the tokenizer. Thus, you can use the following program segment in handling the input data:

```
while (token.nextToken( ) != token.TT_EOF)
        {
        (int) token.nval ...
        ...
        }
```

When you are done with an input file stream, you should close it using

inFile.close( )

16.2. File Output Stream

File output stream is very similar to that of file input stream. First, you need to create an instance of the FileOutputStream class (in the package java.io) to connect your program to a specific file for output.

FileOutputStream outFile = new FileOutputStream ("output.data");

After that you need to declare a PrintStream variable and create a PrintStream instance from a FileOutputStream instance in order to write String instances to the output file.

PrintStream output = new PrintStream(outFile);

After you have a PrintStream instance, you can write strings to the output file using print and println methods.

Once you are done with the output file stream, you need to close it using the following.

outFile.close( );

Example
A program that reads integers from the "input.data" file, sums them up and writes the result to the "output.data" file.

```
import java.io.*;
public class FileIO
        {
        public static void main (String argv[]) throws IOException
                {
                int sum = 0;
                FileInputStream inFile = new FileInputStream("input.data");
                StreamTokenizer token = new StreamTokenizer(inFile);
```

```
        FileOutputStream outFile = new FileOutputStream ("output.data");
        PrintStream output = new PrintStream(outFile);
        while (token.nextToken( ) != token.TT_EOF)
                {
                ...
                sum = sum + (int) token.nval;
                ...
                }
        output.println(sum);
        inFile.close( );
        outFile.close( );
        }
}
```

If the "input.data" contains the data as indicated in 16.1, then after the program finishes, the "output.data" contains the result of 37.

Assignments

1. Modify the exercises from unit 15 to keep all numerical records in a data file.
   A. First, create a data file called "myrecord.data". Output all numerical records to the file when the user completes the input.
   B. Second, add the file input process so that the program will read the information from the data file when it starts.
   C. The final program shall be able to perform the following three abstract processes.
      a. File input process: the program reads the existing record of each inventory item.
      b. User input process: the user inputs/changes the records
      c. File output process: the program puts all numerical records back to the data file.
   D. ***Remember:*** Copy your original file before making any change.

```
        FileOutputStream outFile = new FileOutputStream ("output.data");
        PrintStream output = new PrintStream(outFile);
        while (token.nextToken( ) != token.TT_EOF)
                {
                ...
                sum = sum + (int) token.nval;
                ...
                }
        output.println(sum);
        inFile.close( );
        outFile.close( );
        }
}
```

If the "input.data" contains the data as indicated in 16.1, then after the program finishes, the "output.data" contains the result of 37.

Assignments

1. Modify the exercises from unit 15 to keep all numerical records in a data file.
   A. First, create a data file called "myrecord.data". Output all numerical records to the file when the user completes the input.
   B. Second, add the file input process so that the program will read the information from the data file when it starts.
   C. The final program shall be able to perform the following three abstract processes.
      a. File input process: the program reads the existing record of each inventory item.
      b. User input process: the user inputs/changes the records
      c. File output process: the program puts all numerical records back to the data file.
   D. ***Remember:*** Copy your original file before making any change.