

Terminology, Ethics, and Introduction to a Simple Program

I. Terminology

Electronic Computers:

First generation computers (1946 - 1958) - used vacuum tubes

ENIAC - a computer without mechanical parts (developed by Eckert and Mauchly in 1946). Instead of mechanical switches it used vacuum tubes. It was developed for the U.S. Army to calculate ballistic tables. It occupied 15,000 square feet of space to hold the thousands of vacuum tubes in it. It could perform 5000 additions per second. Limitations of vacuum tubes are that they generated high heat (120 degrees if not cooled), the tubes burned out frequently, and to change the program could take up to two days. In 1952 these two men sold the first commercial computer, *UNIVAC* (to the Census Bureau). In 1953 IBM sold its first computer and within two years dominated the industry.

Second generation (1959 - 1964) - used *transistors* instead of vacuum tubes.

Third generation (1965 - 1970) - integrated circuits (electronic circuit etched on a silicon **chip**).

Fourth generation (1971 - now) - microprocessors (multiple functions on a single chip).

Fifth generation (under development) - atomic level research, etc.

Types of Digital Computers

- 1) Hand held - small programmable calculators
- 2) Microcomputer - "personal" computer or PC with limited memory used by individuals
- PC - personal computer
- 3) Minicomputer - usually used by one person possibly in a workstation environment
for
a small business
- 4) Mainframe - large, fast computers used by medium and large sized companies
usually with multiple terminals attached (frequently at removed locations)
- 5) Supercomputer - most powerful, fastest computer. The newest ones can perform 3 trillion calculations per second (just 10 - 15 years ago the speed was in the hundreds of millions of operations per second).

Terms

Computer - a machine that can

- 1) accept data,
- 2) store & retrieve data and instructions,
- 3) process that data, and
- 4) give or show the results

(Trivia note: The word computer comes from the Latin word *computare* which means "to compute".)

Input - feeding or putting data or instructions into the computer

(ex. via: keyboard, disk, cards, modem)

Output - obtain or see the processed data from the computer

(ex. via: printer, monitor, modem)

I/O (Input/Output)

CPU (Central Processing Unit) - controls the operation of the computer (i.e. the brains of the computer). It consists of two sections:

- 1) Control Unit - the component that coordinates and supervises all operations of the stored program (i.e. a traffic cop)
- 2) Arithmetic/logic Unit - the component that performs arithmetic and logic operations
(i.e. a calculator and decision maker)

Note: The speed of a microprocessor (really the speed of its internal clock) is measured in terms of megahertz (MHz) which means millions per second.

Types of Microprocessors:

- 1) CISC (Complex Instruction Set Computer) Note: used in all computers until 1994
- 2) RISC (Reduced Instruction Set Computer) Note: newer and faster processor

Errors:

- 1) Syntax or Compile-time Error - violation of the rules of the programming language
- 2) Logic or Run-time Error - Program executes but with the wrong result due to an error by the programmer.

Bug - an error in a program

Debug - fix or remove errors in a program

Types of Computer Systems

- A) Single-user system: ex.) a home computer
- B) Networked computers: ex.1) our lab ex.2) the Internet - a world-wide network

Storage and Memory

- a) RAM (Random Access Memory) - *primary* or main storage used for *temporary* storage of data or instructions.
Note: This memory is part of the main circuit board or motherboard
Note: This information is lost when the computer is shut off and is therefore sometimes called volatile storage. (RAM ran away)
- b) Secondary or auxiliary storage (*removable*) - used for permanent storage of data outside or away from the motherboard Note: *backup* copies are important so that work is not lost.
(ex. tape drive, floppy disk i.e.. magnetic record, CD, or hard disk which can be internal)
- c) ROM (Read Only Memory) - permanent storage of data on memory chips on the main circuit board. Note: This information is not lost when the computer is shut off.
(Usually it is internal instructions to the computer so that it will operate correctly, put there by the manufacturer.)

Bus - the system of connections that links the microprocessor to the RAM, ROM, and input & output ports

Interface - a connecting link that allows two different components or system to work together

IDE (Integrated Debugging Environment) - We will be using the *Ready to Program with Java Technology IDE*

Data Storage and Representation:

When a key is depressed on a computer keyboard electrical impulses are sent down the pathways of the microchip opening or closing tiny circuits . Each opening or closing represents a single unit of information. This opening or closing (on or off) - similar to a light switch turned on or off - can be represented by the digits of 0 or 1 (note: open circuit or off = 0 and closed circuit or on = 1). There are only two digits (0 and 1) possible since there are only two states possible (open/on or closed/off). The number system of base 2 which is called the binary number system can therefore be used to represent what has so far been discussed.

Groups of these circuits next to each other are needed to represent numbers or letters.

It turns out that computers use groups of 8 circuits represented by binary digits (bits) to represent a single number, letter, or character.

BITS (binary digits) - the smallest unit of information that can be recognized by a computer represented by the digits 0 or 1.

BYTE - storage location for a single number, letter, or character containing 8 bits.

Note: Nibble - half of a byte i.e. 4 bits, kilobyte - 2^{10} (1024 bytes), megabyte - 2^{20} (1,048,576 bytes), gigabyte - 2^{30} (1,073,741,824 bytes), terabyte - 2^{40} (1,099,511,627,776 bytes)

Miscellaneous Terms

Hardware - the physical equipment of a computer system

Peripherals - devices attached to the computer

Software - any material that allows one to operate the computer
(ex. programs, written instructions etc.)

OS (Operating System) - instructions that manage the hardware resources
(ex. DOS, MS-DOS, ProDOS)

GUI (Graphic User Interface) ex. Mac and Windows

Programming Language - a set of rules, symbols, and special reserved words used to construct a program

1) Low-Level

a) machine language - uses binary code thus directly communicating with the computer

b) assembler language - uses mnemonics to represent machine language code

2) High-Level - languages that are closer to ordinary English

ex. Pascal, FORTRAN, COBOL, BASIC, Modula-2, LISP, Ada, Java, C, and C++

Note: C and C++ (a better C) have features built into it that allows it to have the power and flexibility of the low-level languages.

Note:

For the computer to understand and use a high-level language it needs to be translated to machine language by either a program called an Interpreter or a Compiler.

A program written with a high-level language is called source program (or source code). After translation by the compiler we now have a machine language program called the object program (or object file). A special linker program then combines the object file with needed machine code to produce an executable file that, unless changes are made, it will not need to be compiled again.

Java Translation and Execution

A Java compiler translates Java source code into Java bytecode. A Java interpreter translates and executes the bytecode. This bytecode is not associated with any particular machine so is machine independent or also called architectural neutral. This bytecode is written for the Java Virtual Machine (JVM). A JVM has been written for every major operating system. JVM is like a simulated CPU that lives on top of the operating system. The Java compiler and interpreter are part of the Java Software Development Kit (SDK) or also called the Java Development Kit (JDK) and can be downloaded free from Sun Microsystem Web site (java.sun.com).

II. Ethics (an AP topic)

Topics to consider: pirating, copyright and patent violations, hacking, spam, privacy and confidentiality issues, giving credit for intellectual property (copying items or pictures from a Web page)

III. Introduction to a Simple Program

Note: Syntax - the formal rules (grammar & punctuation) for how instructions are to be written so that they will be understood by the computer

Semantics - the rules that determine the meaning of the instructions

A. Output Introduction - **System.out.println();**

```
ex.1) public class APch1ex1
      {
        public static void main(String[ ] args)
        {
            System.out.println("My name is Bob.");
            System.out.print(" Hello World!");
        }
      }                                     // end of the main method
                                           // end of the APch1ex1 class
```

print() and **println()** - are *methods* (like a function) that performs some action which is to show something on an output screen. It must be associated (connected) to what Java calls an *object* by a period actually called a *dot operator*. The object in this and all outputs is called *out* (the standard console window). An object can only be used only if we know what *Class* it is from. The class for *out* is the *System* class. (More about Classes and objects later.)

An output alternative:

```
ex.2) import hsa.Console;
      public class APch1ex2
      {
        static Console c;                       // The output console
        public static void main(String[ ] args)
        {
            c = new Console( );                 // declaring an object
            c.println("My name is Bob.");
            c.print(" Hello World!");
        }
      }
```

The object is now *c* . If using our Intro. Programming software to output you must have the import statement of **import hsa.Console;** (if using the object of *c*). The class for *c* is the *Console* class. The *Console* class is written by the author of our Intro. book and creates an output window screen designed by him.

AP Programming - Chapter 1 Lecture

Note: Normally all lines output (print) immediately after the preceding output if using a `print()` and will be on separate lines if using `println()` - see next overhead

OR

```
ex.3) print("The answer is " + 5 + ".");
```

OUTPUT: The answer is 5.

This last ex.) is an example called *concatenation* i.e. connecting strings together. The `+` makes everything into one long string that is then output. The `+` is a special method that is part of the String class which is found in the **java.lang** package which is a package that is automatically imported into all Java programs (so we never need to import it) Note: **java.lang** also contains some Math classes and methods as we will find out later and the also contains the System class.

B) Starting New Lines in Output (two methods)

- 1) To start a new output line, you can include `\n` (called an *escape character*) in a quoted literal string. The `\n` is typed as two symbols with no space between them. (Note: AP topic)

```
ex.1) print("Hi\n");  
      print("Bye\n");
```

OR

```
ex.2a) print("Hi\n" +  
            "Bye\n");
```

```
or ex.2b) print("Hi\n" + "Bye\n");
```

OUTPUT: Hi
Bye

- 2) You can also start a new line by using **println()**

```
ex.1) println("Hi");  
      println("Bye");
```

OUTPUT: Hi
Bye

AP Programming - Chapter 1 Lecture

3) Both in one example

```
ex.1a) print("The answer is\n") ;  
        println(5) ;  
        println("Bye") ; // or print("Bye") ;
```

OR

```
ex.1b) println("The answer is") ;  
        print(5 + '\n') ; // or println(5 + "\n") ;  
        println("Bye") ; // or print("Bye") ;
```

OR

```
ex.1c) println("The answer is") ;  
        println(5) ;  
        println("Bye") ; // or print("Bye") ;
```

OUTPUT: The answer is
5
Bye

Note: The last example is much better!

C) More Escape characters - \t, \", \', \\,

(Note: See Appendix A6 page 1150-1151 in the Big Java textbook for these and more that we will not study.)

1) \t - tabs over

2) \" - shows a double quote mark " (Note: AP topic)

3) \' - shows a single quote mark '

4) \\ - shows a single backslash mark \ (Note: AP topic)

D) Program Construction, Structure, and Layout

```
ex.1) // Area of Circle Program
      /* Lecture Example
        for AP Programming */
      public class SomeClassName
      {
          public static void main(String[ ] args)
          {
              final double PI = 3.14;
              double area;
              int radius = 2;
              area = PI * radius;
              System.out.print("area = " + area);
          }
      }
```

1) Comments (two types) - comments or remarks that are ignored by the computer.
(Note: These are optional but recommended.)

- 1) // single line comment
- 2) /* ... */ single or multiple line comment

2) **import** statement - To create an object, you must specify the class from which the object is to be created. The class can be located in one of two locations. It can be located in the same directory as the program, or it can be located in a group of classes called a package. If the class is located in the same directory as the program, the Java compiler finds it automatically. If the class is located in a package, the program must specify the name of the package using an import statement. The format of the import statement is:

import [package-name].[Class-name];

There are a number of packages in the Java class library and they all have names that start with java. For example, the classes used to read from and write to files are all located in the **java.io package**. Several of the classes used in the Intro. book are created specifically for use in a teaching environment. These classes are all located in the *hsa* package. For example, to use the *Console* class, you must include the statement **import hsa.Console**;

- 3) **public class** SomeClassName - all Java programs must be started by a heading which contains the name of the class. The modifier **public** is used so that other programs can access this one, if necessary.
- 4) Braces { and } - they mark the start and end of a block of code
Note: A block is a sequence of zero or more statements enclosed by a { } pair.
- 5) **public static void** main(String[] args) - a method
this is the heading of the *main* method that this class contains. In any Java program one of the classes must contain a *main* method and there can only be one of these! This is the point where the computer actually starts carrying out the program. The three modifiers **public static void** must be used and again is an advanced topic. the variable args (which could be any variable but by convention everyone uses args) is called a parameter and is of type String array which is another advanced topic.
Note: The class file that contains the *main* method is called the application program.
Note: A method heading does NOT end in a semicolon!
- 6) Statement(s) - special executable computer instructions ending with a semicolon
Note: The null statement is a line consisting of only a semicolon, which does nothing, but is allowed.
- 7) Output Shortcut from ex.2) page 6 - the two lines **static** Console c; and c = **new** Console(); are used to create a new object called c which is of Console class type. This new Console object is the output window on which the results of programs will be displayed. Yes this is a shortcut; we will see in a later chapter how to output without using the Console class supplied by our book's author but by using the System class (supplied in all Java programs found in java.lang)
Note: The reserved word **new** is always used when creating (instantiating) an object. More about creating objects later.

E) Introduction to Objects - Terminology

Recall: Data Types - **int, char, double, float, boolean** ex.) **int** number;

1) Object - a "special" variable created from a *class* not from a built-in data type

2) Class - a programmer-defined data type out of which objects and methods can be created

3) Member Methods - functions associated with a class that can only be accessed (called) by the objects also in that class

Dot Operator syntax- the period (dot) used in the call to a member method that connects the object to the member method. It starts with the class object (variable), then a dot, then the member method name, then the argument.

ex.) **c.print(x)**

c is an object of the Console class and `println()` is a member function of the Console class

//Example of a programmer defined class:

//Only a portion of the application and implementation files for the Example class

```
import hsa.Console; // or import hsa.*;
public class Example
{
    static Console c;
    public static void main(String[ ] args)
    {
        c = new Console( );
        Circle circleA, circleB;
        circleA = new Circle( );
        circleB = new Circle( );
        circleA.output( );
        circleB.output( );
        circleA.area( );
        circleB.area( );
    }
    public class Circle
    {
        public void output( )
        {
            c.println("Center is " + x + "," + y);
            c.println("Radius is " + r);
        }
        public void area( )
        {
            double pi = 3.14;
            c.println("The area is " + pi*r*r);
        }
    }
}
```

Name the following from the example above:

- 1) the three classes (i.e. the *type* like **int** or **double**) -
- 2) the three objects (i.e. the *variables* of the class) -
- 3) the two class member methods associated with the objects -