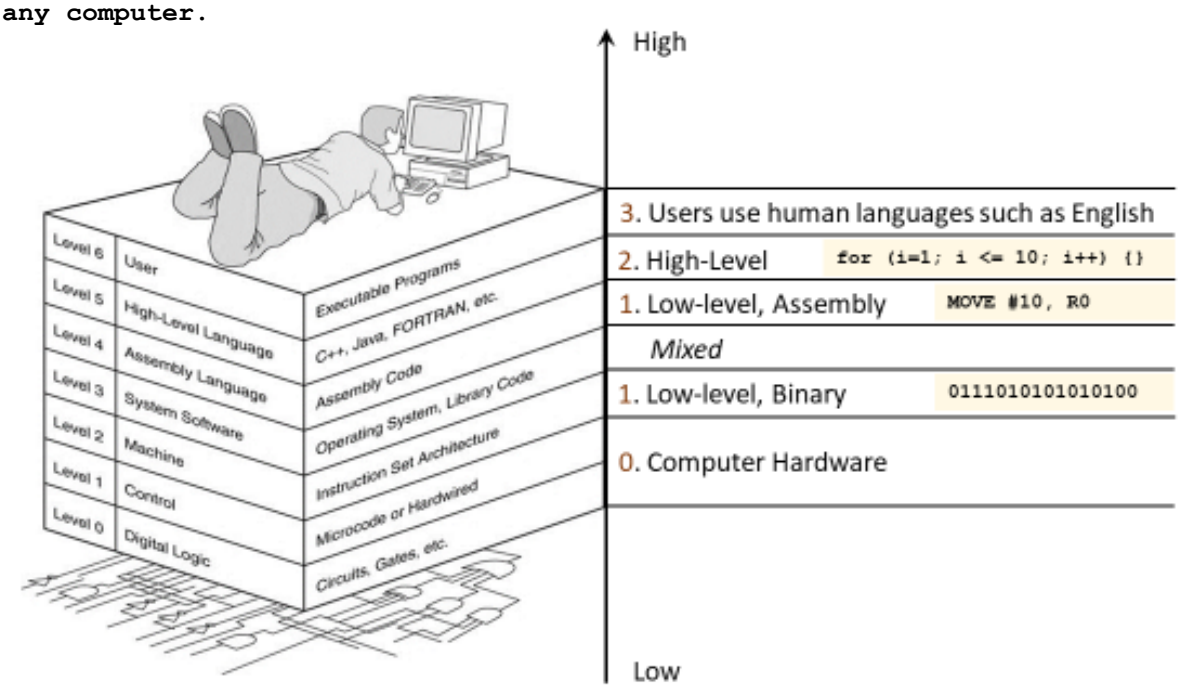
PKG 02: INTRODUCTION and ELEMENTARY PROGRAMMING

KEY TERMS ---Daniel Liang

- ***Computer programming*** **is the writing of instructions (i.e. code) for computers to perform.**
- ***Machine language*** **is a set of primitive instructions built into every computer.**
- ***Assembly language*** **is a low-level programming language in which a mnemonic is used to represent each machine-language instruction.**
- ***High-level languages*** **are English-like and easy to learn and program.**
- ***Source program*** **is a program written in high-level language.**
- ***Compiler*** **is a software program that translates the source program into a machine-language program.**
- ***Java*** **is platform independent, meaning that you can write a program once and run it on any computer.**



Computer Level Hierarchy Programming Language Hierarchy

- **Every Java program is a set of class definitions. The keyword *class* introduces a class definition. The contents of the class are included in a *block*. A block begins with an opening brace { and ends with a closing brace }.**
- ***Methods*** **are contained in a class. To run a Java program, the program must have a main method. The main method is the entry point where the program starts when it is executed.**
- **Every *statement* in Java ends with a semicolon ; or the statement terminator.**
- ***// line comment*** **Java comments are preceded with two slashes.**
- ***/* block comment or paragraph comment */*** **One or several lines.**

```
Latin:   Experientia Docet
English: Experience Teaches
```

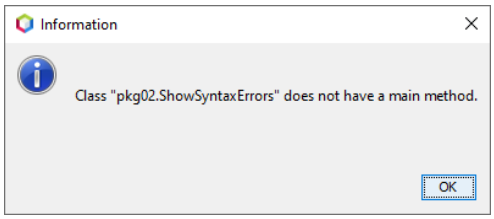
```
/**
 * @author CSC 210.03 Students
 */
public class IntroToProgramming {

    public static void main(String[] args) {
        // Display SFSU motto
        System.out.println("Latin:   Experientia Docet");
        System.out.println("English: Experience Teaches");
    }
}
```

INTRODUCTION and ELEMENTARY PROGRAMMING

- ***Syntax errors or compile errors*** **are errors reported by a compiler.**
- ***Runtime errors*** **are errors that cause a program to terminate abnormally.**
- ***Logic errors*** **occur when a program does not perform the way it was intended to.**

SYNTAX ERRORS




```java
package pkg02;

public class ShowSyntaxErrors {

    public static main(String[] args) {
        System.out.println("Welcome to Java); // Syntax Error
    }
}
```

RUNTIME ERRORS

```java
package pkg02;

public class ShowRuntimeErrors {

    public static void main(String[] args) {
        System.out.println(1 / 0);  // Divided by Zero
    }
}
```

```
Build (try) × 2019FA_CSC210_JAVA (run-single) ×
Updating property file: E:\Box Sync\IDEs\NetBeansProjects\2019FA_CSC210_JAVA\build\built-jar.properties
Compiling 1 source file to E:\Box Sync\IDEs\NetBeansProjects\2019FA_CSC210_JAVA\build\classes
compile-single:
run-single:
Exception in thread "main" java.lang.ArithmeticException: / by zero
        at pkg02.ShowRuntimeErrors.main(ShowRuntimeErrors.java:6)
E:\Box Sync\IDEs\NetBeansProjects\2019FA_CSC210_JAVA\nbproject\build-impl.xml:1339: The following error
E:\Box Sync\IDEs\NetBeansProjects\2019FA_CSC210_JAVA\nbproject\build-impl.xml:948: Java returned: 1
BUILD FAILED (total time: 1 second)
```

LOGIC ERRORS

```java
package pkg02;

public class ShowLogicErrors {

    public static void main(String[] args) {
        System.out.println("Celsius 35 is Fahrenheit degree ");
        System.out.println((9 / 5) * 35 + 32);  // 67, wrong, 9/5 = 1
        System.out.println((9.0 / 5) * 35 + 32);  // 67, wrong

    }
}
```

```
Celsius 35 is Fahrenheit degree
67
95.0
```

INTRODUCTION and ELEMENTARY PROGRAMMING

01. Which of the following statements is correct?
   a. Every line in a program must end with a semicolon.
   b. Every statement in a program must end with a semicolon.
   c. Every comment line must end with a semicolon.
   d. Every method must end with a semicolon.
   e. Every class must end with a semicolon.

02. Which of the following statements is correct to display Welcome to Java?
   a. System.out.println('Welcome to Java');
   b. System.out.println("Welcome to Java");
   c. System.println('Welcome to Java');
   d. System.out.println('Welcome to Java");
   e. System.out.println("Welcome to Java');

03. Java compiler translates Java source code into _____.
   a. Java bytecode
   b. machine code
   c. assembly code
   d. another high-level language code

04. _____ is a software that interprets Java bytecode.
   a. Java virtual machine
   b. Java compiler
   c. Java debugger
   d. Java API

05. The extension name of a Java bytecode file is
   a. .java
   b. .obj
   c. .class
   d. .exe

06. The extension name of a Java source code file is
   a. .java
   b. .obj
   c. .class
   d. .exe

07. Which of the following lines is not a Java comment?
   a. /** comments */
   b. // comments
   c. - comments
   d. /* comments */
   e. ** comments **

08. Which of the following are the reserved words?
   a. public
   b. static
   c. void
   d. class

09. Every statement in Java ends with _____.
   a. a semicolon (;)
   b. a comma (,)
   c. a period (.)
   d. an asterisk (*)

10. A block is enclosed inside _____.
   a. parentheses b. braces
   c. brackets d. quotes

SOME FUN

---

11. Programming style is important, because _____.

a. a program may not compile if it has a bad style
b. good programming style can make a program run faster
c. good programming style makes a program more readable
d. good programming style helps reduce programming errors

12. Which of the following code has the best style?

A:

```
public class Test {
public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

B:

```
public class Test {
    public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

C:

```
public class Test {
  public static void main(String[] args) {
  System.out.println("Welcome to Java!");
  }
}
```

D:

```
public class Test {
  public static void main(String[] args) {
    System.out.println("Welcome to Java!");
  }
}
```

13. If a program compiles fine, but it produces incorrect result, then the program suffers _____.

a. a compilation error
b. a runtime error
c. a logic error

14. If you forget to put a closing quotation mark on a string, what kind of error will be raised?

a. a compile error
b. a runtime error
c. a logic error

PROGRAMMING STYLE and DOCUMENTATION

- **Appropriate Comments and Comment Styles**
- **Proper Indentation and Spacing**
- **Block Styles**

ALT + SHIFT + F

1. Program Header

Each file must begin with a *program header*. This is a comment block that has the following format (fill in the parts in square brackets with the appropriate information):

```
/*************************************************************
 *
 *    File: [name of file]
 *    By: [name of author(s)]
 *    Date: [date last revised]
 *
 * Description: [short description on what program does, what user
 *               enters, what result(s) are displayed]
 *
 *************************************************************/
```

Note that this illustrates another way to indicate a comment block in Java. (Earlier, we saw that all text after // in the same line make up a comment.) Similarly, all lines in between /* and */ are comments. Hence:

```
/*
      a comment
      more comments
      even more comments
      etc etc
*/
```

2. Variable Names and Comments

Names of major variables should be *descriptive* of their role or usage in the program. For example, for a variable that contains the number of students in a class, numStudents is usually a better name than n.

Each major variable should have a comment describing its role in the program, when it is declared. For example:

```
      int numStudents; // number of students in class
```

3. Using Whitespaces and Indentation

Blank lines should be used to separate major blocks or sections of your program. This makes your program more readable.

Mark each level of nesting of statements with a separate level of indentation. You should indent with at least 2 spaces or a tab for each level; some prefer more spaces. Be consistent about how many spaces you use. Consider this code fragment:

```
      if (score >= 70) {   // process passing score
        System.out.println("Passed");
        pass = pass + 1;
      }
      else { // process failing score
        System.out.println("Failed");
        fail = fail + 1;
      } // end if (score >= 70)
```

Note that the statements for printing “Passed” and adding 1 to pass both belong in the block after if (score >= 70); hence, they are both indented a few spaces to the right of the if statement.

Similarly, the statements for printing “Failed” and adding 1 to fail both belong in the block after the else; they are both indented a few spaces to the right of the else.

A more complex fragment:

```
while (score >= 0) {
   if (score >= 70) {   // process passing score
      System.out.println("Passed");
      pass = pass + 1;
   }
   else { // process failing score
      System.out.println("Failed");
      fail = fail + 1;
   } // end if (score >= 70)

   // get next score
   System.out.println(“Enter next score: “);
      score = input.nextInt();
} // end while (score >= 0)
```

The leftmost (outermost) level is the while loop. The next level is the body of the while loop, which consists of the if-else block. The level after that are the printf and increment statements within the if-block and the else-block.

Note that the close curly brace of a block should be lined up with the statement that begins that block. Note from the same example:

```
while (score >= 0) {

      // some stuff omitted

} // end while (score >= 0)
```

There are two common options for placing the open curly brace. I usually put it on the same line as the statement it follows. It’s also possible to place it on the following line by itself, indented at the same level as the statement it follows:

```
while (score >= 0)
{
   if (score >= 70)     // process passing score
   {
      System.out.println("Passed");
      pass = pass + 1;
   }
   else                 // process failing score
   {
      System.out.println("Failed");
      fail = fail + 1;
   } // end if (score >= 90)

   // get next score
   System.out.println( “Enter next score: “;
   score = input.nextInt();

} // end while (score >= 0)
```

Either way is fine. Just be consistent.

Do not put more than one statement on the same line. Statements that are too long should be spread over multiple lines in a readable way. For example:

```
x = (- b + sqrt( pow(b, 2.0) – 4 * a * c) ) )
     / (2. * a);
```

4. INSERTING COMMENTS IN SOURCE CODE

Comments should be inserted to describe the tasks being undertaken. For example, in the if-else block:

```
if (score >= 70)  // process passing score
   {
      System.out.println("Passed");
      pass = pass + 1;
   }
else              // process failing score
   {
      System.out.println("Failed");
      fail = fail + 1;
   } // end if (score >= 90)
```

Or:

```
// get next score
System.out.println(“Enter next score: “);
score = input.nextInt();
```

Different programmers prefer different densities of comments. In general, there should be enough comments for a reader to follow the main actions in the program relatively easily.

Note also the use of comments to match close curly braces with open curly braces. This improves readability in large blocks. We see that the last closed curly brace is for the while loop, and the second to last close curly brace ends the if-else block:

```
while (score >= 0) {
   if (score >= 70) {   // process passing score


   }
   else { // process failing score


   } // end if (score >= 70)


} // end while (score >= 0)
```