**CSCI 1302**

**Final Exam Review**

Languages:

* Machine Level: Binary language
* Low Level: Assembly language, Byte Code
* High Level: Java language

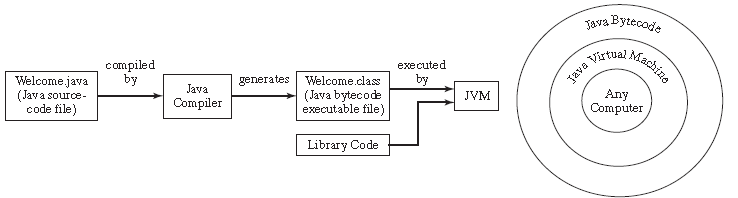
Know:

* API: Application Program Interface: the Java syntax is defined here and the Java Library is defined here
* JDK: Java Development Toolkit: the software for developing and running java programs
* JVM: Java Virtual Machine: interpreter that runs the Bytecode
* Compiler: transforms the entire source code (high-level language) as a single unit into another code (such as Bytecode) or machine level language
* Interpreter: executes programs line by line, like the JVM translates the Bytecode line by line into machine code (binary language)

Which does Java use – compiler or interpreter?

Both: Java source code --> compiles into Bytecode -->JVM (interpreter)

Know the diagram: page 17, Figure 1.8



Errors – 3 types

* syntax: compile errors: detected by the compiler
* runtime: causes a program to terminate abnormally
* logic: when a program does not perform the way it was intended

Algorithm: describes how a problem is solved by listing the actions that need to be taken and the order of their execution

Data Types:

* Primitive:
  + int, double, char, boolean
* Reference:
  + String, array

What are the problems with data types?

* Java does not warn of overflows
* floating-point numbers are not stored with complete accuracy

Valid identifier names & conventions

What is a variable? represents a value stored in the computers memory that may be changed in the program

How to declare variables

dataType identifier;

How to assign variables

dataType identifier = value;

Named constants: final int STAYSTHESAME = 5;

Prompt: directing the user to enter an input

Declare, assign & use Scanner

What has to be imported to be able to use the Scanner object?

page 38 import java.util.\*;

Math operations: + - \* / %

Casting/converions

int x = (int) 8.3;

double y = 9/(double) 8;

Comparison Operators: <, <=, ==, !=, >, >=

Evaluate: &&, ||, !, ^

if, if…else, if..else if..else statements

be able to generate a random number

Math functions: pow, sqrt, ceil, floor, rint, round, max, min, abs

char & its methods:

* isDigit(ch)
* isLetter(ch)
* isLowerCase(ch)
* isUpperCase(ch)
* toLowerCase(ch)
* toUpperCase(ch)

String & its methods:

* length()
* charAt(index)
* concat(st)
* toUpperCase()
* toLowerCase()
* equals(st)
* equalsIgnoreCase(st)
* compareTo(st)
* compareToIgnoreCase(st)
* startsWith(“zzz”)
* endsWith(“zzz”)
* contains(“zzz”)
* substring(index)

print vs println& printf

print: cursor stays - it does not advance to the next line

println: corsor advances to the next line

printf: formatting the output \*\*\*\* be able to use it!!!\*\*\*\*

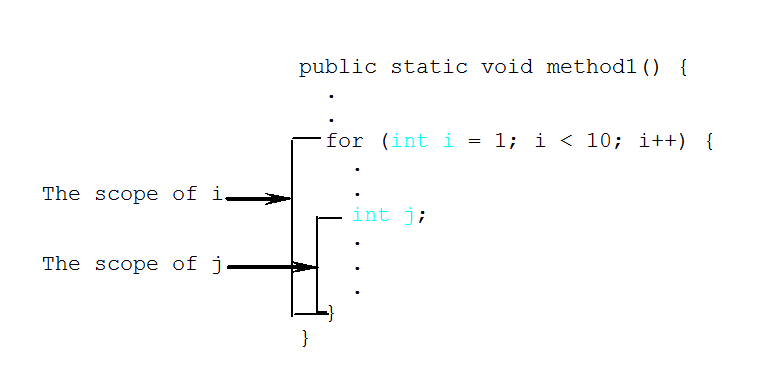
while loop:

* sentinel value
* count
* conditional
* infinite loop
* off-by-one error

for loop:

**Scope of local variables**

* A local variable: a variable defined inside a method
  + A parameter is a local variable
* Scope: the part of the program where the variable can be referenced.
* The scope of a local variable starts from its declaration and continues to the end of the block that contains the variable. A local variable must be declared before it can be used.
* You can declare a local variable with the same name multiple times in different non-nesting blocks in a method, but you cannot declare a local variable twice in nested blocks.



**Method** headers and the method call/invoke have to match-up on:

* The method name/identifier
* The return type
* The parameters:
  + Which have to match-up on the:
    - How many
    - The type
    - The order of these variables



\*\*\*\*\*\*\*\*\*\*\* In Java, methods are **pass-by-value** \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Methods modularize code making the code easy to maintain and debug and enables the code to be reused

**Overloading Methods:**

* Overloading Methods enables you to define the methods with the same name as long as their signatures are different.
* Thus two or more methods have the same name in one class but the parameter list is different.
* The Java compiler determines which method to use based on the method signature.
* Makes programs clearer and more readable. Methods that perform the same function with different type of parameters should be given the same name.
* Must have different parameters – not modifiers or return types
* Ambiguous invocation:
  + a compile error
  + when two or more possible matches for an invocation of a method and the compiler cannot determine the specific match

**File IO**

Declare and assign files

Declare and create PrintWriters, Scanners

**Arrays**

How to declare arrays:

int [] ary;

How to create:

ary = new int [size];

Array Initializers: Declare, create, and assign values

int [] ary = {1, 2, 3, 4};

How to assign values:

ary [0] = 1;

Process arrays:

for (int i = 0; i < myList.length; i++) {

System.out.print(myList[i] + " ");

}

What is passed to the method? The reference to the array

Understand why the binary search is faster than the linear search:

Because it only searches half of the array, then half of that, etc till it finds the key or will return where the key could be inserted or -1, meaning it was not found

Be able to trace through a segment of code

Be able to correct a segment of code

Be able to write a segment of code

**Be able to write a program!**

**Numberic Converions**

Be able to convert: Decimal to Binary, Binary to Decimal, Hex to Binary, Binary to Hex