تعريفات مهمة CS140

Java Virtual Machine (JVM) – a typical sequence of machine instructions

String: a sequence of characters enclosed in double quotation marks

compile-time error: A violation of the programming language rules that is detected by the compiler

Run-time error: Causes the program to take an action that the programmer did not intend

Algorithm: A sequence of steps that is:

- *unambiguous*
- *executable*
- *terminating*

Pseudocode: An informal description of an algorithm

Type defines a set of values and the operations that can be carried out on the values

The double type denotes floating-point numbers

Variable to store a value that you want to use at a later time

Identifier: name of a variable, method, or class

Assignment operator: = Used to change the value of a variable

Uninitialized Variables: It is an error to use a variable that has never had a value assigned to it

Object: entity that you can manipulate in your programs (by calling methods)

Method: sequence of instructions that accesses the data of an object

Class: declares the methods that you can apply to its objects

Public Interface: specifies what you can do with the objects of a class

Overloaded method: when a class declares two methods with the same name, but different parameters

length: counts the number of characters in a string

toUpperCase: creates another String object that contains the characters of the original string, with lowercase letters converted to uppercase

Parameter: an input to a method

Implicit parameter: the object on which a method is invoked "`greeting.length()`"

Explicit parameters: all parameters except the implicit parameter
`System.out.println (greeting).`

Return value: a result that the method has computed for use by the code that called it

Construction: the process of creating a new object

Accessor method: does not change the state of its implicit parameter

Mutator method: changes the state of its implicit parameter

API: Application Programming Interface

API documentation: lists classes and methods in the Java library

Package: a collection of classes with a related purpose

Object reference: describes the location of an object

Applet: program that runs inside a web browser

Instance variables store the data of an object

- You should declare all instance variables as private
- Private instance variables can only be accessed by methods of the same class

Instance of a class: an object of the class

Encapsulation is the process of hiding object data and providing methods for data access

- To encapsulate data, declare instance variables as private and declare public methods that access the variables

Constructors: contain instructions to initialize the instance variables of an object

***Unit test*:** Verifies that a class works correctly in isolation, outside a complete program

***Tester class*:** A class with a main method that contains statements to test another class

A **final** variable is a constant

•Once its value has been set, it cannot be changed

Math class: contains methods sqrt and pow to compute square roots and powers

- Math.pow(x, n)
- Math.sqrt(x)

Cast converts a value to a different type (int dollars = (int) balance;)

Math.round converts a floating-point number to nearest integer (Math.round(balance);

Remainder operator (%): returns the remainder of two numbers (`10 % 3` Is 1)

A **static** method does not operate on an object

The if statement lets a program carry out different actions depending on a condition

Nested if statement: if statement inside another if or else if statement

equals: To test whether two strings are equal to each other string1.equals(string2)

compareTo: compares two strings lexicographical order string1.compareTo(string2)

null reference refers to no object

Black-box testing: Test functionality without consideration of internal structure of implementation

White-box testing: Take internal structure into account when designing tests

Test coverage: Measure of how many parts of a program have been tested

Convert to number or to String:

- Integer.parseInt(str)
- Double.parseDouble(x)
- Integer.toString(n)

Substring: method has two variants and returns a new string that is a substring of this string

- Str.substring(x, y) x= begin Index, y= end Index

Scanner: class read keyboard input in a convenient manner

- nextIntreads a int
- nextDoublereads a double
- •extLinereads a line (until user hits Enter)
- next reads a word (until any white space)

Loop is a sequence of instruction s that is continually repeated until a certain condition is reached

Infinite loop is a sequence of instructions which loops endlessly (Loop runs for ever)

For Use a for loop when a variable runs from a starting value to an ending value with a constant increment or decrement

Nested Loop one loop inside the body of another loop

Array: Sequence of values of the same type int[] data = new int[10];

When array is created, all values are initialized depending on array type:

- Numbers : 0
- Boolean: false
- Object References : null

Arrays have fixed length

array.length Get array length, (Not a method!)

Bounds error: Accessing a nonexistent element results

ArrayList class manages a sequence of objects Can grow and shrink as needed

- ArrayList<Integer> data = new ArrayList<Integer>();
- get To Obtain the value an element at an index data.get(1)
- set To set an element to a new value data.set(1,"dd")
- remove To remove an element at an index data.remove(1)

Size ArrayList method yields number of elements

Wrapper classes to treat primitive type values as objects, you must use wrapper classes.

For each Loop used to traverse array or collection elements

Arrays.copyOf to make a true copy of an array

Two-Dimensional When constructing a two-dimensional array, specify how many rows and columns are needed

- String[][] board = new String[ROWS][COLUMNS];
- board[1][1] = "x";

Class represents a single concept from the problem domain

Name for a class should be a noun that describes concept
Actors objects do some kinds of work for you ex: Scanner

Utility classes –no objects, only static methods and constants ex Math

Cohesion A class should represent a single concept

Coupling A class depends on another if it uses objects of that class

- High coupling= Many class dependencies
- Minimize coupling to minimize the impact of interface changes

Accessor: Does not change the state of the implicit parameter
Mutator: Modifies the object on which it is invoke

Immutable class: Has no mutator methods (e.g. String)

Call by value: Method parameters are copied into the parameter variables when a method starts
Call by reference: Methods can modify parameters

Precondition: Requirement that the caller of a method must meet

Postcondition: requirement that is true after a method has completed

Static Methods A static method is not invoked on an object

Static variable belongs to the class, not to any object of the class
Static variables should always be declared as private

Scope of variable: Region of program in which the variable can be accessed
Scope of a local variable extends from its declaration to end of the block that encloses it