**CSCI 2302 Test 1 Concepts**

**Chapters: Multidimensional Arrays, Object & Classes,**

**and Object-Oriented Thinking**

**UML, UNIX**

UNIX commands: know the following commands and what the following do:

ls = list contents in directory

submit = submit the file(s) to the instructor account

cd = change directory – go to that directory

mv = move to specified directory

mkdir = make directory – makes the specified folder/directory

Create 2D arrays & know information about them, execute code

What does OOP stand for? What does it involve?

* Object-Oriented Programming
* Programming using objects

What are objects? How do we implement objects?

* Represents a real world entity in the real world that can be distinctly identified
* By creating/defining a class

What is a class?

* A contract that defines what an object’s data fields and methods will be

A class is also known as ADT (abstract data type)

What does each object have?

* Unique identity (name)
* State= data fields/variables
* Behavior = methods

Constructors: no-arg constructor, default constructor, explicitly defined

* A constructor is invoked to create an object using the new operator
* No arguments – normally provided
* If a constructor is not defined/implemented, then the no-arg, default is automatically provided

What is the difference between a primitive variable and a reference variable?

* Reference variable contains the reference to the object
* Primitive variable contains the value

What is an instance variable? Be able to identify one. What can one access & invoke?

* Instance variable: a variable that is dependent on the object – the specific instance of an object
* Can invoke an instance method & a static method
* Can access a instance data field & a static data field
* An instance variable is dependent on the object

What is a static variable? Be able to identify one. What can one access & invoke?

* static variable: A variable that is shared by all objects of the class
* cannot invoke an instance method nor access an instance data field
* can invoke a static method and a static data field
* non-dependent on an object, it is related to the class

Conversion

* boxing: a primitive type can be automatically converted to an object using a wrapper class
* unboxing: converting an object to a primitive type
* Java will automatically do this

What are modifiers? What are the access for each?

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Modifier | Symbol | Accessed from the same class | Accessed from the same package | Accessed from a subclass in a different package | Accessed from a different package |
| Public | + | 🗸 | 🗸 | 🗸 | 🗸 |
| Protected | # | 🗸 | 🗸 | 🗸 |  |
| Default |  | 🗸 | 🗸 |  |  |
| Private | - | 🗸 |  |  |  |

Data encapsulation – What is it? Why do we use it?

* To protect the data by using the private modifier
* To prevent data from being tampered with
  + To access or modify, have to use the setters & getters
* makes the class/object easy to maintain as each part is compartmentalized

What an accessor/getter? What is a setter/mutator?

* accessor/getter: to make the data field accessible
* setter/mutator: to enable a private data field to be updated

this keyword

* refers to the object itself
* can access hidden variables
* to invoke a constructor

What is passed when an object is passed to a method?

* The reference of the object

What is an Immutable object?

* All data fields must be private
* There can’t be any mutators for the data fields
* No accessor methods can return a reference to a data field that is mutable

What is the scope a local variable?

* Defined within the scope of what declared it – i.e. a method

What is the scope of a class variable? What variables are referred to as class variables?

* The entire class regardless of where it is declared
* Instance variables & static variables

Be able to create UML diagram

Be able to implement an UML diagram

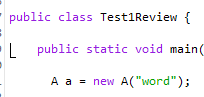
UML association

UML aggregation

UML composition

How are these relationships implemented?

Association:



Aggregation: has-a relationship



Composition: has-a relationship, where the child cannot exist independent of the parent



Wrapper classes

* A primitive type can be automatically converted to an object using a wrapper class, and vice versa, depending on the context, using the API (the library). The wrapper classes provide constructors, constants, and conversion methods for manipulating various data types.
* \*\*\* the wrapper classes do NOT have no-args constructors\*\*\*
* \*\*\* all instances of the wrapper classes are immutable\*\*\*

BigInteger & BigDecimal

* BigInteger allows very large integers
  + Has to be instantiated with a String value
* BigDecimal allows high-precision floating-point values – with no limit to the precision
  + Could have issues with the divide method if it cannot be terminated
  + has a better precision using the String argument than a number argument
  + Methods: add, subtract, multiply, divide, and remainder

Strings

* Manipulation
  + Immutable! Creates a new string
* Memory location
  + Pool: String s1 = "test"
  + Object: String s1 = new String("test")
* Methods:
  + indexOf: returns the index of the first occurrence of the specified character or string
  + startsWith: returns a Boolean value, true, if the string starts with the specified string
  + endsWith: returns a Boolean value, true, if the string ends with the specified staring
  + contains: returns a Boolean value, true, if the string contains the specified string
  + replace: returns a new String that replaces as set in the parameters
  + split: returns an array of Strings consisting of the substrings split by the delimiter
  + equals: returns a Boolean value; measures if two Strings have the same content with same capitalization
  + equalsIgnoreCase: returns a Boolean value; measures if two Strings have the same content ignoring the capitalization
  + compareTo: returns an int value; compares two Strings on the character values
  + compareToIgnoreCase: returns an int value; compares two Strings on the character values ignoring the capitalization
* Regex/matches
  + varString.matches("[A-Za-z0-9]{6}")

StringBuilders

* The difference between String & StringBuilder & which to use
  + StringBuilder – flexible – can append, insert, delete characters
* Methods
  + toString: StringBuilder to a String
  + capacity: how much space is allotted for the StringBuilder
  + insert: inserting character(s) at specified point
  + charAt: returns the character at specified index
  + setCharAt: sets a new character at specified index
  + append: adds to the end
  + delete: deletes based on the arguments
  + reverse: reverses the contents of the StringBuilder
  + replace: replaces character(s) based on the arguments