Students will be allowed 3 hours for the mid-term. 8:00am – 10:50am, Tuesday 5/19/15.

Students will be able to use their text book, handwritten notes, digital notes, and any other assignments or program created by the student. NO INTERNET

Students will not be allowed to use the internet for the final and will be asked to close all web browsers.

There will be 3 Sections for the Final

* SHORT ANSWER SECTION
* DIAGRAM THE BASIC STRUCTURES ASSOCIATED WITH PROGRAMMER CREATED CLASS
* SHORT PROGRAMMING TASKS

(Assignment #17)

FOR-EACH Style **for** Loop / enhanced **for** loop –

the enhanced for loop allows you to iterate through a collection without having to create an Iterator or without having to calculate beginning and end conditions for a counter variable. The enhanced for loop is the easiest of the new features to immediately incorporate in your code. In this tip you will see how the enhanced for loop replaces more traditional ways of sequentially accessing elements in a collection.

for (type *iteration-variable* : collection/array) *statement block*

Pg. 577**,** Java Programming *A comprehensive Introduction*

[**http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format**](http://www.oracle.com/technetwork/java/javase/documentation/index-137868.html#format)(Detailed explanation of Java documentation)

[**http://www.tutorialspoint.com/java/java\_documentation.htm**](http://www.tutorialspoint.com/java/java_documentation.htm)

[**http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments**](http://www.liferay.com/community/wiki/-/wiki/Main/Javadoc+Guidelines#section-Javadoc+Guidelines-Class+Comments)

Internal Documentation- if the notes on how and why various parts of code operate is included within the source code as comments. It is often combined with meaningful variable names with the intention of providing potential future programmers a means of understanding the workings of the code.

Internal documentation would be comments and remarks made by the programmer in the form of line comments and boiler plates.

External Documentation- External documentation would be things like flow charts, UML diagrams, requirements documents, design documents etc.

Java Doc Tags- **is a documentation generator from Oracle Corporation for generating API documentation in HTML format from Java source code. The HTML format is used to add the convenience of being able to hyperlink related documents together.**

(Assignment #18)

[**http://en.wikipedia.org/wiki/Linear\_search**](http://en.wikipedia.org/wiki/Linear_search)

**Linear Search (explain)-**

**In computer science, linear search or sequential search is a method for finding a particular value in a list that checks each element in sequence until the desired element is found or the list is exhausted.[1] The list need not be ordered. Linear search, also known as sequential search, is a process that checks every element in the list sequentially until the desired element is found. The computational complexity for linear search is O(n), making it generally much less efficient than binary search (O(log n)). But when list items can be arranged in order from greatest to least and the probabilities appear as geometric distribution (f (x)=(1-p) x-1p, x=1,2), then linear search can have the potential to be notably faster than binary search.**

**Traverse- like searching in data structure, move on nodes and check them**

**Iterate- Iteration is the repetition of a process in a computer program, usually done with the help of loops.**

(Assignment #19)

Procedural Programming vs Object Orientated Programming explain the difference.

Procedural programming is a programming paradigm, derived from structured programming, based upon the concept of the procedure call.

Object-oriented programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic.

Object- An object can be a variable, a data structure, or a function. In the class-based object-oriented programming paradigm, "object" refers to a particular instance of a class where the object can be a combination of variables, functions, and data structures.

Object orientated programming-

Object-oriented programming (OOP) is a programming language model organized around "objects" rather than "actions" and data rather than logic.

(Assignment #20)

**Encapsulation-**

**Encapsulation is the packing of data and functions into a single component. The features of encapsulation are supported using classes in most object-oriented programming languages, although other alternatives also exist.**

**this**.keyword- this is a reference to the current object — the object whose method or constructor is being called. You can refer to any member of the current object from within an instance method or a constructor by using this.

constructor chaining-

Calling another constructor in the same class from another constructor is called constructor chaining. By using this() we can call another constructor in the same class. Incase we want to call another constructor, this() should be the first line in the constructor. Below example shows code for

constructor chaining-

(Assignment #21)

**Polymorphism-**

**The word 'polymorphism' literally means 'a state of having many shapes' or 'the capacity to take on different forms'. When applied to object oriented programming languages like Java, it describes a language's ability to process objects of various types and classes through a single, uniform interface**

**Method Overloading-(Define and Give a short code example)**

**Method Overloading is a feature that allows a class to have two or more methods having same name, if their argument lists are different. In the last tutorial we discussed constructor overloading that allows a class to have more than one constructors having different argument lists.**

**Class something {**

**Public void some(int a) {**

**System.out.println(a);**

**}**

**Public void some(int a, double b) {**

**System.out.println(a);**

**}**

**Constructor Overloading-(Define and Give a short code example)**

**Constructor overloading is not complex you just need to create another constructor, obviously same name as of class but different signature but there are certain rules related to Constructor overloading which needs to be remembered while overloading constructor in Java. e.g. One Constructor can only be called from inside of another Constructor and if called it must be first statement of that Constructor.**

**Class something {**

**Int aa, bb;**

**Public void something(int a, int b) {**

**Aa = a;**

**Bb = b;**

**}**

**Public void something(int a) {**

**Aa = a;**

**Bb = 0;**

**}**

(Assignment #22)

Recursion

Recursion is a basic programming technique you can use in Java, in which a method calls itself to solve some problem. A method that uses this technique is recursive. Many programming problems can be solved only by recursion, and some problems that can be solved by other techniques are better solved by recursion.

Base Case:

The case in which we end our recursion is called a base case

(Assignment #23)

Static variables-

static variable is a variable that has been allocated statically—whose lifetime or "extent" extends across the entire run of the program.

Static methods-

Java static method program: static methods in Java can be called without creating an object of class.

(Assignment #24)

Describe the difference between an **Error** in Java and an **Exception.**

**Errors should not be caught or handled (except in the rarest of cases). Exceptions are the bread and butter of exception handling.**

Describe the difference between **Checked Exceptions** and **Unchecked Exceptions**

Checked: are the exceptions that are checked at compile time. If some code within a method throws a checked exception, then the method must either handle the exception or it must specify the exception using throws keyword.

Unchecked are the exceptions that are not checked at compiled time.

Give a few examples of **Checked Exceptions.**

|  |
| --- |
| class Main {      public static void main(String[] args) {          FileReader file = new FileReader("C:\\test\\a.txt");          BufferedReader fileInput = new BufferedReader(file);            // Print first 3 lines of file "C:\test\a.txt"          for (int counter = 0; counter < 3; counter++)              System.out.println(fileInput.readLine());            fileInput.close();      }  } |

Output:

Exception in thread "main" java.lang.RuntimeException: Uncompilable source code -

unreported exception java.io.FileNotFoundException; must be caught or declared to be

thrown

at Main.main(Main.java:5)

Give a few examples of **Unchecked Exceptions.**

|  |
| --- |
| class Main {     public static void main(String args[]) {        int x = 0;        int y = 10;        int z = y/x;    }  } |

Output:

Exception in thread "main" java.lang.ArithmeticException: / by zero

at Main.main(Main.java:5)

Java Result: 1

Describe basic structure of

**try**{

}

**Catch**

{

}

A try statement is used to catch exceptions that might be thrown as your program executes. You should use a try statement whenever you use a statement that might throw an exception That way, your program won’t crash if the exception occurs.