**Exam 1 Topic Review**

**Objects**

An object is an instance of a class and can a reference, variable etc.

Create an object by using its constructer and assigning it a name

**Constructor** is a similar to a method that's called when an instance of an object is created.

**A method**is a set of code which is referred to by name and can be called (invoked) at any point in a program simply by utilizing the method's name.

**State of and object** Example: A dog has states - color, name, breed as well as behaviors – wagging the tail, barking, eating

**Abstraction and Encapsulation**

Abstract away the implementation details

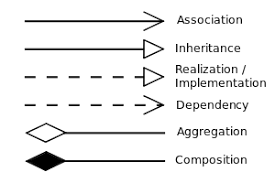
Users of your classes shouldn’t need to know the Implementation

**Encapsulation** in Java is a mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. ... Declare the variables of a class as private.

Aggregation – Has A relationship

Immutable classes have objects that cannot be changed and mutable have classes can be changed

**UML**



underline = static

*italic* = abstract

+ sign = public

- sign = private

# sign = protected

~ sign or nothing = package

ALL CAPS = constant

**Input/Output**

**File reading:**

BufferedReader br = new BufferReadered (new FileReader(“Filename.txt”));

String str = br.readLine();

String str = “Hello”;

Str. Substring(int start, int finish)

Scanner scan = new Scanner(System.in)

**String.format**

%(flags)(width)(.precision)(specifier)

flags – pads output with spaces or 0s

width – specifies the minimum number of characters to be printed

precision – number of characters after the decimal

%d = int, long, short

%f = float, double

%s = string

String.format(%04.2d, 146.78332) //output = 0146.78

**Inheritance**

Can only inherit one class

Apple extends fruit/Apple is a fruit

Can access all public and protected methods and variables

**Overloading** has same name but different parameters

**Overwriting** has same name and redefines a super class method

Super(); calls a super classes constructor

Super.methodName(): calls a super classes method

**Polymorphism** allows you to use super and

subtypes in reference declarations

Apple object = new Apple();

Fruit apple = new Apple();

**ArrayList**

Non fixed array

ArrayList <object> name = new ArrayList<object>();

Name.add(object);

Name.get(index);

Name.size();

Use for loops for iteration

**Exception and Errors**

Exceptions cause programs to halt

Shows line number where error occurred

Shows the stack trace

Syntax:

Try

{

//possibly problematic code  
}

catch (ExceptionName e)

{

//fix the problem

}

finally

{

//executes no matter what

}

We deal with exceptions to keep our programs from breaking

Errors are things we can’t do much about like storage

**Hierarchy**

Throwable

Exception

Runtime Exception

nullPointer

arithmetic exception

indexOutOfBounds

IOException

Error

Runtime exceptions are things that code is messed up on but checked exceptions or other exceptions are things that we can anticipate and catch

**Abstract Classes and Interfaces**

**Abstract classes**

cannot be instantiated.

When classes inherit they must define all the methods that are labeled abstract

Used to reduce redundant code

**Interfaces**

Apple inherits fruit

Form guidelines on classes and implement many interfaces

Cannot have implementations and cannot be instantiated

**Comparable and Comparator**

**Comparable**

compares two objects

Compares generics

If first is bigger than second = -1

If second is bigger = 1

If they are equal = 0

Object1.compareTo(object2);

**Comparator** compares two objects

Opposite of compareTo();

Compare(object1, object2);