COP 3330, Spring 2013

Final Exam Review

Instructor : Arup Ghosh
04-19-13

School of Electrical Engineering and Computer Science
University of Central Florida

Discrete Structures Review

- Set Theory
- Proof Techniques
- Equivalence Relations

Just

Kidding!

Final Exam

- Monday, April 29 at 1:00 PM in this room.
- 2 hr 50 minutes long.
- No aids of any kind.
- Things to bring
  - Yourself
  - PEN
  - ID

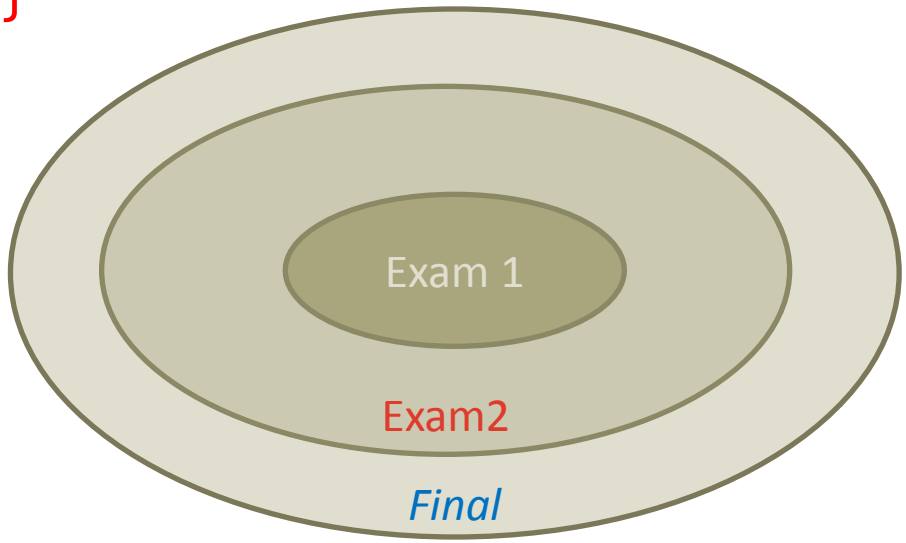
Final Exam Format

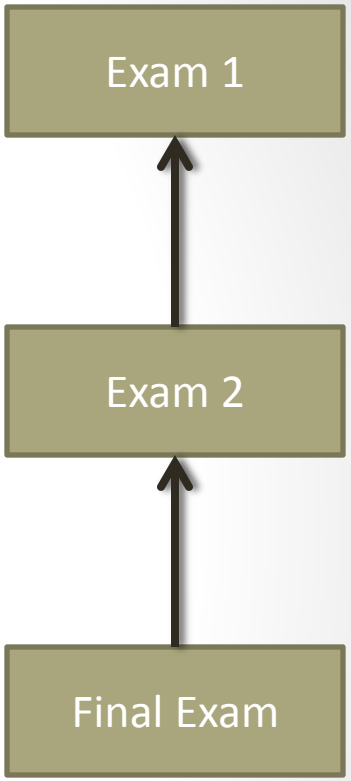
- Two sections – Total : 100 points
- First Section – Multiple Choice / TF (20%)
  - Quiz 1 to 8 – 10%
  - New – 10%

- Second Section – Free response (80%)

Read reviews for the previous exams

```
class FinalExamReview extends Exam2Review {
      public FinalExamReview () {
            super();
            stuffSinceThen();
      }
}
```





- So basically, go read the reviews for the previous exams again.

Essential topics

- Interfaces, inheritance, classes
- Using the Comparable interface to sort objects
- The modifiers final and protected
- Polymorphism (is-a)
- Collections
- Exceptions, errors
- Enums
- GUIs
- Threads
  - Runnable interface
  - Thread class

GUI

- JOptionPane is used to pop up canned dialogs:
  - JOptionPane.showMessageDialog() – Pop up a message and an OK button
  - JOptionPane.showConfirmDialog() – Pop up a message and Yes, No, and Cancel buttons
  - JOptionPane.showInputDialog() – Pop up a message and collect a string from the user

GUI

- Useful Swing Components
  - JFrame – A window
  - JTextArea – A multi-line text entry area
  - JTextField – A single line text entry area
  - JLabel – A text or image display area
  - JButton – A push button
  - Canvas – (From java.awt) A drawing area
    - To draw on a Canvas, you need to get access to its Graphics object

GUI

- Listeners
  - ActionListener
  - KeyListener
  - MouseListener
  - MouseMotionListener
- Implementation and use

JavaDoc

- JavaDoc comments are structured like so:

```
/**
 * [Description]
 * [Block Tags]
 */
```

JavaDoc

- Most important tags:
  - Parameters: @param
  - Return value: @return
  - Exceptions: @throws
- Other tags:
  - @author
  - @version
  - @see
  - @since
  - @deprecated
- Reference: http://en.wikipedia.org/wiki/Javadoc

Threads

- To create a Thread:
  - Pass a Runnable object to the constructor of Thread
  - Call the start() method on the Thread object to tell the JVM *it* can call the run() method of the Runnable object in a new thread when it has time

A Look Back

- And now to older material…

Language Basics

- Variables
  - Primitives
    - boolean, byte, char, short, int, long, float, double
  - Objects
- Control structures
  - if / else if / else, for, while, do-while, switch

Language Basics

- Literals
  - boolean literals (true, false)
  - byte, short, int, long literals
  - float, double literals
  - char literals
  - String literals

Language Basics

- Expressions
  - Arithmetic Expressions
    - +, -, *, /, %
  - Assignment
    - =, +=, -=, *=, /=, %=
  - Comparison
    - ==, !=, <, >, <=, >=
  - Logical Expressions
    - &&, ||, !

Language Basics

- Automatic widening
- Narrowing - Type casting
- Declaring constants

I/O

- Scanners
  - next(), nextInt(), nextLine(), etc.
- PrintStreams (i.e. `System.out`)
  - println(), print(), printf()

Strings

- String literals
- String concatenation (+)
- String comparison
  - compareTo()
  - equals()
  - equalsIgnoreCase()

Arrays

- One dimensional arrays
- Multidimensional arrays
- length field
- Declaring hard-coded arrays

Comments and Whitespace

- Line comment (//)
- Block comment (/* */)
- Indent properly! Code is unreadable otherwise!

Whitespace Example

```
import java.util.*;
public class x {
public static void main(String[] args) {
Scanner a=new Scanner(System.in);
System.out.print("input");
long c=1;
for(int b=a.nextInt(),d=1;d<=b;++d)c*=d;
System.out.println(c);
} }
```

Whitespace Example

```
public class Factorial {
  public static void main(String[] args) {

        Scanner stdin = new Scanner(System.in);

        System.out.print("Please input a number> ");
        int number = stdin.nextInt();

        long fact = 1;
        for (int i=2; i<=number; ++i) {
                fact*=i;
        }

        System.out.printf("%d! = %d%n",number,fact);

  }
}
```

Errors

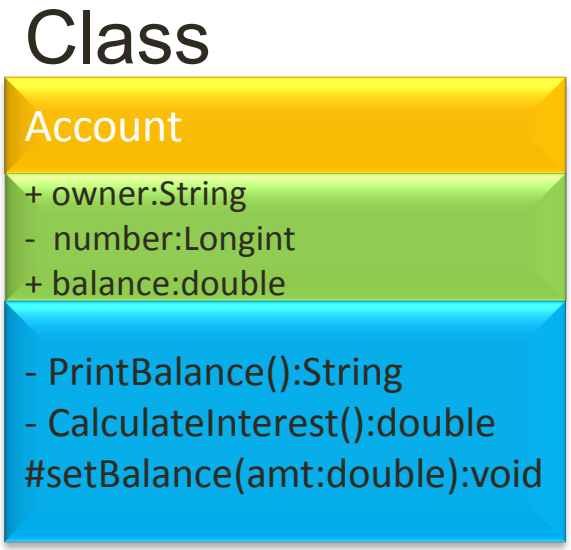
- Compilation Errors
- Runtime Errors
- Logic Errors

Class Basics

- Instance Variables
- Instance Methods
- Static Variables
- Static Methods
- Constructors

UML Basics

- Given a class description, be able to write code for this class with appropriate fields and methods.



Methods

- A method is laid out as follows:

```
modifiers return-type identifier(formal parameters)
{
    body
}
```

- Non-void methods must have a return statement
- Two methods can have the same name, but only if the parameters are different

Terms to know

Instance vs. Static
Constructor
private vs. public
Parameter
Local variable
Declaration
Initialization
Overriding
Overloading
Encapsulation
Information Hiding
Garbage Collection

Important Containers

- ArrayList
- TreeSet
- TreeMap

- Also know about iterator for loop

Inheritance

- The keyword `extends` defines an is-a relationship between classes
  - If Foo extends Bar, then Foo is-a Bar
  - That means that variables of type Bar can also hold objects of type Foo
- The keyword `implements` also defines an is-a relationship, but between classes and interfaces
  - If Foo implements Comparable<Foo>, then Foo is-a Comparable<Foo>

Polymorphism and Dynamic Binding

- Polymorphism refers to the ability to call different code with the same method call depending on the type of the object
- Dynamic binding refers to waiting until runtime to determine what code a method call will go to
- Polymorphism is a concept
- Dynamic Binding is an implementation

Inheritance

- Interfaces provide a set of methods that must be written in anything that implements the interface
- Abstract classes provide partial implementations of a class, but it's left to anything that inherits from it to fill in the gaps

`instanceof`

- `foo instanceof Bar` evaluates to
  - true: only if foo holds an object that can be treated as a Bar
  - Compile error: if it is illegal for foo to hold something that could be treated as a Bar
  - false: if foo could hold something that can be treated as a Bar, but doesn't

`super`

- `super` refers to the parent class
  - In a constructor, the very first thing that happens is a call to the super constructor
  - In any instance method, `super` is the part of this object that comes from the parent class

Method Modifiers

- public – Anything can access it
- protected – Can only be accessed by things that inherit from the class
- private – Can only be accessed from within the class
- abstract – Must be overridden (only allowed in abstract classes)
- final – Cannot be overridden

Exceptions

- Catching exceptions
  - Use try/catch/finally
- Throwing exceptions
- Creating your own exceptions

Exceptions

- Unchecked exceptions
  - Anything that inherits from RuntimeException is an unchecked Exception
  - You don't need to declare that your method throws unchecked exceptions
- Checked exceptions
  - Checked exceptions are exceptions that don't inherit from RuntimeException

Enums

- An enum defines a finite number objects of the same type
  - Begins by explicitly enumerating those objects
  - After the semicolon, it's basically like any other class (except that the enumerated objects are the only ones that can exist)
- The toString() method returns the name of the object
- The static values() method returns an array of the enumerated values

Sample Problems

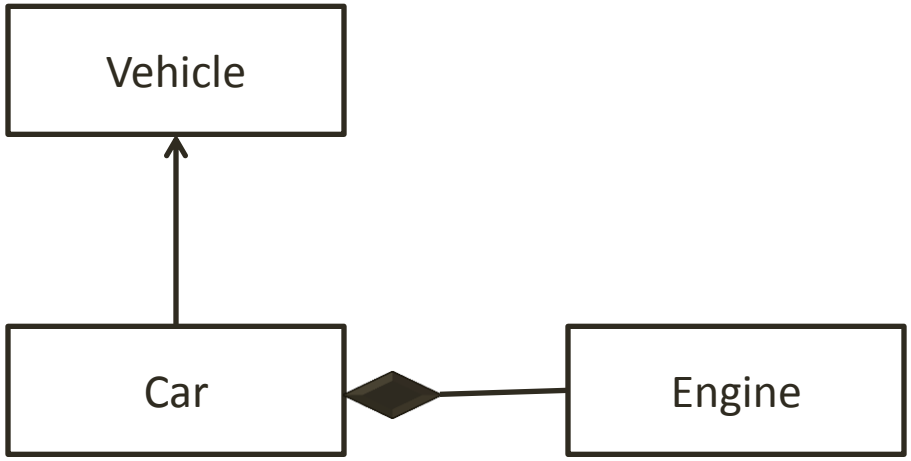
- You are given the class CarTest, shown below, that involves the multiple use of Car objects.

- Draw a UML diagram for the Car class described below.
- Write the java code for the Car class.
- State: A car object needs to store weight and engine size.
- Behavior: Car can have many methods. However, you will need to provide move method.
- move(): description of the method.
- public class CarTest {

```
        public static void main(String[] args) {
                Car car1 = Car(600, 1000)
        }
}
```

Sample Problems

- UML Class Diagrams



- (above diagram is incomplete, in example we will have complete diagram.)
- Write Car class specified by the diagram.

Sample Problems

- Write a method that takes in an int[] array and two ints a and b and returns the maximum value in the array between a (inclusive) and b (exclusive). If a or b are out of range of the array size, throw an ArrayOutOfBoundsException. If a is >= b, throw an InvalidRangeException. Comment the method with javadoc style comments.

Sample Problems

- Resistors are used in circuits and are measured in a unit called Ohms. In particular, two resistors can either be placed in series or be placed in parallel. If two resistors are placed in series, they are equivalent to a third resistor that is the sum of their resistances. (Thus, a 3 Ohm resistor in series with a 4 Ohm resistor is equivalent to a 7 Ohm resistor.) If two resistors with resistances R1 and R2 are placed in parallel, they are equivalent to a resistor with resistance (R1)(R2)/(R1+R2). (Thus, a 3 Ohm resistor in parallel with a 6 Ohm resistor is equivalent to a 2 Ohm resistor.) Complete the Resistor class below. When a Resistor object is printed out, its resistance should be followed by the string "Ohms". For example, a Resistor object storing 3.6 should print out as "3.6 Ohms". In the constructor, val is negative, throw the InvalidResistanceException and pass to it the string, "No Negative Resistors", otherwise just set the value of resist.

Sample Problems

```
public class Resistor {
        private double resist;
        public Resistor(double val) throws InvalidResistanceException {


       }

        public String toString() {

        }

       public Resistor series(Resistor other) throws InvalidResistanceException {

       }

        public Resistor parallel(Resistor other) throws InvalidResistanceException {

        }
}
```

Sample Problems

- Fix the errors in the following code:

```
int a=0;
for (int i=0; i<5 i++)
  a = a+i;
  sum += a;
System.out.println(sum);
```

- What is the output, once the errors are fixed?

Sample Problems

- Define an enum which stores the name of the twelve months of the year.

Sample Problems

- Write a class Animals that stores the weight and the color of an animal. The user should not be able to instantiate an Animal.
- Then write classes Cat and Snake which inherit from Animal. Cat should have a field for clawLength, and Snake should have a field for hibernationTime.
- Include appropriate constructors.

Sample Problems

- Write a method which takes in a a Collection of any type and computes and returns the 2 smallest values in the Collection.
- Enforce any required assumptions about the collection.

Sample Programs

- Parts of a class

Sample Problems

- Circle the Primitives.
  - int, String, ..
- Circle the Literals
  - …….

Sample Problems

- GUI: Questions on Swing classes
  - Containers, Components

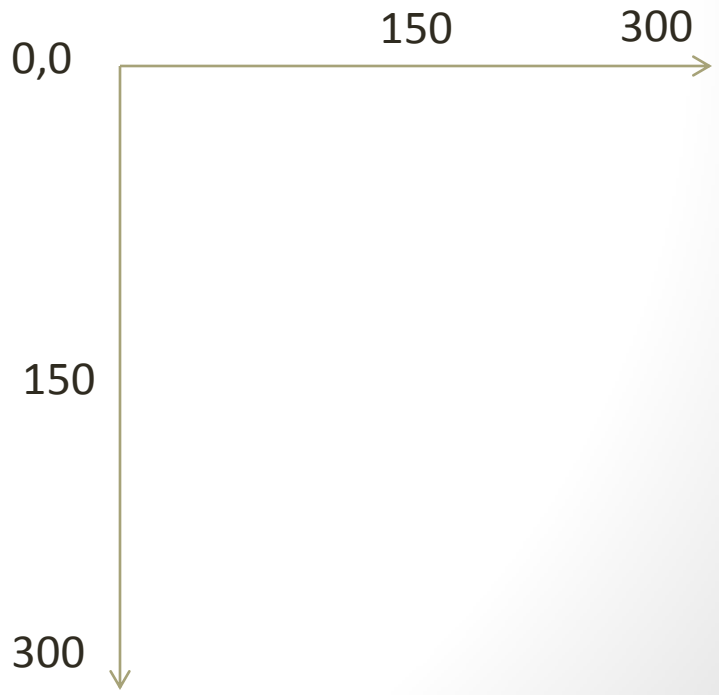
Sample Programs

- What will be drawn by the code in the paint method given below? (You may assume that all other supporting code is working properly.) Place your drawing below.

```
public void paint(Graphics g) {

    g.drawRect(10,10,50,100);
    g.clearRect(60,10,2,100);
    g.drawOval(70,10,50,100);
    g.drawLine(130,10,130,110);
    g.drawOval(130,10,50,50);

}
```



My Teaching Philosophy

- *"Give a man a fish, and he will eat for a day. Teach a man to fish and he will eat for a lifetime."*

- In this class, we did not give out fish! I was here to teach you how to fish.
- I hope by now most of you have learnt how to fish.

- Thanks!

- Contacts –
  - akghosh@cs.ucf.edu
  - Facebook, Linkedin – Arup Ghosh

Free Fish..