Final Exam

- Tuesday, March 16, 10:15 - 12:15
- Bravaco text, chapters 1-10, 12, 13, 16
- Anything from slides
- Anything from readings and assignments

Final Exam Topics

- Emphasis on material since midterm
- But exam is comprehensive and there may be questions on material from first half
  - Class coding and methods, public/private
  - Variable scope
  - Flow control
  - Java types and conversions
Final Exam Topics

- Static versus instance
  - Static methods called without object
  - Instance methods require an object
  - Static data is per class, not per object
  - Instance data is for each object
  - `this` reference cannot be used in static method

- Mutable and immutable objects
  - Immutable objects cannot change after creation
  - `String, BigInteger` are immutable
  - Mutable objects have instance data which can change
  - `StringBuffer, ArrayList` are mutable

Final Exam Topics

- Object references
  - Difference between primitive variable and object
  - Object creation with `new`
  - What assignment means for objects
  - What `==` means for objects
  - What it means for object variable to be null
  - Wrapper classes for primitive types
  - Garbage collection

- Event – Listener model
  - Relationship between objects
  - How one object knows about another
Final Exam Topics

- Arrays
  - Definition syntax
  - Initialization syntax
  - Element access syntax
  - Arrays are objects
  - Constant public `length` field of array
  - Valid index values for array access

- Multi-dimensional arrays

Final Exam Topics

- Inheritance
  - Inheriting and overriding methods
  - Polymorphism
  - Abstract classes and Interfaces

- Recursion
  - Recursive definitions
  - Recursive solutions to problems
  - Recursive programming
Final Exam Topics

- Abstract Data Types
  - Linked list
  - Stack
  - Use of object references in links
- Java collection classes
  - ArrayList, LinkedList, Stack
- Two’s Complement

Studying for the Final Exam

- Review material covered in book
- Review slides
- Review assignments
  - Understand model solution
  - Understand what is right and wrong with your solutions
- Review midterm
- Take sample final questions as serious test
Final Exam Question Format

- Multiple choice questions
- Programming questions
- Logic questions
- Design questions

Example Questions

Assume \( x \) and \( y \) are String variables with
\( x = "Hello" \) and \( y = \text{null} \).
The result of \( x.length() + y.length() \) is

- a) 0
- b) 5
- c) 6
- d) 10
- e) a thrown exception
If the statement `y=new String("Hello");` is executed, then the value of `(x==y)` is

a) true
b) false
c) `x` and `y` becoming aliases
d) `x` being set to the value `null`
e) a run time error

Example Questions

If the statement `y="Hello";` is executed, then the value of `x.equals(y)` is

a) true
b) false
c) -1
d) 0
e) a run time error
Example Questions

If the statement
\[
x.\text{replace('H', 'J').toLowerCase();}
\]
is executed, then the value of \(x\) is

\[
\begin{align*}
a) & \text{ "Jello"} \\
b) & \text{ "jello"} \\
c) & \text{ "jELLO"} \\
d) & \text{ "Hello"} \\
e) & \text{ a run time error}
\end{align*}
\]

Example Questions

If the statements
\[
\text{StringBuffer } s = \text{ new StringBuffer(x);} \\
\text{s.replace(0, 1, "J");}
\]
are executed, then the value of \(s\) is

\[
\begin{align*}
a) & \text{ "Jello"} \\
b) & \text{ "Hello"} \\
c) & \text{ "J"} \\
d) & \text{ a run time error}
\end{align*}
\]
Example Questions

In Java, arrays are

a) primitive data types
b) objects
c) interfaces
d) primitive data types if the type stored in the array is a primitive data type and objects if the type stored in the array is an object

What does this code do? (list is an array of int, temp is some int value, and c is an int initialized to 0)

```java
for(j=0; j<list.length; j++)
    if(list[j] < temp) c++;
```

a) finds smallest value and stores in temp
b) finds largest value and stores in temp
c) finds number of values equal to smallest in list
d) finds number of values in list less than temp
e) sorts values in list to be in ascending order
Example Questions

Which is true regarding outer and inner classes?

a) the outer class can access the inner class' private data  
b) the inner class can access the outer class' private data  
c) neither class can access any data of the other  
d) each class can access all of the other's data

The relationship between a class and an object is best described as

a) classes are instances of objects  
b) objects are instances of classes  
c) objects and classes are the same thing  
d) classes are programs while objects are variables
Example Questions

If an int array is passed as a parameter to a method `foo`, which of the following would properly define the parameter list for the method signature?

a) `foo(int[ ])`
b) `foo(a[ ])`
c) `foo(int[ ] a)`
d) `foo(int a)`

Example Questions

Assume an int array, `candy`, stores the number of candy bars sold by a group of 12 children where `candy[j]` is the number of candy bars sold by child `j`. Which of the following could be used to compute the total number of bars sold by the children?

a) `for(int j=1; j<12; j++) sum += candy[j];`
b) `for(int j=0; j<12; j++) sum = candy[j];`
c) `for(int j=0; j<12; j++) candy[j]++;
```
d) `for(int j=0; j<12; j++) sum += j;`
e) `for(int j=0; j<12; j++) sum += candy[j];`
Example Questions

What does the following code do?

```java
int value1 = 0, value2 = 0;
for(int j=0; j<12; j++)
    if(candy[j] > value1) {
        value1 = candy[j]; value2 = j;
    }
return value2;
```

a) returns total number of candy bars sold
b) returns total number of children who sold > 0 candy bars
c) returns largest number of candy bars sold by any child
d) returns index of the child who sold the most candy bars

---

Example Questions

A linked list that stores int values would be comprised of a group of Nodes. We could best define the Node by

a) ```java
class Node { Node next; }
```
b) ```java
class Node { int next; }
```
c) ```java
class Node { int data; Node next; }
```
d) ```java
class Node { int[] data; Node next; }
```
Example Questions

Following is a recursive implementation of a method to compute Fibonacci numbers:

```java
int fib(int n) {
    if (n == 0)      return 0;
    else if (n == 1) return 1;
    else             return fib(n-1) + fib(n-2);
}
```

What does `fib(2)` return? 1  \((\text{called 3 times})\)
What does `fib(3)` return? 2  \((\text{called 5 times})\)
How many times is `fib` called to compute `fib(4)`? 9 times

Example Questions

Give a recursive definition of the number of ways to draw two cards from a deck of \(n\) cards.

Base case(s):

For \(n=2\), \(\text{waysToDraw}(n)\) is 1

Recursive case:

For \(n>2\), \(\text{waysToDraw}(n)\) is \(\text{waysToDraw}(n-1) + (n-1)\)
Example Questions

Given the linked list of Nodes with int values and head as the first node, implement the method `add` to add the Node to the end.

```java
void add(Node node) {
    if (head == null) head = node;
    else {
        Node temp = head;
        while (temp.next != null)
            temp = temp.next;
        temp.next = node;
    }
}
```

Example Questions

Code a `recursive` method `maxFrom` that takes a node and returns the maximum value of all nodes from that point on.

```java
public class IList {
    private class IntNode {
        int val; IntNode next;
        IntNode(int v) {val=v; next=null;}
    }
    private IntNode head = null;
    public int max() { return maxFrom(head); }  
}
```
Example Questions

```java
int maxFrom(IntNode node) {
    if (node == null) {
        return 0;
    }
    else if (node.next == null) {
        return node.value;
    }
    else {
        int tailmax = maxFrom(node.next);
        if (tailmax > node.value) return tailmax;
        else return node.value;
    }
}
```

Example Questions

Code a method named `twiceArray` that takes an array of ints and returns an array of ints where the values are all double the original values. The originals should not be changed.

```java
public int [] twiceArray(int [] orig) {
    int [] twice = new int [orig.length];
    for (int i = 0; i < orig.length; ++i)
        twice[i] = 2 * orig[i];
    return twice;
}
```
Example Questions

The following code implements a class named `Table` that stores integer values where the value is the sum of the row index and twice the column index. Find all the errors in the code and indicate if they are compile time or runtime errors and describe each error.

```java
public class Table {
    private int[][] data;
    public Table(int size) {
        for (int i = 0; i < size; ++i)
            for (j = 0; j < size; ++j)
                data[i][j] = i + 2 * j;
    }
    public String toString() {
        String result;
        for (int i = 0; i <= data.length; ++i) {
            for (int j = 0; j <= data[i].length; ++j)
                result += i + "+2*" + j + "=" + data[j][i] + " ";
        }
        return result;
    }
}
```

- Need initialization (runtime): `data = new int[size][size];`
- Return value should be String (compile)
- `j` not declared (compile)
- Out of range, use `<` (runtime)
- Not a method (compile)
- Result not initialized (runtime)
- Wrong order of indices (runtime)
Example Questions

Fill in the code for classes Part and Labor that implement the interface Billable.

```java
public interface Billable {
    public final NumberFormat fmt = NumberFormat.getCurrencyInstance();

    // Compute charge for this item
    public double charge();

    // Realize the item as a String
    public String toString();
}
```

Interface Question Continued..

This driver shows use of Part and Labor.

```java
public class BillTest {
    public static void main(String args[]) {
        Billable p1 = new Part("oil filter", 1, 5.95);
        Billable p2 = new Part("oil", 5, 1.79);
        Billable l = new Labor(.5, 25.00);

        double total = 0.0;
        total += p1.charge();
        total += p2.charge();
        total += l.charge();

        System.out.println("Total charge is "+Billable.fmt.format(total));
        System.out.println("Detail of bill:");
        System.out.println(p1 + "\n" + p2 + "\n" + l);
    }
}
```
public class Part implements Billable {
    private String desc;    // Description of item
    private int quantity;   // Number of items
    private double price;   // Item price

    Part(String d, int n, double p) {
        desc = d; quantity = n; price = p;
    }

    public double charge() {    return quantity * price; }

    public String toString() {
        return desc + ": " + quantity + " at " + fmt.format(price);
    }
}

public class Labor implements Billable {
    private double hours;   // Hours of labor
    private double rate;    // Hourly labor rate

    Labor(double h, double r) {
        hours = h; rate = r;
    }

    public double charge() { return hours * rate; }

    public String toString() {
        return "Labor: " + hours + " hours at " + fmt.format(rate);
    }
}
Exam Strategy

- Do the multiple choice first
  - Over half of the points
  - Probably easiest, but read carefully!
- Pace yourself
- Leave time to check your work
- Relax