Reading:
   1. Deitel & Deitel, Chapters 10

Objectives:
1. Continue discussing Object Oriented Programming
2. Final Exam Review

Concepts:
1. More About Polymorphism

References:
2. http://java.sun.com/j2se/1.4.2/docs/api/
More About Polymorphism:

This lecture continues the discussion of polymorphism from the previous lecture. Recall that polymorphism is when a reference variable may at different times in a program refer to objects of different (but related) classes.

**Topic:**

- Arrays of related objects.

Let us use the hierarchy of classes used in the previous chapter to illustrate these ideas. The diagram shows that the class Card is an abstract class (which, therefore, cannot be instantiated.)

Card has a single abstract method, greeting(). The other three classes are ordinary classes which can be instantiated. They inherit from Card, so each must define a greeting() method which is not abstract.

**Extended Hierarchy:**

The Card hierarchy is extended by adding two new classes:

- A YouthBirthday birthday card for young people.
  - This card will add the line "How you have grown!" to the usual birthday greeting.
- An AdultBirthday birthday card for old people.
  - This card will add the line "You haven't changed at all!" to the usual birthday greeting.

The class Birthday is the parent for these two new classes. The diagram omits the other two classes, but assume that they are still defined. Each of the new classes would ordinarily inherit the greeting() method from Birthday. But we want to override that method with a specialized method in each of the new classes.
Here is the class definition of Birthday from the previous chapter:

class Birthday extends Card
{
    int age;

    public Birthday ( String r, int years )
    {
        recipient = r;
        age = years;
    }

    public void greeting()
    {
        System.out.println("Dear "+ recipient + ",\n");
        System.out.println("Happy "+ age + "th Birthday\n\n");
    }
}

Array of Related Objects

Now any object that belongs in the hierarchy can fit into any slot of the array. This is like
the greeting card display at the drug store that holds a different selection of cards in
different seasons of the year. Here is a program snippet:

Card[] cards = new Card[12];

cards[0] = new YouthBirthday( "Valerie", 7 );
cards[1] = new AdultBirthday( "Walter", 47 );
cards[2] = new Birthday( "Zoe", 30 );
cards[3] = new Holiday( "Kelly" );
cards[4] = new Valentine( "Jill", 42 );

for ( int j=0; j <= 4; j++ )
    cards[j].greeting();
This code will create 5 cards objects of various varieties and put them into the array. Then it will ask each object in the array to write out its own version of the greeting. This will work fine:

Dear Valerie,
Happy 7th Birthday
How you have grown!

Final Exam Review:

The final exam will be 2 hours long. The final exam is cumulative but the emphasis will be on the material which was covered after the second midterm. Review the Midterm 1 (attached copy), and Midterm 2 (attached copy). Questions for previous topics go into more depth now that you have had more practice programming. Questions for new topics also require the use of previous programming constructs.

LAB: There is no lab assignment for this week. Students are encouraged to use their lab time to work on their project. TAs will be available during this lab session to answer any questions students have.
1. Briefly describe the difference between a Java applet and an Application (5):

The main difference between JAVA applets and applications are its usage. applet is primarily implemented onto a website, a html file, and cannot write to files. On the other hand, a JAVA application is used just like a regular application on a computer with full functionality. JAVA applications and applet are different in their coding style.

2. Use pseudocode to describe how to make a bed (10):

Answers may vary, read pseudocode to determine correctness

3. Write a Java program segment to execute the following pseudocode (15):

Array a = {1,2,3,4,5}.
Increment each value of a by 1 when the number is even and by 2 when the number is odd.

```java
int array = {1,2,3,4,5};
for (int i=0; i< array.length; i++){
    if(array[i] % 2 == 0 ) array[i] = array[i]+1;
    else array[i] = array[i]+2;
}
```

5. Briefly explain what the following program is doing and add comments to explain every statement (15):

```java
public class Test {
    public static void main( String args[] ) {
        final int ARRAY_LENGTH = 10;
        int array[];
        array = new int[ ARRAY_LENGTH ];
        for ( int counter = 0; counter < array.length; counter++ )
            array[ counter ] = 2 + 2 * counter;
        String output = "Index\tValue\n";
        for ( int counter = 0; counter < array.length; counter++ )
            output += counter + "\t" + array[ counter ] + "\n";
        System.out.println( output );
    }
}
```

This program instantiates an array to have the values: 2, 4, 6, 8, 10, 12, 14, 16, 18, 20. Then it outputs the values of the array.
Midterm II:

1. Carefully read the following code.
   ```java
   int limits = 0;
   String message = "llnos";
   String new_message = new String[20];
   int n = message.length();
   int j = 0;
   for (i=0; i<n; i++)
       if (i == limit || i % 3 == limit)
           new_message[i] = "i";
           n++;
       else
           new_message[i] = message[j];
           j++;
   }
   System.println(new_message.substring(0, n));
   ```

   What is the output?
   illinois

   If instead the following values were given:
   message = "msssspp" & limits = 1
   What is the output?
   mississipp

2. Name and describe 4 of the functions that commonly must be overridden when working with Applets.
   - `init( )` – Initialization when applet is loaded.
   - `start( )` – Where the applet start after it is initialized.
   - `stop( )` – Runs when the user exits the page.
   - `paint( )` – What applet uses to display something on screen.
   - `destroy( )` – Allows the applet to clean up after itself.

3. Define and describe the differences between recursion and iteration.
   - **Recursion** – A programming method in which a routine calls itself.
   - **Iteration** – A programming method where there is a pass through a group of instructions, which is repeated using loops.

   **Iteration**
   - Uses repetition structures (for, while or do…while)
   - Repetition through explicitly use of repetition structure
   - Terminates when loop-continuation condition fails
   - Controls repetition by using a counter
Recursion
- Uses selection structures (if, if…else or switch)
- Repetition through repeated method calls
- Terminates when base case is satisfied
- Controls repetition by dividing problem into simpler one
- More overhead than iteration
- More memory intensive than iteration
- Can also be solved iteratively
- Often can be implemented with only a few lines of code

4. Describe through an example the Object Oriented concept of Inheritance. Make sure to explain the relationships between your objects.
   For example, mountain bikes, racing bikes, and tandems are all kinds of bicycles. Mountain bikes, racing bikes, and tandems are all subclasses of the bicycle class. Similarly, the bicycle class is the superclass of mountain bikes, racing bikes, and tandems.