Name _____________________________________________

CS 110
Practice Final Exam
originally from Winter, 2003

Instructions: closed books, closed notes, open minds,
3 hour time limit.

There are 4 sections for a total of 49 points.

Part I: Basic Concepts, Boolean Logic (pp. 2-3) 14 pts
Part II: Writing Methods (pp. 4-8) 22 pts
Part III: Linear and Binary Search (p. 9) 8 pts
Part IV: Inheritance and Dynamic Binding (pp. 10-11) 5 pts
Part I – Basic Concepts and Boolean Logic  (14 points total)

1. Match each programming concept on the left with the appropriate description on the right.  
   
   (6 points)

   _____ class  
   _____ interface  
   _____ abstract class  
   _____ class documentation  
   _____ access identifier  
   _____ instance variable  
   _____ class variable  
   _____ local variable  
   _____ scope  
   _____ constructor  
   _____ method  
   _____ method signature  
   _____ method invocation  
   _____ parameter type  
   _____ formal parameter  
   _____ actual parameter  
   _____ return type  
   _____ inheritance  
   _____ subclass  
   _____ superclass  
   _____ dynamic binding  
   _____ responsibility  
   _____ collaboration  
   _____ linear (sequential) search  
   _____ binary search  
   _____ aggregation  
   _____ recursion  

   a) descriptions not provided in practice exam  
   b)  
   c)  
   d)  
   e)  
   f)  
   g)  
   h)  
   i)  
   j)  
   k)  
   l)  
   m)  
   n)  
   o)  
   p)  
   q)  
   r)  
   s)  
   t)  
   u)  
   v)  
   w)  
   x)  
   y)  
   z)  
   aa)
2. Circle all of the code fragments below that are logically equivalent to the following statement: (6 pts)

```java
for ( int i = 10; i > 0; i-- )
    System.out.println("Hi!");
```

a) ```java
i = 10;
while ( i > 0 )
{
    i--;
    System.out.println("Hi!");
}
```

b) ```java
for ( i=1; i < 11; i++ )
    System.out.println("Hi!");
```

c) ```java
i = 11;
do
{
    System.out.println("Hi!");
i--;
}
while ( i > 0 );
```

d) ```java
for ( i=0; i < 10; i++ )
    System.out.println("Hi!");
```

e) ```java
i = 0;
while ( i < 10 )
{
    System.out.println("Hi!");
i--;
}
```

f) ```java
for ( i=0; i < 20; i+=2 )
    System.out.println("Hi!");
```

3. Are the following two code segments logically equivalent? Explain your answer. (2 points)

```java
if ( i > 0 )
    return false;
else if ( j <= i )
    return false;
else return true;
```

```java
if ( i > 0 )
    return false;
if ( j <= i )
    return false;
else return true;
```

The two code segments are not logically equivalent. In the first segment, the `else if` statement writes the `false` result back to the `return` statement, potentially changing the result of the `if` condition. In the second segment, the `false` result of the `if` statement is not written back to the `return` statement, and thus it is not modified. Therefore, the order of evaluation matters in the first segment, whereas the second segment is order-independent.
4. Write a method, `isReverse`, that takes two `ArrayList` parameters, `x` and `y`, and returns true if the values in `y` are the same as those in `x`, except in the opposite order. That is, `isReverse` should return true if the two lists have the same number of elements and the first element of `x` is the last element of `y`, the second element of `x` is the second to last element of `y`, and so on. The method should return `false` otherwise.

    public boolean isReverse(ArrayList<String> x, ArrayList<String> y)
5. The standard Java `java.util.GregorianCalendar` class has a method that determines whether or not a given year is a leap year.

   ```java
   public boolean isLeapYear(int year)
   ``

Write a driver that tests this method. Note that any year that is divisible by 4 is a leap year except those years that are divisible by 100 but not by 400 (for example, 1900 is not a leap year but 2000 is a leap year). Explain why you included each test case. (4 Pts)

```java
import java.util.GregorianCalendar;
public class MyDriverClass {
    public static void main(String[] args) {
        GregorianCalendar cal = new GregorianCalendar();
        System.out.println("Year: Expected   Actual");
        System.out.println("2003: false   " + cal.isLeapYear(2003));
    }
}
```
Questions 6 - 8 refer to the Date, Flight, and Airport classes provided on the supplemental sheet. You may assume that all methods you are not writing have been implemented correctly.

6. Implement the Flight method late, which returns true if the flight has arrived (the actual time is initialized) and the actual arrival time is later than the estimated arrival time. The late method also returns true if the flight has not arrived, but the current time is later than the estimated arrival time. In all other cases late returns false. Note that you can get the current date and time by constructing a new Date object. (4 pts)

```java
public class Flight
{
    public boolean late()
    {
```
7. Implement the Airport method `nbrLateFlights`, which returns the number of the specified airline’s flights that are late today. (5 pts)

```java
public class Airport {
    public int nbrLateFlights(String airline) {
```
8. Implement the `Airport` method `mostLateFlights`, which returns the name of the airline with the most late flights today. 

(5 pts)

```java
public class Airport {
    public String mostLateFlights() {
```
Part III – Sequential and Binary Search  (8 points total)

9. We have discussed linear and binary search techniques. Consider an array, A, containing the following 16 elements:

<table>
<thead>
<tr>
<th>index</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>11</td>
<td>13</td>
<td>25</td>
<td>26</td>
<td>29</td>
<td>30</td>
<td>31</td>
<td>32</td>
<td>43</td>
<td>44</td>
<td>47</td>
<td>55</td>
<td>59</td>
<td>60</td>
<td>62</td>
<td>73</td>
</tr>
</tbody>
</table>

a) How many elements must we examine to determine whether the value 30 is in array A using a linear search? (1 pt)

   1 2 3 4 5 6

b) How many elements must we examine to determine whether the value 30 is in array A using a binary search? (1 pt)

   1 2 3 4 5 6

c) How many elements must we examine to determine whether the value 11 is in array A using a linear search? (1 pt)

   1 2 3 4 5 6

d) How many elements must we examine to determine whether the value 11 is in array A using a binary search? (1 pt)

   1 2 3 4 5 6

10. Suppose we define a structure, called a linked list, which can store all of the same types of information that an array can. The only way to access the elements in a linked list is to start at the beginning of the list and step through all the items in it. (We say a linked list does not have random access because you cannot access the $i$\textsuperscript{th} element directly; you have to start at the beginning and step through until you get to the $i$\textsuperscript{th} element.) Pictorially, a linked list may look like the following:

   Start: 11 -> 13 -> 25 -> 26 -> 29 -> 30 -> |

For each of the following types of data, would a linear or binary search be the better choice? (4 pts)

a) an array of sorted data:

b) an array of unsorted data:

c) a linked list of sorted data:

d) a linked list of unsorted data:
Part IV – Inheritance and Dynamic Binding (5 points total)

11. What inheritance relationships would you establish among the following classes?

    Student
    Professor
    TeachingAssistant
    DepartmentChair
    Person
    Course
    Seminar
    Lecture
    ComputerLab

Draw pictures (they could look like trees or like object diagrams) of the various classes and the inheritance relationships. Label the lines showing the relationships to make it clear the type and direction of the relationships. Do not try to indicate all the other relationships that might exist among the classes.

(5 pts)