Points off	1	2	3	4	5	Total off	Net Score

CS 307 – Midterm 1[corrected] – Spring 2008

Your Name_____

Your UTEID _____

Circle yours TA's name: Ruchica Mario Vishvas

Instructions:

1. Please turn off or silence your cell phones
2. There are **5** questions on this test.
3. You will 2 hours to complete the test.
4. You may not use a calculator or any other electronic devices while taking this test.
5. Please make your answers legible.
6. When code is required, write Java code.
7. When writing a method, assume the preconditions of the method are met.
8. When writing a method you may add helper methods if you wish.
9. When you complete the test show the proctor your UTID and give them the test and any scratch paper. Please leave the room quietly.

1. (2 points each, 30 points total) Short answer questions. Place your answers on the attached answer sheet. For code sample state the output. If the code would cause a syntax error answer "syntax error". If it would cause an exception error answer "exception". If it would result in an infinite loop answer "infinite loop".

A. What is the output of the following code?

```
int x = 2;
int y = 7;
int z = y / x + y * 2;
System.out.println( z );
```

B. What is the output of the following code?

```
double a = 5 / 2;
double b = a;
a++;
System.out.println( b );
```

C. Consider the following method.

```
public int process(int[] data, int m1, int m2){
     int t = 0;
     for(int i = 0; i < data.length; i++){
          if( data[i] < m1 || data[i] > m2 ){
               t++;
          }
     }
     return t;
}
```

What is printed out by the following code?

```
int[] list = {5, 8, 13, 10, -5, 0, 9, 7, 12};
System.out.println( process(list, 3, 10) );
```

D. This question uses the method named `process` from part C.
What is the output of the following code?

```
int[] list2 = null;
System.out.println( process(list2, 0, 12) );
```

E. Consider the following method.

```
public void manip(int[][] mat){
     for(int i = 0; i < mat[0].length; i++){
          for(int j = i + 1; j < mat.length; j++){
               mat[j][i] += mat[j - 1][i];
          }
     }
}
```

Consider the following code.

```
int[][] cells = {{1, 2, 3}, {-4, 5, 7}, {2, 5, 4}};
manip( cells );
```

What are the contents of the 2d array of ints named `cells` after the code above completes?

For questions F – M consider the following classes.

```
public class School{
      private int numStudents;

      public School(int numStudents){
            this.numStudents = numStudents;
      }

      public int getSize(){
            return numStudents;
      }

      public String toString(){
            return "Size: " + this.numStudents;
      }

      public void graduate(){
            this.numStudents -= this.numStudents / 4;
      }

}

public class HighSchool extends School{
      private String classification;

      public HighSchool(int numStudents, String classification){
            super( numStudents );
            assert validClassification( classification );
            this.classification = classification;
      }

      public String toString(){
            return "class: " + this.classification;
      }

      public static boolean validClassification(String s){
            boolean result = s != null && s.length() >= 0 && s.length() <= 5;
            if( result )
                  result = s.equals("A") || s.equals("AA") || s.equals("AAA") ||
                        s.equals("AAAA") || s.equals("AAAAA");
            return result;
      }
}

public class University extends School{
      private String name;

      public University(int numStudents, String name){
            super(numStudents);
            this.name = name;
      }

      public String toString()
      {     return this.name + ", " + super.toString();   }
}
```

F. What is the output when method `two` is called?

```
public static void one(int pop){
     pop *= 2;
     System.out.print( pop + " " );
}

public static void two(){
     School rr = new School(1000);
     one( rr.getSize() );
     System.out.println( rr.toString() );
}
```

G. What is the output when method `four` is called?

```
public static void three(School s){
     s.graduate();
     System.out.print( s.toString() + " " );
}

public static void four(){
     School gt = new School(1000);
     three(gt);
     System.out.println( gt.toString() );
}
```

H. What is output when method `six` is called?

```
public static void five(School s){
     s.graduate();
     s = new School( 1000 );
}

public static void six(){
     School cp = new School( 100 );
     five( cp );
     System.out.println( cp );
}
```

I. What is the output of the following code?

```
HighSchool schs = new HighSchool(1000, "BB");
System.out.println( schs.toString() );
```

J. What is the output of the following code?

```
School ucsd = new University( 1000, "UCSD" );
ucsd.graduate();
System.out.println( ucsd.toString() );
```

K. What is the output of the following code?

```
HighSchool katy = new School(100);
System.out.println( katy.toString() );
```

L. What is the output of the following code?

```
HighSchool austinHigh = new HighSchool(1000, "AAAAA");
austinHigh.graduate();
System.out.println( austinHigh.toString() );
```

M. What is the output of the following code?

```
University u = new University(1000, "UT");
School s = u;
u.graduate();
System.out.println( s.toString() );
```

For questions N and O consider the following interface.

```
public interface TimeStamp{

      public int getTime();
}
```

N. Does the following class compile?

```
public class Book implements TimeStamp, Comparable{

      public int compareTo(Object other){
            return this.getTime() - ((TimeStamp)other).getTime();
      }
}
```

O. Consider the following class.

```
public class Trade implements TimeStamp, Comparable{
      public Trade(){ // details not shown }
      // other implementation not shown, but class will compile
}
```

What data types can replace **<*1>** in the code below so that the code will compile?

```
<*1> sc = new Trade();
```

2. Arrays and Classes. (20 Points) In class we developed a class to represent a List of `int`s to demonstrate encapsulation and some of the details of building a class in Java. As discussed in class the internal storage container may have extra capacity.

Complete a method for the `IntList` class, `moveMinToFront`. This is an instance method in the `IntList` class that finds the minimum value in the list and moves it to the front of the list. If there is more than one value equal to the minimum the value closest to the front of the list, index 0, is moved to the front.

This method does not change the size of the list, but it may result in some elements being moved. The relative order of all elements except the first occurrence of the minimum must remain the same.

You may not use any other methods in the `IntList` class unless you write them and you may not use methods from the Java `Arrays` class or the Java `ArrayList` class.

Here are some examples of the result of a call to `moveMinToFront` on various lists. The element at the far left is at position 0 in the `IntList`.

Original list	Resulting list
`[0, 3, -2, 4, 5]`	`[-2, 0, 3, 4, 5]`
`[3]`	`[3]`
`[0, 3, -2, 4, 5, -2, 1]`	`[-2, 0, 3, 4, 5, -2, 1]`
`[-2, 0, 0, 0, 3, 3]`	`[-2, 0, 0, 0, 3, 3]`
`[4, 3, 2, 1]`	`[1, 4, 3, 2]`

```
public class IntList{

     private int[] iValues;
     private int iSize;

     public int size(){
          return iSize;
     }

     /* pre: size() > 0
        post: first occurrence of min value moved to front of list
     */
     public void moveMinToFront(){
          assert size() > 0 : "Failed precondition.";
          // Complete this method on the next page.
```

```
/* pre: size() > 0
   post: first occurrence of min value moved to front of list
*/
public void moveMinToFront(){
     assert size() > 0 : "Failed precondition.";
     // Complete this method.
```

3. Arrays and Classes. (20 points) In class we altered our `IntList` class to make in a more generic `List` class. This demonstrated some of the features of inheritance and polymorphism in Java. As discussed in class the internal storage container may have extra capacity.

Complete an instance method for the `List` class `isReverse`. This method returns true if the calling object has the same elements as the explicit parameter, except the elements are in reverse order.

Your method is not to alter either `List`.

You may not use any other methods in the `List` class unless you write them (Except for the size method.) and you may not use methods from the Java `Arrays` class or the Java `ArrayList` class.

Here are some examples of the result of a call to `isReverse` on various lists. Assume the integer values shown are actually `Integer` objects. Note the `List` objects may contain values equal to `null`.

Calling Object	Explicit Parameter	returned value
`["A"]`	`["A"]`	`true`
`["A"]`	`["A", "B"]`	`false`
`["A", "B"]`	`["B", "A"]`	`true`
`["A", 1]`	`[1, "A"]`	`true`
`["A", 1]`	`[2, 1]`	`false`
`[null, "A", "B"]`	`["B", "A", null]`	`true`
`[null, "A"]`	`["B", null]`	`false`

```
public class List{

     private ~~i~~Object[] iValues;
     private int iSize;

     public int size(){
          return iSize;
     }

     /* pre: other != null
        post: return true if this List contains the same elements
          as other in reverse order, false otherwise.
     */
     public boolean isReverse(List other){
          assert other != null : "Failed precondition.";
          // Complete this method on the next page.
```

Recall that in the `List` class you have access to all `List` objects' private instance variables, not just the calling object's.

```
/* pre: other != null
   post: return true if this List contains the same elements
     as other in reverse order, false otherwise.
*/
public boolean isReverse(List other){
     assert other != null : "Failed precondition.";
     // Complete this method.
```

4. 2D arrays. (15 points) Complete a method that determines if the floor of a room is completely covered in paint or not. The floor is modeled as a 2d grid as shown below. Each cell can either be clean or covered with paint.

Initially the entire floor is clean. Paint balls are being tossed into the room from the top, left corner. Each paintball lands in a particular square and covers that square with paint. Each ball also splatters to the right and down. Each ball splatters based on its size. For example, a size 1 ball does not splatter, it only covers the cell it lands in. A size 2 ball covers the cell it lands in and the cell to the right and below.

For example, assume a size 1 ball lands in row 0, col 1. It only covers that cell.

	X_1			

Now assume a size 2 ball lands in row 1, col 1. It covers that cell and the cells at row 1, col 2 and row 2, col 1. The landing cell is shown with X_2 and the splattered cells are shown with S_2.

	X_1			
	X_2	S_2		
	S_2			

Now assume a size 4 ball (marked X_3)lands in row 0, col 2. It covers the cells shown below. Note the paint balls only splatter into cells in the same row or column as the cell that is hit.

	X_1	X_3	S_3	S_3
	X_2	S_3		
	S_2	S_3		
		S_3		

Note, a cell is either covered in paint or it is not. There is no measure of how many times it has been hit or splattered.

Complete a method that given the size of the floor and an array of `PaintBall` objects, returns true if the floor is completely covered with paint, false otherwise. The public portion of the `PaintBall` class is shown below.

```
public class PaintBall{
     //constructors not show

     // returns the row this PaintBall lands in. The returned value
     // will be greater than or equal to 0.
     public int getRow()

     // returns the column this PaintBall lands in. The returned value
     // will be greater than or equal to 0.
     public int getCol()

     // returns the size of this PaintBall. The returned value will be
     // greater than 0.
     public int getSize()
}
```

Note, it is possible that a `PaintBall` will be outside the bounds of a given room, in which case it hits the wall and does not get any paint on the floor.

Complete the following method. You may assume the preconditions of the method are met. Do not write assert statements to check preconditions.

```
/* pre: numRows > 0, numCols > 0, hits != null, no elements of
     hits = null. numRows and numCols specify the size of the room.
   post: return true if the floor is completely covered in paint,
     false otherwise.
*/
public boolean isCovered(int numRows, int numCols,
                         PaintBall[] hits){

// Complete the method on the next page.
```

```
/* pre: numRows > 0, numCols > 0, hits!= null, no elements of
     hits = null. numRows and numCols specify the size of the room.
   post: return true if the floor is completely covered in paint,
     false otherwise.
*/
public boolean isCovered(int numRows, int numCols,
                         PaintBall[] hits){
```

5. (Implementing classes, 15 points)

Complete a class that models a University course.

A course has the following properties:

- the number of credits the course is worth. This cannot be negative
- the course number. This will be an integer value between 0 and 99,999 inclusive.
- The department the course is taught by. This will be a String.

Complete a complete `Course` class that has the following properties.

- Instance variables to store all of the relevant information for a course as explained above.
- A constructor to initialize all values based on parameters.
- A method to change the number of credits a course is worth.
- A toString method.

Complete the class below. You may add other methods if you wish. State your preconditions and use `assert` statements to check them.

```
// more room on the next page if necessary
```

```
// more room for the Course class if necessary
```

Scratch paper

Question 1 answer Sheet

Name_____

A. _____

B. _____

C. _____

D. _____

E. _____

F. _____

G. _____

H. _____

I. _____

J. _____

K. _____

L. _____

M. _____

N. _____

O. _____

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper

Scratch Paper