Instructions:

KEEP TEST BOOKLET CLOSED UNTIL YOU ARE INSTRUCTED TO BEGIN.

This exam is double sided (front and back)!

No calculators, notes, textbooks, or any other aids are allowed except writing utensils (pens, pencils, crayons, erasers, etc.) or other aids provided to you by the instructor. If you need extra scratch paper, please pick it up from the front of the class.

You should also be provided with an appendix that provides helpful documentation.

All answers must be submitted on (or attached to) this exam sheet. All answers must be clearly legible.

Commenting code: Comments aren’t required, but if you provide good comments, they can earn you partial credit if your code isn’t correct.
Example True or False (10 points):
Instructions: Circle either True or False based on the validity of the statement.

1. In object oriented programming, the superclass inherits properties from the subclass.
   
   true  false

[There are usually 5-10 questions of this type]
Example Multiple Choice (10 points):

Instructions: Circle the letter of the best answer.

1. The data for which of these data types is stored in the Heap memory (select all that apply):
   A. boolean
   B. int
   C. double
   D. String
   E. int[]

   [There are usually 5-10 questions of this type]
Example Short Answer (10 points):

Instructions: In your own words, answer the questions as best as possible in one or two sentences.

1. What is the difference between a Class and an Object.

[There are usually 5-10 questions of this type]
Example Entomology - Study of Bugs (10 points):
Instructions: Find 3 bugs in the following program (there are more than three):
1) Specify the line number of the bug (Note: there can be more than one bug per line)
2) Check the box next to the type of error; either “Logic” for logic errors or “Other” for (syntax, type, etc.).
3) Write a short explanation or fix for the error.

```java
/**
 * Exercise06_03: Count occurrences of numbers entered (values between
 * <code>START_NUM_RANGE</code> and <code>END_NUM_RANGE</code>)
 */
public class Exercise06_03 {
    public static final int START_NUM_RANGE = 1;
    public static final int END_NUM_RANGE = 100;

    /**
     * Main program driver
     * @param args is ignored
     */
    public static void main(String[] args) {
        // Initialize counter array (using one extra index to simply logic)
        int[] number_counter = new int(END_NUM_RANGE + 1);

        // Get user inputed values
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter integers between "+ START_NUM_RANGE + " and "+ END_NUM_RANGE + ": ");
        int input = scanner.nextInt(); // get next input
        while (input != 0) {
            number_counter[input] += 1; // update counter
            input = scanner.nextInt(); // get next input
        }

        // Print out count for values that were encountered
        for (int i = 0; i < number_counter.length(); i++) {
            if (number_counter[i] == 0) {
                System.out.println(i + " occurs " + number_counter[i] + " time");
            } else if (number_counter[i] > 1) {
                System.out.println(i + " occurs " + number_counter[i] + " times");
            }
        }
    }
}
```

<table>
<thead>
<tr>
<th>Bug #</th>
<th>Line #</th>
<th>Type of Error</th>
<th>Explanation or Fix</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>[ ]</td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>[ ]</td>
<td>Logic</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>[ ]</td>
<td>Logic</td>
<td></td>
</tr>
</tbody>
</table>
Example What is the Output (10 points):

Instructions: What is the output if the following code is run?

```java
public class InsertionSort {
    public static void main(String[] args) {
        int[] ints = { 9, 5, 7, 1, 3 };  
        
        int numUnsorted;
        int itemToCompare;
        int numSorted;
        
        // move unsorted items their proper spot in the sorted part of the array
        for (numUnsorted = 1; numUnsorted < ints.length; numUnsorted++) { // skip first item
            itemToCompare = ints[numUnsorted];
            // move value to proper position in sorted list
            for (numSorted = numUnsorted - 1; (numSorted >= 0) && (itemToCompare < ints[numSorted]); numSorted--)
            {
                ints[numSorted + 1] = ints[numSorted];
            }
            ints[numSorted + 1] = itemToCompare;
            
            // After every item is moved, print out the new array
            for (int i : ints) {  
                System.out.print(i + " ");
                System.out.println();
            }
        }
    }
    
    /**
     * Takes and array and swaps values from <code>index1</code> and <code>index2</code>
     * @param ints - array of values
     * @param index1 - one of the indexes to swap the contents of
     * @param index2 - the second index to swap the contents of
     */
    public static void swap(int[] ints, int index1, int index2) {
        int temp = ints[index1];
        ints[index1] = ints[index2];
        ints[index2] = temp;
    }
}
```

Answer:
Example Comment the Code (10 points):

Instructions: The following code was written, but the documentation was left out. Analyze the code to determine what each function does and then what the program as a whole does. Fill in the missing comments. Be as specific as possible. Remember that Javadocs should explain WHAT the code does, and inline comments are generally best if they explain why the code is doing what it does.

```java
/**
 *
 *
 *
 */
static int[] dostuff(int size, int min, int max)
{
    Random rand = new Random();
    // ______________________
    int[] ints = new int[size];
    // ______________________
    for (int i = 0; i < ints.length; i++) {
        int nextRandom = rand.nextInt(max-min + 1);
        // ______________________
        nextRandom += min;
        // ______________________
        ints[i] = nextRandom;
    }
    return ints;
}

/**
 */
static int doSomething(int[] ints) {
    int sum = 0;
    // ______________________
    for (int i = 0; i < ints.length; i++) {
        sum += ints[i];
    }
    // ______________________
    return sum;
}
```
Example Fill in the Code (10 points):

Instructions: Fill in the missing code.

```java
public class Player {
    String name;
    int wins, losses;  // Number of wins and loses this player has had
    int health = 100;  // Percentage from 0-100, if 0, player is dead

    Player() {
        name = "Player";
    }

    /**
     * Copy constructor
     *
     * @param player to make a copy of
     */
    public Player(Player player) {
        // Rest of code cut for brevity
    }

    /**
     * @return a String representation of this player
     */
    public String toString() {
        // Rest of code cut for brevity
    }

    /**
     * Determines if this player is equivalent to another player
     *
     * @param player to compare this player to
     * @return true if this player and <code>player</code> have all equivalent data, otherwise false
     */
    public boolean equals(Player player) {
        // Rest of code cut for brevity
    }
}
```
Example Write the Code:

In java code, write a class that has the following data and methods associated with it:

Ball:
---data--------------
- xPosition: int
- yPosition: int
- xSpeed: int
- ySpeed: int
- size: int
---methods--------
+ Ball()
+ Ball(xPosition: int, yPosition: int, xSpeed: int, ySpeed: int, size: int)
+ updatePosition()
+ reverseSpeedY()
+ reverseSpeedX()
Example Write the Code:

In Java code, write the following method that implements Selection Sort:

```java
public static void selectionSort(int[] array) {
}
```