1	10/
2	10/
3	60/
4	10/
5	20/
6	30/
7	10/
	150/

CMSC 331 Second Midterm Exam

Name: _____ Username:_____

You will have seventy-five (75) minutes to complete this closed book exam. Use the backs of these pages if you need more room for your answers. Describe any assumptions you make in solving a problem. We reserve the right to assign partial credit, and to deduct points for answers that are needlessly wordy.

0. Warm up (0 pts)

You're trying to get to Truthtown. You come to a fork in the road. One road leads to Truthtown (where everyone tells the truth), the other to Liartown (where everyone lies). At the fork is a man from one of those towns -- but which one? You get to ask him one question to discover the way. What's the question? **Which town are you from?**

1. Bugs (10 pts)

The complete program to the right is supposed to compute and print out the product of two numbers, **5** and **8**. It contains one (major) syntax error and one logic error. Correct both by making minimal chages (e.g., don't completely rewrite the code – for example, don't redo the while loop as a for loop unless it's essential).

The syntax error is that the code is not inside a constructor or a method. The logic error is that x ends up being 0, so that the message printed is "0*8=40". Here's a version that corrects both problems:

```
public class Product {
   int x = 5, y = 8, p = 0;
   public Product () {
     int i=x;
     while (i > 0) {  i--; p = p + y; }
     System.out.println(x+" *" + y+ " ="+p);  }   }
```

```
public class Product {
  int x = 5, y = 8, p = 0;
  while (x > 0) {
    x--;
    p = p + y;
  }
  System.out.println(x+"*"+y+"="+p);
}
```

2. Our life is frittered away by detail... simplify, simplify (10 pts)

Rewrite the method to the right, using as few *symbols* (words, operators, etc.) as possible. Don't worry about minimizing curly brackets or semi-colons.

```
boolean ok(int[] array, int index) {
  return(index>=0 && index<=array.length);
}
```

```
boolean ok(int[] array, int index) {
  boolean legal;
  if (index < 0) {
     legal = false;
  }
  else if (index >= array.length) {
     legal = false;
  }
  else {
     legal = true;
  }
  return legal;
}
```

3. True/False (60 pts)

For each of the following questions about Java, circle T (true) or F (false).

T F Using a relational operator (e.g., ==, >=, !=) always produces a Boolean result. **(T)**

T F If one variable is assigned (using the = operator) to another, then comparing them with the == operator (immediately after the assignment) will always return true. **(T)**

T F Every method has a return type that is either a primitive data type or some kind of object. **(F)**

T F If a method has no parameters, then we don't need parentheses when we call it. **(F)**

T F The `toString` method is typically a method that converts between an object and a String value. **(T)**

T F Methods are said to be *overloaded* if they are in the same scope and have the same names but different signatures. **(T)**

T F The `static` modifier on a member variable associates it with the entire class itself rather than separately for each object. **(T)**

T F In Java, all methods strictly must belong to either individual objects or classes. **(T)**

T F The declaration statement `Object o` actually creates an object in memory. **(F)**

T F There is no limit to the number of data members an object can have. **(T)**

T F In Java, references are said to be passed by value, and not by reference. **(T)**

T F All objects have, in addition to any explicitly declared data members, a self-referencing pointer called `this`. **(T)**

T F Java uses single inheritance rather than multiple inheritance. **(T)**

T F Single inheritance means that any class can have at most one subclass that extends it. **(F)**

T F if Engine extends CarPart, then a variable of type Engine can be assigned a CarPart object. **(F)**

T F If Engine extends CarPart, then a variable of type CarPart can be assigned an Engine object. **(T)**

T F If `e` is an instance of `Engine`, then `e instanceof Engine` will be false. **(F)**

T F Java arrays are restricted to primitive types. **(F)**

T F If a class is defined to be final, it can not be extended. **(T)**

T F If a class is defined to be final, it can not be instantiated. **(F)**

T F Design patterns are used to document object-oriented programs. **(F)**

T F In the model-view-controller design pattern, the model implements the user interface. **(F)**

T F A Java class must define at least one constructor. **(F)**

T F A Java class can define at most one constructor. **(F)**

T F A method must declare a return type or null if no value is returned. **(F)**

T F Only one class can be defined in a file. **(F)**

T F Java's assert statement is useful for both debugging and documentation. **(T)**

T F Including the statement `assert false;` in a Java program will always generate a compiler error. **(F)**

T F Executing a program with more than one Java thread requires a computer with multiple processors. **(F)**

4. What Happens? (10 pts)

The following complete program prints four lines when executed. Show the four lines that are printed in the order in which they are printed.

```
public class ArrayTest {

  public static void main(String[] args) {
   int[] test = new int[2];
   test[0] = test[1] = 5;
   System.out.println(test[0] + "," + test[1]);
   fiddle(test, test[1]);
   System.out.println(test[0] + "," + test[1]);
  }

  static void fiddle(int[] test, int element) {
   test[0] = 10;
   test[1] = 11;
   element = 12;
   System.out.println(test[0] + "," + test[1]
       + "," + element);
   test = new int[2];
   test[0] = 20;
   test[1] = 21;
   System.out.println(test[0] + "," + test[1]);
  }
}
```

5,5
10,11,12
20,21
10,11

5. Short answers (20 pts)

Please keep all your answers short and to the point. Read the questions carefully and answer only the question that was asked--do not provide extra information that was not asked for.

(a) A class that is like an array that gets bigger when you add things to it is ____**Vector**_____.

(b) A method whose purpose is to create instances of a class is called a _____**Constructor**_____.

(c) If we construct an object of type Foo as defined by class Foo{int x=5; int y; Foo(){x=2;}}, the value of x in the new object will be ____**2**_____ and the value of y will be ____**0**_____.

(d) The name of the most general kind of object in Java is _____**Object**_____.

(e) Three examples of primitive data types in Java are _____**int**_____, _____**char**_____, and ____**boolean**_____. Or any other primitive datatype including byte, short, long, double or float.

(f) A class used to create an object from a primitive, so that the primitive can be used where an object is required, is called a(n) _____**wrapper**_____ class.

(g) When you *declare* an object (for example, `String s;`), space allocated for a(n) _____**pointer or reference**_____.

(h) CRC cards are used as an informal design aid. What does the acronym CRC stands for _____*Class-Responsibility-Collaborator*_____.

(i) A method that is *declared* but not *defined* is called a(n) _____**Abstract**_____ method.

(j) A class that is defined within another class is called a(n) _____**Inner**_____ class.

6. OO design (30 pts)

You've been hired to write a checkout system for the CSEE library. We want to keep track of who's checked out which books, and when the books are due. In particular, we want to make sure no patron has more than 20 books at a time, and we want to send a detailed overdue notice to any patron with late books. Tell what classes you would define, what data should be in each class, and briefly describe the most important methods in each class. Do *not* write any code, thought if you find it convenient you can partially describe a member variable using its type and a method by giving its signature.

The main classes might include the following:

- **Book: a class to represent an individual book with member variables Sting title, Person checkedOutTo, Date dueDate. Key methods will be**
  - **int numberOfBooks()**
  - **checkOut(Person p)**
  - **checkIn(),**
- **Patron: a person with member variables String Name, int libraryCardNumber, Vector<Book> booksOut**
- **Library: a class to represent a library with member variables String name, Vector<Book> books, Vector<Patron> patrons, Date now. Important methods are**
  - **checkout(Book b, Patron p)**
  - **checkIn(Book b, patron p)**
  - **sendOverdueNotices().**
- **Date: a class to represent a date with member variables int day, int month, int year. Methods might be**
  - **Date future(Date d, int numberOfDays)**
  - **before(Date d1, Date d2)**
  - **after(Date d1, Date d2).**

7. An exceptional question (10 pts)

Assume that the exception FooEx is properly defined. What will be the output of compiling and executing the main method of the following class?

The program prints the following text:

Start
In aMethod
aMethod's catch
aMethod's finally
After method
main's finally
End

```
class ExQ {

 public static void main(String[] args) {
  try {
   System.out.println("Start");
   ExQ x = new ExQ();
   x.aMethod();
   System.out.println("After method");
   }
  catch (FooEx error) {
   System.out.println("main's catch");
   }
  finally {
   System.out.println("main's finally");
   }
  System.out.println("End");
  };

 public void aMethod() throws FooEx {
  try {
   System.out.println("In aMethod");
   throw new FooEx();
   }
 catch (FooEx error ) {
  System.out.println("aMethod's catch");
  throw new FooEx();
  }
 finally {
  System.out.println("aMethod's finally");
  return;
  }
 };
}
```