Midterm Exam #1

This is a 50 minute closed book exam. You may use 1 page (8.5" x 11") (2 sides) of handwritten notes. Please keep track of the time— all exams will be collected at the end of the class period (2:15pm). Read all instructions carefully! We will give partial credit, so be sure to show your work and write down even a partial solution. Please read and sign the statement below and be sure to provide all of your info. Good luck!

I certify that the work on this exam is solely the result of my own efforts and that I have used no additional materials other than 1 page of handwritten notes, received no outside help or information, nor in any way contributed to another's work during this exam.

Signature: _____

Print Name: _____

Student ID Number: _____

Question	Total points possible	Points earned
1	20	
2	30	
3	30	
4	20	
Total	100	

1. Java expressions, operations, and loops. (20 points)

Write the output of the following lines of code. You can write the output of each part immediately below the corresponding code. Study the code carefully!

```
public class TestClass {
public static void main(String[] args) {

	// ***** Part a *****
	int i=0, j=5;
	i += j;
	j = i%2;
	System.out.println("j-- is: "+ (j--));
	System.out.println("j is now: " + j);




	// ***** Part b *****
	double num1=3.4, num2=6.1;
	boolean testVar = false;
	testVar = (num1 > num2);
	if((!testVar) && num2 > 0) {
		System.out.println("Condition 1 is true");
	}
	else if(num1 > 0) {
		System.out.println("Condition 2 is true");
	}
	else {
		System.out.println("Neither is true");
	}




	// ***** Part c *****
	i=0;
	int count=0;
	String sentenceString = "How|much|wood|would|a|woodchuck|chuck?";
	char[] sentence = sentenceString.toCharArray();

	while(i < sentence.length) {
		if(sentence[i] == '|') { count++; }
		if(sentence[i] == 'a') { break; }
		i++;
	}
	System.out.println("Stopped on "+sentence[i]);
	System.out.println("The number of words found is: "+count);
```

1. (continued)

```
// ***** Part d *****
int[] intArray = {5, 6, 0, 1, 2, 1, 4};
int sum=0;
for(i=1; i < intArray.length; i += 2) {
      sum += intArray[i];
}
System.out.println("sum is: "+sum);
```

```
// ***** Part e *****
int[][] twoDimArray = {{5, 6, 0},{1, 2, -1},{4,3,-2}};
sum=0;
int m=0,n=0;
for (m=0; m < twoDimArray.length; m++) {
      sum = 0;
      for (n=0; n < twoDimArray[m].length; n++) {
            sum += twoDimArray[m][n];
      }
      System.out.println("m:"+ m + ",n:" + n + ",sum:" +sum);
}
```

```
} // end main

} // end TestClass
```

2. Class definition. (30 points)

Write a class definition for a Matrix class, including any member variables and method implementations that are required. Your matrix class should have the functionality of storing a 2-dimensional matrix of integers, and here are the method specifications:

- One constructor (`Matrix`): input: two integers representing the number of rows and columns in the matrix; output: an object of type Matrix where the matrix elements are all initialized to zero

- `getElement`: input: two integers representing the row and column indices; output: the integer at the corresponding position in the matrix. Assume indices start with 0.

- `setElement`: input: three integers representing the row and column indices, and the value to be set at that position; output: integer that reports the status of the operation (0: element was set correctly, -1: error setting the element, e.g. indices out of bounds)

- `plus`: input: another object of type Matrix to be added to the current matrix; output: integer that reports the status of the operation (0: the matrices were added correctly, -1: error adding the matrices, e.g. their dimensions don't match). Behavior: this method should update the state of the current matrix by adding the corresponding elements of the input matrix. For anyone rusty on their matrix addition skills, here's an example:

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} + \begin{bmatrix} -1 & 0 \\ 1 & 2 \end{bmatrix} = \begin{bmatrix} 0 & 2 \\ 4 & 6 \end{bmatrix}$$

  NOTE: matrix addition is only defined when the two matrices have *exactly* the same dimensions.

- `getTranspose`: input: none; output: a new object of type Matrix that is the transposed version of the current matrix (NOTE: the state of the current matrix is not changed). The "transpose" of a matrix just means to "flip" the matrix on its side so the rows become the columns and the columns become the rows, e.g.

$$transpose\left(\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right) = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$$

  You don't have to implement them, but you can assume that these methods *already exist* in your class and you can use them if necessary:

  - `public int getNumRows()`: input: none; output: integer with the number of rows in the matrix

  - `public int getNumCols()`: input: none; output: integer with the number of columns in the matrix

Use any data members necessary to implement the functionality, and <u>all data members should be declared private</u>. Write your class definition on the following page, using the back of the page if necessary.

2. Class definition. (continued)

```
public class Matrix {




}
```

3. Inheritance and polymorphism. (30 points)

Study the two classes defined as shown below. Answer the questions that follow.

```
public class Vehicle {
        private String makeModel;
        private double speed, fuelLevel, mpg;

        public Vehicle(String makeModelIn) {
                makeModel = makeModelIn;
                speed=0;        // default speed is 0 miles/hr
                fuelLevel=15;  //default # of gallons
                mpg=20;         //default miles/gallon
        }

        public void updateFuelLevel(int hours) {
                fuelLevel -= (speed*hours)/mpg; }

        public String getStatus() {
                String info = new String("Vehicle:" + makeModel + ", Speed:"+speed+", Fuel left:
                "+fuelLevel);
                return info; }

        //assume the following standard accessor/mutator methods are also implemented
        public void setMakeModel(String makeModel) { ... }
        public String getMakeModel() { ... }
        public void setSpeed(double speed) { ... }
        public double getSpeed() { ... }
        public void setFuelLevel(double fuelLevel) { ... }
        public double getFuelLevel() { ... }
        public void setMPG (double mpg) { ... }
        public double getMPG() { ... }
}


public class Bus extends Vehicle {
        private String route;
        private int numPassengers;

        public Bus(String makeModelIn, String routeIn, int numPass) {

                // missing-- implement this method for Part (b)

        }

        public String getStatus() {
                String info = super.getStatus();
                info += "Route: "+route+", Num. Passengers: "+numPassengers;
                return info; }

        //assume the following standard accessor/mutator methods are also implemented
        public void setRoute(String route) { ... }
        public String getRoute() { ... }
        public void setNumPassengers(int numPass) { ... }
        public int getNumPassengers() { ... }
}
```

3. (continued)

Part (a). Which is the base and which is the derived class?

Part (b). Fill in the code for the Bus constructor method here. Assume that the default speed, fuel level and miles per gallon are 0, 100 and 5 respectively for instances of the Bus class. Your constructor should properly update the state of data members from both the derived and the base class.

```
public Bus(String makeModelIn, String routeIn, int numPass) {




}
```

Part (c). The `updateFuelLevel` method defined in the Vehicle class isn't precise enough for updating the fuel level of a Bus, because the amount of fuel spent by a bus depends on the number of passengers. Assume that the miles per gallon of fuel drops from the default of 5 mpg by 0.05 units per passenger. For example, for 10 passengers, the new mpg would be 5 – 10*(.05) =4.5 mpg. Redefine the `updateFuelLevel` method in the Bus class to fix this problem.

```
public void updateFuelLevel(int hours) {




}
```

Is this an example of overloading or overriding? _____

Part (d). Assume that you've finished Parts (b) and (c) correctly. Will the following code compile and run, and if so, what is the output? If not, why not?

```
public class TestClass {
public static void(String[] args) {
      Vehicle[] vehicleArr = new Vehicle[3];
      vehicleArr[0] = new Vehicle("Honda Accord");
      vehicleArr[1] = new Vehicle("Volkswagen Passat");
      vehicleArr[2] = new Bus("Volvo 7700 Hybrid","Mpls-metro 16",50);

      for(int i=0; i < vehicleArr.length; i++) {
            System.out.println(vehicleArr[i].getStatus());
      }
}
```

3. (continued)

Part (e). Assume that Parts (b) and (c) are implemented correctly. For each of the following sets of statements, indicate whether they will compile and run without error or not and if not, why not. Assume each of them is implemented in its own main method.

```
/**** Section 1 ****/
Vehicle myCar = new Vehicle("Honda Accord");
Bus myBus = (Bus)myCar;
myBus.setPassengers(4);
myBus.setRoute("Mpls-carpool");
```

```
/**** Section 2 ****/
Vehicle cityBus = new Bus("Volvo 7700 Hybrid","Mpls-metro 16",50);
cityBus.setSpeed(40);
cityBus.updateFuelLevel(2);
cityBus.setRoute("Mpls-metro 2");
```

```
/**** Section 3 ****/
Bus cityBus = new Vehicle("Volvo 7700 Hybrid");
myBus.setPassengers(15);
cityBus.setRoute("Mpls-metro 2");
```

4. Recursive vs. iterative algorithms. (20 points)

A triangular number is the number of equally spaced dots that it takes to fill an equilateral triangle where each side is of length N (e.g. see the series for N=1-4 below). This number shows up in many places—for example it is the number of possible pairings of N objects (e.g. number of handshakes required for everyone to shake hands in a room of N people) and it is the basis of a traditional East African game called tarumbeta.

N = 1	N = 2	N = 3	N = 4
T(1) = 1	T(2) = T(1) + 2 = 3	T(3) = T(2) + 3 = 6	T(4) = T(3) + 4 = 10

Your task is to implement a method that takes an integer as input and calculates the triangular number corresponding to that integer. Example: after the statement `int result = triangularNumber(4);` the variable `result` should contain the value 10.

Implement a Java version of the method `triangularNumber` using <u>either</u> recursion or iteration, but specify which type of solution you are using.

My algorithm is (recursive or iterative): _____

```
public int triangularNumber(int n) {




}
```

4. (continued)