CS 1322 Test 2 - Version 4 Wednesday June 25^{th}

Name: _____

GT Number: _____

TA: _____

Commenting your code on this test is optional, however it may help the grader understand your code better. Other style issues such as using good variables names and using abstraction are still required. Write your gt number in the spaces provided and color in the corresponding numbers and letters.

gt	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	d	e	f	g	h	0	1	2	3	4	5	6	7	8	9	-	-	-	-	-	-	-	-	-	-	-
	0	1	2	3	4	5	6	7	8	9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0	1	2	3	4	5	6	7	8	9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	0	1	2	3	4	5	6	7	8	9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	a	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

Problem	Points Possible	Points Received	Graded By
1 Matching	10		
2 Multidimensional Arrays	10		
3 Interfaces	10		
4 Abstract Classes	10		
5 Recursion	10		
6 Stacks and Queues	10		
7 Static vs. Instance	10		
8 Polymorphism	10		
9 Linked List	10		
10 Tracing	10		
TOTAL	100		

Please remember: Any academic misconduct (including, but not limited to, the list below) could result in a 0 (zero) on the exam, an F grade in the course, or other penalty:

- Communication with anyone other than a proctor for ANY reason in ANY language in any manner.
- Sharing of ANYTHING (e.g. pencils, paper, erasers).
- Writing on paper that is not given to you by a proctor.
- Failure to follow directions given by the proctor.
- Use of cell phones, beepers, handheld computers, calculators, during the exam.
- Using books or other reference material.
- Disruption of the exam setting.

By taking this exam, you signify that it is your work and that you have neither given nor received inappropriate help during the taking of this exam in compliance with the Academic Honor Code of Georgia Tech.

Signature: _____

(you must sign this for your exam to be graded!)

1. Matching[10 pts]

Choose the **best** word for each statement.

1. _____ The process in which a reference of a given class can refer to any instance of a subclass.
2. _____ A class where methods are declared, but not all are implemented.
3. _____ A programming technique whereby a method calls itself.
4. _____ FIFO data structure.
5. _____ Holds the location of an object.

A. Constructor
B. Polymorphism
C. Abstract
D. Compiler
E. Recursion
F. Reference
G. Inheritance
H. Interface
I. Private method
J. Dynamic Binding
K. Constructor Overloading
L. Stack
M. Queue

2. Multidimensional Arrays[10 pts]

Write public Point[][] createPoints(int x, int y)which creates and returns a new matrix of dimensions x by y, with Points instantiated with its row and column (ie Point(i,j)). The Point class has a constructor that takes in two intergers : public Point(int x, int y) which constructs and initializes a point at the specified (x, y) location in the coordinate space..

public Point[][] createPoints(int x, int y){

}

3. Interfaces[10 pts]

Declare a class **Energy** that has the following properties:

- Has a private double called force
- Has an accessor and modifier for the private double.
- Implements the Comparable interface. Items should be compared by the rforce field to determine if it is less than, equal, or greater than the Object it is being compared to as specified by the interface.

4. Abstract Classes[10 pts]

Given the following class:

```
public abstract class Toy{
        private String name;
        public void setName(String name){ this.name = name; }
        public String getName(){ return name; }
        public abstract double playWithMe();
}
```

Declare an instantiable (concrete) subclass of Machine. You may make the subclass do anything as long as it compiles.

5. Recursion[10 pts]

Write the method **public long geomScheme(long y)** which calculates the following geometrics series. If the input is less than 5, the method returns 2. Otherwise, the method returns the sum of **geomScheme(y%2)** plus **geomScheme(y/4)**.You must use recursion to solve this problem. If you use any iteration, or do not use recursion, you will not receive credit.

public long geomScheme(long y) {

}

6. Stacks and Queues[10 pts]

Assume you have the following **LinkedList** class. You may only assume that you have the following methods fully implemented and fields in the LinkedList class:

- private LLNode head;
- public void addToFront(Object o){...}
- public void addToBack(Object o){...}
- public Object removeFromFront(){...}
- public Object removeFromBack(){...}

Write the following methods for the **Queue** class. The methods should maintain the intergrity of the Queue data structure.

```
public class Queue extends LinkedList {
```

5 (a) **public void enqueue(Object o){**

}

5 (b) **public Object dequeue(){**

}

}//end class Queue

7. Static vs. Instance[10 pts]

Given the following class, write the output when the main method is run.

```
public class TryMeB{
  private static int one;

  private int two;

  public static void setOne(int x){
    one = x;
  }
  public static int getOne(){
    return one;
  }
  public void setTwo(int two){
    this.two=two;
  }
  public int getTwo(){
    return two;
  }
  public int specialSum(){
    return one*2+two/10;
  }

  public TryMeB(int x){
    one = x;
    two = x/10;
  }
  public static void main(String[] args){
    TryMeB a = new TryMeB(4);
    TryMeB b = new TryMeB(6);
    TryMeB c = new TryMeB(20);
    System.out.println("a.getOne() is " + a.getOne());
    System.out.println("c.getOne() is " + c.getOne());
    System.out.println("c.specialSum() is " + c.specialSum());
    a.setOne(6);
    System.out.println("b.getOne() is " + b.getOne());
    b.setTwo(5);
    System.out.println("a.getTwo() is " + a.getTwo());
  }
}
```

8. Polymorphism[10 pts]

Given the following class hierarchy with listed methods:

- **public abstract class Emotion**
  has: **public void express()**
- **public interface Crying**
  has: **public void tears()**
- **public class Joy extends Emotion implements Crying**
  has: **public void smile()**
  has: **public void tears()**
- **public class PureJoy extends Joy**
  has: **public void exult()**
- **public class Anger extends Emotion**
  has: **public void yell()**

Determine which of the following statements will compile and run without any errors or exceptions. If there is no error, write **OK**; if there will be an error at compile time, write **Compile Error**; if there will be a problem during during execution, write **Runtime Error**. If you know there is an error but cannot determine whether it is at Run-time or compile-time, you may write error and explain why in 15 words or less. If you write both, you will not receive credit for that part.

2 (a)
```
Crying c = new Crying();
```

2 (b)
```
Emotion my = new Anger();
```

2 (c)
```
Emotion great = new PureJoy();
great.smile();
```

2 (d)
```
Emotion well = new Anger();
((Joy)well).smile();
```

2 (e)
```
PureJoy lastone = new Joy();
```

9. **Linked List[10 pts]**

Given the following **class LLNode**:

```
public class LLNode{
        private Comparable data;
        private LLNode next=null;
        public void setData(Comparable data){ this.data = data; }
        public void setNext(LLNode next){ this.next = next; }
        public Comparable getData(){ return data; }
        public LLNode getNext(){ return next; }
        public LLNode(Comparable data){ this.data = data; }
}
```

Write the **removeAllOccurrences(Comparable c)** method for the LinkedList class, which will remove all occurrences of c from the Linked List. You may not assume anything else about the LinkedList class or the LLNode class. You may write any helper methods you see fit to solve the problem.

```
public class LinkedList{
        private LLNode head;

        public void removeAllOccurrences(Comparable c){




        }
}
```

10. **Tracing[10 pts]**

Given the following classes, write the output when the main method is run.

```
public class ParentClass{
  public String random(){ return "Cheese"; }
  public String getName(){ return "Me"; }
  public ParentClass(){
    this(null);
    System.out.println("In ParentClass()");
  }
  public ParentClass(String x){
    System.out.println("In ParentClass(String)");
  }
}//end ParentClass

public class ChildClass extends ParentClass{
  public String random(){ return "Mechanic"; }
  public ChildClass(){
    System.out.println("In ChildChildClass()");
  }
  public ChildClass(String x){
    this();
    System.out.println("In ChildChildClass{String)");
  }
  public static void main(String[] args){
    ParentClass a = new ParentClass();
    ChildClass b = new ChildClass();
    System.out.println(a.random());
    System.out.println(a.getName());
    System.out.println(b.getName());
    a = b;
    System.out.println( a.random() );
    System.out.println( ((ParentClass)a).random() );
  }
}
```