CMP 326 Final Fall 2015 Name: _____

There is a blank page at the end of the exam if you need more room to answer a question.

1) (10 pts) Fill in the blanks to specify the missing keywords or definitions.

```
public class SomeClass _____ OtherClass _____ SomeInterface {…}
```

If a class, variable, or method is _____, then it can be referenced only in the class it is defined in.

A(n) _____ variable is declared inside of a method.

A(n) _____ is passed into a method.

A(n) _____ class cannot be instantiated.

If one class contains two methods with the same name, but different signatures, then these methods are _____ .

To test if `instance1` is an instance of the class `SomeClass`, we write
`instance1 _____ SomeClass.`

A(n) _____ variable in a class is shared by all instances of that class.

If an object is passed by _____ to a method it can be modified and then this change will be _____ by the caller method.

2) (25 pts total; 5 pts. per part) Write the output of each piece of code. If the code gives an error, write any output produced before the error, and then write "ERROR". If the code results in an infinite loop, write the output from two iterations of the loop, and then write "INFINITE".

	Code	Output

a)	Given the method `mys1`:
	
	`public static int mys1(int[] a, int i){`
	`  if ((i< 0) || (i > a.length)) {`
	`  return 0;`
	`  }`
	`  return a[i] + mys1(a,i-1);`
	`}`
	
	What will be the output from the code:
	`int[] arr = {1,2,3,4,5};`
	`System.out.println(mys1(arr,1));`
	`System.out.println(mys1(arr,2));`
	`System.out.println(mys1(arr,3));`	

b)	What will be the output from the call `mys2(7)`?
	
	`public static void mys2(int num) {`
	`  System.out.println(num);`
	`  if (num == 0 || num ==1 ) {`
	`  return;`
	`  }`
	`  mys2(num-2);`
	`}`	

c)	`for( int i = 2; i < 5; i++ ) {`
	`  for( int j = i; j > 0; j-- ) {`
	`  System.out.print("*");`
	`  }`
	`  System.out.println();`
	`}`	

d) What will be the output from calling `System.out.println(f(7))`?

```
public static int f(int n){
  int[] x = {1, 1};

  for(int i = 1; i < n; i += 2) {
    System.out.println(x[0] + " " + x[1]);
    for(int j = 0; j < x.length; j++) {
      x[j] += x[1-j];
    }
  }
  return x[n%2];
}
```

e) What will be the output from calling `main()`?

```
public class Cla {
  private String a;
  private static boolean b;

  public Cla(String inA, boolean inB){
      a = inA;
      b = inB;
  }

  public String toString() {
      return a + " " + b;
  }

  public static void main(String[] args) {
      Cla m1 = new Cla("Lehman",false);
      System.out.println(m1);
      Cla m2 = new Cla("College",true);
      System.out.println(m1);
      System.out.println(m2);
      Cla m3 = m1;
      m3.a = "Bronx";
      System.out.println(m1);
      System.out.println(m2);
      System.out.println(m3);
  }
}
```

3) (40 pts) Use the classes below to complete this section.

a) (25 pts) Do the following for the class `BankAccount`:

I. (5 pts) Write a constructor that initializes the three instance variables. `name` and `balance` should be initialized based on parameters. `accountID` should be unique for each `BankAccount` object. The first `BankAccount` should have an `accountID` of 1, the second 2, the third 3, and so on. Class variable `count` should keep track of how many `BankAccount` objects are instantiated.

II. (5 pts) Write a method called `deposit` that increases the `balance` of the `BankAccount` by the given parameter.

III. (5 pts) Write a method called `withdraw` that reduces the `balance` of the `BankAccount` by the given parameter. If the `balance` would become negative, do not reduce the `balance` and instead throw an exception.

IV. (5 pts) Define the `compareTo` method so that it compares the `balance` of the `BankAccount` calling it with the `balance` of the `otherBankAccount`. The method should return a negative number if the `balance` of the calling `BankAccount` is less than that of `otherBankAccount`, `0` if the two are equal, and a positive number otherwise.

V. (5 pts) Override the default `toString` method so it returns a nicely-formatted `String` containing the names and values of all the `BankAccount's` variables.

Example Output:

```
Account: 8 Name: John Balance: $400
```

```
public class BankAccount implements Comparable<BankAccount> {
   private String name;
   private int accountID;
   private int balance;     // balance of account in dollars

   protected static int count = 0;

    // answer for part (a)I:
   public BankAccount(                                              )
    {









   }
```

```
// answer for part (a)II:
```

```
// answer for part (a)III:
```

```
// answer for part (a)IV:
```

```
   public int compareTo(BankAccount otherBankAccount){









}

      // answer for part (a)V:
   public String toString(){









   }
}
```

b) (15 pts) Do the following for the class `CheckingAccount`, which is a subclass of `BankAccount`:
I. (5 pts) Define a constructor so it does everything the constructor from the parent class does, and also initializes `numchecks` based on a parameter.
II. (5 pts) Create a method called `setNewChecks` that increases `numchecks` by an amount specified in its parameter and throws an `Exception` if the input parameter is i) less than 20
or ii) greater than 100 and not a multiple of 20.
III. Define method `checkCashed` that takes in one `int` parameter representing the amount of the check. This method should withdraw that amount of money from the account and decrement the number of checks the customer has by one. If the `balance` would become negative by withdrawing the amount, the method should not reduce the `balance` and instead throw an exception.

```
public class CheckingAccount extends BankAccount
{
   private int numchecks;   // the number of checks the customer has

  // answer for part (b)I:
   public CheckingAccount(                                    )
   {









   }

  // answer for part (b)II:
```

```
    // answer for part (b)III:

}
```

4) (15 points) Write a static method `bringToFront` that takes in a `String` array `inArr` and an `int index`, and returns a `String` array. The returned array should be the same same as `inArr` except:

- the `String` at position `index` has been moved to position 0
- the `String`s in positions `0` to `index-1` have been moved to one position higher in the array
- the `String`s in positions `index+1` and higher should not be changed

If `index` is not a valid position in `inArr` (i.e. it is too big or too small), return `inArr`.

For example, if `inArr` is

"a"	"b"	"c"	"d"	"e"	"f"	"g"

then the call `bringToFront(inArr,4)` should return the array

"e"	"a"	"b"	"c"	"d"	"f"	"g"

5) (15 pts) Given the `Rectangle` and `Circle` classes below:

a) Define an abstract class called `Shape` from which the `Rectangle` and `Circle` class can be derived. Extract as much code as possible into the `Shape` class.

b) Modify the code below so that `Rectangle` and `Circle` are derived from `Shape`. You should mark your changes directly on the exam.

```
public class Rectangle {
   int x;    // x-coordinate of center of rectangle
   int y;    // y-coordinate of center of rectangle
   int w;    // width of rectangle
   int h;    // height of rectangle

   public Rectangle(int x, int y, int w, int h){
      this.x = x;
      this.y = y;
      this.w = w;
      this.h = h;
   }

   public double area() {
      return (double)(w * h);
   }

   public void moveBy(int dx, int dy) {
      x += dx;
      y += dy;
   }
}


public class Circle {
   int x;    // x-coordinate of center of rectangle
   int y;    // y-coordinate of center of rectangle
   int r;    // radius of circle

   public Circle(int x, int y, int r) {
      this.x = x;
      this.y = y;
      this.r = r;
   }

   public double area() {
      return Math.PI * r * r;
   }

   public void moveBy(int dx, int dy){
      x += dx;
      y += dy;
   }
}
```

Define your class `Shape` here:

6) (15 points) Write a **RECURSIVE** method `sumEven` that has one parameter, an `int n`, and returns the sum of the first `n` even numbers. Assume `n > 0` and the first even number is 2.

For example, `sumEven(1)` should return 2,
`sumEven(2)` should return 6 (= 2 + 4)
`sumEven(3)` should return 12 (= 2 + 4 + 6)

7) (10 points) Write a method called `firstWord` that takes in two `String`s, `infileName` and `outfileName`. The method should open and read from the file `infileName`. For each line in `infileName`, this method should print the first word of that line to the file `outfileName`. All exceptions should be handled within this method by printing an error message and exiting the program.

For example, if `infileName` is:

```
To be, or not to be, that is the question—
Whether 'tis Nobler in the mind to suffer
The Slings and Arrows of outrageous Fortune,
Or to take Arms against a Sea of troubles,
```

Then `firstWord` should print the following to `outfileName`:

```
To
Whether
The
Or
```