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.

```java
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.

toString() is a ___________________ in the class Object.

A(n) ______________________ class can not 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.

The value in a(n) ______________________ variable in a class can be accessed by any instance of that class.
2) (30 pts) 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”.

<table>
<thead>
<tr>
<th>Code</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) (5 pts)</td>
<td>What will be the output from the call to mystery(7);</td>
</tr>
<tr>
<td></td>
<td>public static void mystery(int num){</td>
</tr>
<tr>
<td></td>
<td>if(num &lt;= 0) {</td>
</tr>
<tr>
<td></td>
<td>System.out.println(num);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>else if(num % 2 == 1){</td>
</tr>
<tr>
<td></td>
<td>mystery(num - 1);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>else {</td>
</tr>
<tr>
<td></td>
<td>//before recursive call</td>
</tr>
<tr>
<td></td>
<td>System.out.println(num);</td>
</tr>
<tr>
<td></td>
<td>mystery(num - 2);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>b) (5 pts)</td>
<td>What will be the output from the call mystery(6)?</td>
</tr>
<tr>
<td></td>
<td>public static void mystery(int num){</td>
</tr>
<tr>
<td></td>
<td>if(num % 2 == 0){</td>
</tr>
<tr>
<td></td>
<td>mystery(num - 1);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>else if(num == 1) {</td>
</tr>
<tr>
<td></td>
<td>System.out.println(num);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>else {</td>
</tr>
<tr>
<td></td>
<td>mystery(num - 2);</td>
</tr>
<tr>
<td></td>
<td>//after recursive call</td>
</tr>
<tr>
<td></td>
<td>System.out.println(num);</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td>c) (5 pts)</td>
<td>String a = &quot;abcde&quot;;</td>
</tr>
<tr>
<td></td>
<td>for( int i = 0; i &lt; 3; i++ ) {</td>
</tr>
<tr>
<td></td>
<td>for( int j = 3; j &gt; i; j-- ) {</td>
</tr>
<tr>
<td></td>
<td>System.out.println(a.substring(i,j));</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
<tr>
<td></td>
<td>}</td>
</tr>
</tbody>
</table>
d) String month = "May";
   int hourOfDay = 9;
   if (month.equals("May")) {
      while (hourOfDay < 12) {
         System.out.println("Chilly at " + hourOfDay);
         hourOfDay++;
         System.out.println("Warmer at " + hourOfDay);
      }
      month = "January";
   }
   if (month.equals("January")) {
      System.out.println("It is " + hourOfDay + ", and I am freezing.");
   }

e) public static void mystery(int[] a) {
   if (a.length > 2) {
      System.out.println(a[0]);
      a[0] = 100;
      a[1] = 101;
   }
}
   int[] b = {33, 44, 55};
   System.out.println(b[0]);
   int[] c = b;
   System.out.println(c[2]);
   mystery(b);
   System.out.println(b[1]);

f) What will be the output from calling main()?  
public class Mystery {
   public int a;
   public static String b;

   public Mystery(int inA, String inB) {
      a = inA;
      b = inB;
   }

   public static void main(String[] args) {
      Mystery m1 = new Mystery(4,"hi");
      Mystery m2 = new Mystery(5,"bye");
      System.out.println(m1.a);
      System.out.println(m1.b);
      System.out.println(m2.a);
      System.out.println(m2.b);
   }
}
3) (30 pts) Use the classes below to complete this section.

a) (15 pts) Do the following for the class Building:

I. (5 pts)
Write get and set methods for the numOfEntrances variable. Be sure only positive values are set; negative or zero values should throw an Exception. Assume the other private variables have their getter/setter methods already defined.

II. (5 pts)
Define the compareTo method so that it compares the numOfFloors of the Building calling it with the numOfFloors of the otherBuilding. The method should return a negative number if the numOfFloors of the calling Building is less than that of otherBuilding, 0 if the two are equal, and a positive number otherwise.

III. (5 pts)
Override the default toString method so it returns a nicely-formatted String containing the names and values of all the Building’s variables.

Example Output:
This building has 4 entrances and 5 floors. There is no parking.
This building has 6 entrances and 4 floors. There is parking.

```java
public class Building implements Comparable<Building> {

    private int numOfEntrances;
    private int numOfFloors;
    private boolean hasParking;

    public Building(){
        numOfEntrances=1;
        numOfFloors=1;
        hasParking=false;
    }

    public void setHasParking(boolean canPark) {...}
    public void setNumOfFloors(int numOfFloors) {...}

    // write your get and set methods here
```
public int compareTo(Building otherBuilding) {

}

public String toString() {

}
b) (15 pts) Do the following for the subclass Museum:

I. (5 pts) Define the default constructor so it does everything the default constructor from the parent class does, and also sets hasTempExhibits to true and name to "a museum".

II. (5 pts) Create a setter for numOfEntrances that throws an Exception if the input parameter is less than 2 or greater than 15.

III. (5 pts) Write an equals method for Museum. It should return true if the hasTempExhibits and name of the two Museum instances are the same, and false otherwise.

```java
public class Museum extends Building {
    private String name;
    private boolean hasTempExhibits;

    public Museum() {
    }

    public void setNumOfEntrances(int entrances) throws Exception {
    }
}
```
public boolean equals(Object obj) {

}
4) (15 points) Write a static method that takes in an int array inArr, and returns an int array in which every second entry of inArr has been replaced by the entry directly before it. That is, the first entry should replace the second entry, the third entry should replace the fourth entry, etc.

For example, if inArr is

| 8 | -3 | 4 | 0 | 11 | 29 | 5 |

then this method should return the array

| 8 | 8 | 4 | 4 | 11 | 11 | 5 |
5) (15 points) Write a **RECURSIVE** method countChars that takes in two parameters: a *String* and a *char*. This method should return the number of times the *char* appears in the *String*.

For example, if the *char* is ‘e’ and the *String* is ‘elephant’, then the method should return 2. If the *char* is ‘b’ and the *String* is ‘zebra’, then the method should return 1.
6) (15 points) Write a method called `numberLines` that takes in two Strings, `infileName` and `outfileName`. The method should open and read from the file `infileName`. For each line in `infileName`, this method should write the line number, followed by that line, to `outfileName`. The line numbers should start at 1.

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 `outfileName` should be:

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