CMP 326 Final Spring 2014 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 {…}
```

A(n)_____ has at least 1 abstract method along with defined methods, and variables.

A(n)_____ has all its methods defined (no abstract methods).

A(n)_____ has only abstract methods and constant variables declared.

The keyword _____ references methods and variables in the base/parent class.

The keyword _____ makes a variable shared among all objects of its class.

The keyword _____ references methods and variables in the derived/child class.

A method is _____ when there is another method in a subclass with the same signature, but the new method has a different body definition.

A method is _____ when there are multiple methods within the same class that have the same name, but different signatures.

2) (25 pts) Write the output of each piece of code. If the code gives an error, write "ERROR".

	Code	Output

a)
 	
(5 points)
	What will be returned by the following call to `mystery`? Show all work.
	
	`mystery (3);`
	
	`public static int mystery (int n){`
	`  if (n <= 1) {`
	`  return 1;`
	`  } else {`
	`  return mystery (n - 1) + n;`
	`  }`
	`}`	

b)
 	
(5 points)
	`String a = "apricot";`
	`String e = "watermelon";`
	
	`System.out.println(`
	`  a.substring(1,4) +`
	`  e.substring(6,9));`	

c)
 	
(5 points)
	`int i = 56;`
	`while( i > 1 ) {`
	`  i /= 3;`
	`  System.out.println(i * 2);`
	`  i += 2;`
	`}`	

d)
 	
(5 points)
	`String[] a = {"orange", "banana",`
	`  "apple", "kiwi"};`
	`String[] b = a;`
	
	`System.out.println(a[3]);`
	`a[3] = "lemon";`
	`System.out.println(a[2]);`
	`System.out.println(a[3]);`
	`System.out.println(a[0]);`
	`System.out.println(b[1]);`
	`System.out.println(b[3]);`	

e)
 	
(5 points)
	`boolean done = true;`
	`int n = 5;`
	`int k = 2;`
	`do {`
	`  System.out.println("n = "`
	`  + n + "; k = " + k);`
	
	`  n = n - 2 * k;`
	`  if (n < 0) {`
	`  done = true;`
	`  } else {`
	`  done = false;`
	`  k = 5;`
	`  }`
	`} while( !done );`	

3) (42 pts) Use the classes below to complete this section.

a) (21 pts)
Do the following for `MotorVehicle`:

I. (7 pts) Write get and set methods for the `numOfTires` variable. Be sure only positive values are set; negative values should be ignored.
II. (7 pts) Define the `compareTo` method so that it compares the `MotorVehicle` calling it with `otherObject` using `engineLiterSize`. The method should return `-1` if the `engineLiterSize` of the calling `MotorVehicle` is less than that of `otherObject`, `0` if the two are equal, and `1` otherwise. Comment your code.
III. (7 pts) Override the default `toString` method so it returns a nicely-formatted `String` containing the names and values of all the object's variables.

```
public class MotorVehicle implements Comparable<Motor {
    private int numOfTires=0;
    public int numOfDoors=0;
    public double engineLiterSize=0.0;

    public MotorVehicle(){
          numOfTires=2;
    }

    // write your get and set methods here









    public int compareTo(Object otherObject){









    }
```

```
    public String toString(){









    }
}
```

b) (21 pts) Do the following for `RegularCar`:

I. (7 pts) Define the default constructor so that it does everything the default constructor from the parent class does, and also sets the `numOfDoors` to 2, and `numOfTires` to 4.

II. (7 pts) Create a setter for `sizeOfGasTank` that throws an `Exception` if the input parameter is less than 5.

III. (7 pts) Write an `equals` method for `RegularCar`. It should return `true` if the values of all the instance variables in the objects being compared are equal, and `false` otherwise.

```
public class RegularCar extends MotorVehicle {
    private double sizeOfGasTank;

    public RegularCar() {









    }
    public void setSizeOfGasTank(double newSize) throws Exception {









    }
    public boolean equals(RegularCar otherCar) {









   }
}
```

4) (10 points) Write a **RECURSIVE** `static` method `isPalindrome` to check if a `String` looks the same if you read it forward or backwards (i.e. if the `String` is a *palindrome*). This method should take in one parameter, a `String s`, and return a `Boolean` value of `true` if `s` reads the same forward and backward, and returns `false` if not.

For example, if the input `String` is `"kayak"`, then `isPalindrome` should return `true`;
if the input `String` is `"a"`, then `isPalindrome` should return `true;`
and if the input `String` is `"canoe"`, then `isPalindrome` should return `false.`

5) (20 points) Write a method called `readInAs` that takes in three `String`s, `infileName`, `outfileName1`, and `outfileName2`. The method should open and read from the file `infileName`. For each line in the input file, this method should write the line to file `outfileName1` if the first character is `'A'`. If not, the line should be written to `outfileName2`.

For example, if the input file is

```
There is A cat in a hat.
A bird is in a hand.
a fish in a dish.
Some dogs on A log.
```

Output file 1 would be:

```
A bird is in a hand.
```

Output file 2 would be:

```
There is A cat in a hat.
a fish in a dish.
Some dogs on A log.
```

6) (10 points) Consider the Applet below that contains 2 `TextFields`, 1 `Label`, and 2 `Buttons`:

Define the body of the `actionPerformed` method as follows:

- if the button labeled `MULTIPLY` is clicked, then `JLabel lblOutput` is set to be the product of the numbers in `JTextFields` `tf1` and `tf2`,
- if the button `DIVIDE` is clicked, then `JLabel lblOutput` is set to be `JTextField tf1` divided by `TextField tf2`. (Division by zero is not allowed, see below.)

Handle invalid input as follows:

- If either of the `TextFields` is empty when a calculation button is clicked, or if there is a zero in `tf2` when the DIVIDE button is clicked, DO NOT perform a calculation. Instead, set the contents of `lblOutput` to read “Invalid Input”.

```
import java.awt.event.ActionListener;
import javax.swing.JApplet;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JTextField;

public class MathApplet extends JApplet implements ActionListener{
     JTextField tf1 = new JTextField("num1");
     JTextField tf2 = new JTextField("num2");
     JButton btnMult = new JButton("MULTIPLY");
     JButton btnProd = new JButton("DIVIDE");
     JLabel lblOutput = new JLabel("output goes here");
/**
* other methods in the class, including the
* init() method, are not shown here
*/
    @Override
    public void actionPerformed(ActionEvent e){
    // your code will go here









    }
}
```

This page is intentionally left blank.