Java Practice Final Exam Name -

1. Write method `countEvens` which returns the number of even integers found in its array parameter. For example, assume that array `nums` is as shown below.

subscript	0	1	2	3	4	5	6
value	6	3	4	5	2	7	3

then `System.out.println(countEvens(nums));` should display 3 since 6, 4, and 2 are even.

```
public static int countEvens(int[] nums)
{
     // precondition: nums will contain integers greater than zero
     // postcondition: returns the number of even integers found in nums
```

2. Write method `countNums` which returns a `String` that is formatted as the example below. Assume that the array parameter `nums` stores the following values:

subscript	0	1	2	3	4	5	6
value	2	3	4	5	4	4	3

Then the statement `System.out.println(countNums(nums));` should produce the following output:

```
1: 0
2: 1
3: 2
4: 3
5: 1
```

since there are no 1's in the array `nums`, there is one 2, there are two 3's, there are three 4's, & there is one 5.

```
public static String countNums(int[] nums)
{
     //    precondition: nums will only contain integers 1 through 5.
     //    postcondition: prints the integer values 1 through 5 on
     //                   separate lines with each followed by a colon,
     //                   a space and the number of times that integer
     //                   occurs as a value in nums
```

3. This question concerns the `Number` class, partially defined below.

```
public class Number
{
     private double myValue;

     public int getWhole()
     {
          // postcondition: returns the whole-number part
          //                of the value in the myValue property
          <implementation code to be written in an exercise>
     }

     public int getDec()
     {
          // postcondition: returns the fractional part of the value
          //                in the myValue property as an int in the
          //                range 0 to 99
          <implementation code to be written in an exercise>
     }

     <other typical methods as necessary>
}
```

Write the `getWhole` method of the `Number` class. Method `getWhole` returns the whole-number part of the value in the `myValue` property as an `int`. For example:

myValue	Value returned by `getWhole()`
123.40	123
0.33	0
10.00	10
0.00	0

Complete method `getWhole` below.

```
public int getWhole()
{
     // postcondition:   returns the whole-number part of the value
     //                  in the myValue property
```

4. Write the `getDec` method of the `Number` class. Method `getDec` return the fractional part of the value in the `myValue` property as an `int` in the range 0 to 99. You are guaranteed as a precondition that the decimal part of `myValue` will be a number between 00 and 99. That is you are guaranteed that there will be exactly two digits to the right of the decimal point in `myValue`. For example:

`myValue`	Value returned by `getDec()`
`123.40`	`40`
`12.04`	`4`
`0.33`	`33`
`45.00`	`0`

Complete method `getDec` below. In writing method `getDec`, you may include calls to method `getWhole`. Assume that method `getWhole` works as specified, regardless of what you wrote for Exercise #3.

```
public int getDec()
{
     // postcondition: returns the fractional part of the value in the
     //                myValue property as an int in the range 0 to 99
```

The following questions involve the following two incomplete class definitions, which define classes to be used for storing information about the students in a high school Java class.

```
public class Student
{
     private String myFirstName;
     private String myLastName;

     public Student(String firstName, String lastName)
     {
          // implementation code to be written in an exercise
     }

     // other methods and properties not shown
}

public class JavaClass
{
     private ArrayList<Student> myStudents;
     private String myClassName;

     public JavaClass()
     { /* implementation not shown but you can assume it works */ }

     public boolean findStudent(String firstName)
     {

     }

     public int countStudents(String letter)
     {

     }

     // other methods and properties not shown
}
```

5. Write an "other constructor" for the `Student` class that takes two `String` parameters `firstName` and `lastName`.

```
public Student(String firstName, String lastName)
{
```

6. Write an accessor method for the `myFirstName` property of the `Student` class.

7. Write a modifier method for the `myFirstName` property of the `Student` class.

8. Write the `findStudent` method for the `JavaClass` class that returns a true or a false depending on whether or not one or more students with a first name of `firstName` is in the class or not.

```
public boolean findStudent(String firstName)
{
```

9. Write the `countStudents` method for the `JavaClass` class that returns the number of students whose first name begins with the letter stored in the parameter `letter`. You can assume as a precondition that letter stores exactly one uppercase alphabetical character and that all `Student` first names begin with uppercase letters.

```
public int countStudents(String letter)
{
```