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.

<table>
<thead>
<tr>
<th>subscript</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>6</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>3</td>
</tr>
</tbody>
</table>

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

```java
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:

<table>
<thead>
<tr>
<th>subscript</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

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.

```java
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.

```java
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:

<table>
<thead>
<tr>
<th><code>myValue</code></th>
<th>Value returned by <code>getWhole()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>123.40</td>
<td>123</td>
</tr>
<tr>
<td>0.33</td>
<td>0</td>
</tr>
<tr>
<td>10.00</td>
<td>10</td>
</tr>
<tr>
<td>0.00</td>
<td>0</td>
</tr>
</tbody>
</table>

Complete method `getWhole` below.

```java
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:

<table>
<thead>
<tr>
<th>myValue</th>
<th>Value returned by <code>getDec()</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>123.40</td>
<td>40</td>
</tr>
<tr>
<td>12.04</td>
<td>4</td>
</tr>
<tr>
<td>0.33</td>
<td>33</td>
</tr>
<tr>
<td>45.00</td>
<td>0</td>
</tr>
</tbody>
</table>

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.

```java
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.

```java
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.

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

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

```java
```
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.

   ```java
   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.

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