

1. Write method `numInArray`, as started below. `numInArray` should return the number of times an even `int` parameter occurs in array `A`. For example, assume that array `A` is as shown below.

[0]	[1]	[2]	[3]	[4]	[5]	[6]
6	3	4	5	2	7	3

then `numInArray(A)` should return 3 since 6, 4, and 2 are even.

Complete method `numInArray` below.

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

2. Write method `printAllNums`, as started below. The `ArrayList A` contains `Integer` object in `ArrayList A`, `printAllNums` should write (using `System.out.println`) the string `s`, followed by a colon and a space.

	[0]	[1]	[2]	[3]	[4]	[5]	[6]
A:	2	3	4	5	4	4	3

The call `printAllNums(A)` should produce the following output:

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

Complete method `printAllNums` below.

```
public static void printAllNums(ArrayList A)
{
    // precondition: A will only contain Integer objects that
    //                each contain values 1 through 5.
    // postcondition: prints the integer values 1 through 5 each
    //                each followed by a colon and a space and the
    //                number of times that int occurs as a value in
    //                an Integer object stored A
}
```

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

```
public class Number
{
    /** fields */
    private double val;

    /** public methods */
    public String toString()
    { . . . }

    /** private methods */
    private int getWhole()
    {
        // postcondition: returns the whole-number part
        //                  of the value in the val field
    }

    private int getDec()
    {
        // postcondition: returns the fractional part of the value
        //                  in the val field as an int in the
        //                  range 0 to 99
    }
}
```

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

<u>val</u>	<u>Value returned by <code>getWhole()</code></u>
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 val field
}
```

4. Write the `getDec` method of the `Number` class. Method `getDec` return the fractional part of the value in the `val` field as an `int` in the range 0 to 99. You are guaranteed as a precondition that the decimal part of `val` 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 `val`. For example:

<u>val</u>	<u>Value returned by getDec ()</u>
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` as specified in Exercise #3. 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
    //                val field as an int in the range 0 to 99
```

5. This question involves the following two incomplete class definitions, which defines classes to be used for storing information about the students in a computer programming class.

```
public class JavaClass
{
    private StudentInfo[] students;
    private String highestAverage;

    public JavaClass()
    {
        . . .
    }
}

public class StudentInfo
{
    private String name;
    private ArrayList grades;
    private double averageGrade;

    public StudentInfo(String theName, ArrayList theGrades)
    {
        . . .
    }

    public String getName()
    {
        return name;
    }

    public double getAverageGrade()
    {
        return averageGrade;
    }
}
```

Write the constructor for the `StudentInfo` class. The constructor should initialize the `name` and `grades` field using the given values, and then it should compute the average grade and use that value to initialize the `averageGrade` field. (If the number of grades is zero, the `averageGrade` field should be set to zero.)

Complete the constructor below.

```
public StudentInfo(String theName, ArrayList theGrades)
{
```