**Overloaded Methods**

1. Write a class called `Averages` that contains three overloaded static methods named `calculateAverage()`.

   The first `calculateAverage()` method should take two `double` values as parameters and return a `double`, which is the average of the two values. For example:

   ```
   TextIO.putf("%.2f %n", Averages.calculateAverage(4,5));
   ```

   should output 4.50. (Remember that `int` values can be automatically converted to `double`, though obviously we could have written 4.0, 5.0 rather than 4, 5).

   The second `calculateAverage()` method should take three `double` values as parameters and return a `double`, which is the average of the three values. For example:

   ```
   TextIO.putf("%.2f %n", Averages.calculateAverage(4,5,6));
   ```

   should output 5.00.

   The third `calculateAverage()` method should take four `double` values as parameters and return a `double`, which is the average of the four values. For example:

   ```
   TextIO.putf("%.2f %n", Averages.calculateAverage(4,5,6,7));
   ```

   should output 5.50.

   Write a `main()` method in a separate class called `TestAverages` that presents the user with a menu along these lines:

   ```
   1. Calculate the average of two numbers
   2. Calculate the average of three numbers
   3. Calculate the average of four numbers
   4. Quit

   Enter your choice (1-4):
   ```

   If the user chooses 1, the program should prompt the user to enter the two values and then display their average by calling the relevant overloaded method, if they choose 2 the program should prompt the user to enter the three values and then display their average by calling the relevant overloaded method and so on.

   Note: When we come to return to the subject of arrays you will see that it is possible to declare a method along these lines:

   ```
   public static double calculateAverage(double... x)
   ```

   This method can be called with two **`double`** arguments, or with three, or four, or five, etc., and the arguments are copied into an array for the method to process. In this case, using this facility would allow us to avoid overloading the methods as above, and to just have the one method.

2. Write a class called `Division` with two overloaded static methods named `divide()`, one which performs integer division on two `int` values and one which performs real division on two `double` values.

   The first `divide()` method should take two formal `int` parameters named `dividend` and `divisor` and return an instance of this class, which you should add to your project:

```
public class DivisionResult {
      public int quotient;
      public int remainder;
}
```

   For example, the code:

```
DivisionResult answer = Division.divide(7, 2);
TextIO.put("7 divided by 2 is ");
TextIO.putln(answer.quotient + " remainder " + answer.remainder);
```

   would output:

```
7 divided by 2 is 3 remainder 1
```

   The second `divide()` method should take two formal `double` parameters named `dividend` and `divisor` and return a `double`.

   For example, the code:

```
double realAnswer = Division.divide(7.0, 2.0);
TextIO.putf("%s %.2f %n", "7.0 divided by 2.0 is", realAnswer);
```

   would output:

```
7.0 divided by 2.0 is 3.50
```

   Write a `main()` method in a separate class called `TestDivision` that presents the user with a menu along these lines:

```
1. Perform integer division with two integer values
2. Perform real division with two real values
3. Quit

Enter your choice (1-3):
```

   If the user chooses 1, the program should prompt the user to enter the two integer values and then display the result of dividing the first value by the second, calling the relevant overloaded method and giving both the quotient and the remainder, if they choose 2 the program should prompt the user to enter the two real values and display the result of dividing the first real value by the second.

   Remember that `DivisionResult` is a class so you will need to create an instance of it in your `divide(int,int)` method:

```
DivisionResult result = new DivisionResult();
```