Homework 1

CS:2230 Computer Science II: Data Structures

Deadline: 16 September 2015 (Wednesday) at 7 pm.

Instructions

This homework is published well in advance to best accommodate to everyone's schedule. The contents necessary to complete the assignment, however, will be covered until slightly after the announcement time (but before one week before the deadline). This is done so to motivate attention to lectures and discussions.

The completed homework must be submitted though ICON's dropbox tool. A submission consists of source code files (**.java**) only. Neither compressed (.zip, .rar, .tgz, etc.) nor compiled code (.class) files are accepted. Files sent by email will be ignored. Late submissions are not accepted.

All submissions are expected to be of **high quality**. The code must **compile without errors**. If, for some reason, you are not able to complete the homework on time, submit the portion of the code that compiles without errors.

This homework is to be completed **alone**. Sharing source code is considered **cheating**. You can still talk with peers about solution ideas, however. Code found online is also forbidden, and so is the use of additional libraries.

1 Three dimensional vectors

In this problem, you are to implement **three dimensional vectors of integral numbers** (of type `int`). In mathematics, vectors are arrangements of numbers, much like tuples. But we will create operations for them. You will represent them using the `Vex` class that you are to implement, following the sketch shown below:

Vex.java

```
class Vex {

  int x, y, z;

  public Vex( int x, int y, int z ) {
    this.x = x;
    this.y = y;
    this.z = z;
  }

  public String toString() {
    return "Vex( " + x + ", " + y + ", " + z + " )";
  }

  public Vex add( Vex other ) {
    int new_x = this.x + other.x,
        new_y = this.y + other.y,
        new_z = this.z + other.z;
    return new Vex( new_x , new_y, new_z );
  }

  // the methods below must be implemented by you
  public Vex subtract( Vex other )
  public Vex scalarMult( int scale )
  public int innerMult( Vex other )
  public int norm1()
  public double norm2()
  public boolean equals( Vex other )
  public Vex clone()

}
```

You might need to make use of the static methods of the `Math` class in `norm2` and possibly `norm1`.

1.1 Description of the methods

The methods of `Vex` represent normal operations with immutable vectors, i.e., no operation changes the fields of a `Vex` object. Methods:

constructor (Supplied) The constructor defines a new object of type Vex given three numbers (doubles), one representing each component of the vector. To represent vector $x = (x_0, x_1, x_2)$, we do in Java: `Vex x = new Vex(x0, x1, x2)`, where x0, x1, x2 are of type double.

toString (Supplied) Returns a textual description of the vector. Java example:

```
System.out.println( new Vex(0,1,2) ) // Vex( 0, 1, 2 )
```

add (Supplied) Addition of vectors. For Vex x,y, instruction x.add(y) generates a new vector Vex that represents x+y. Mathematically, for $x = (x_0, x_1, x_2)$ and $y = (y_0, y_1, y_2)$, then $x+y = (x_0+y_0, x_1+y_1, x_2+y_2)$. Java example:

```
Vex x = new Vex(1,0,4), y = new Vex(0,2,-1);
System.out.println( x.add(y) ) // Vex( 1, 2, 3 )
```

subtract Subtraction of vectors. For Vex x,y, instruction x.subtract(y) generates a new vector Vex that represents x-y. Java example:

```
Vex x = new Vex(1,0,4), y = new Vex(0,2,-1);
System.out.println( x.subtract(y) ) // Vex( 1, -2, 5 )
```

scalarMult Scalar multiplication. For Vex x and double a, instruction x.scalarMult(a) generates a new vector Vex that represents ax. Mathematically, if $x = (x_0, x_1, x_2)$ is a vector and λ as scalar, then $\lambda x = (\lambda x_0, \lambda x_1, \lambda x_2)$. Java example:

```
Vex x = new Vex(1,0,-3);
System.out.println( x.scalarMult(3) ) // Vex( 3, 0, -9 )
```

innerMult Inner product, a.k.a. dot product. Mathematically, for vectors $x = (x_0, x_1, x_2)$ and $y = (y_0, y_1, y_2)$, the inner product is $x \circ y = x_0y_0 + x_1y_1 + x_2y_2$, i.e., component-wise multiplication. Java example:

```
Vex x = new Vex(1,0,4), y = new Vex(0,2,-1);
System.out.println( x.innerMult(y) ) // -4
```

norm1 First norm. Mathematically, for vector $x = (x_0, x_1, x_2)$, the first norm is $||x||_1 = |x_0| + |x_1| + |x_2|$, where the $|\cdot|$ stands for the absolute value, i.e., the number without sign. Java example:

```
Vex x = new Vex(1,0,4), y = new Vex(0,2,-1);
System.out.println( x.norm1() ) // 5
System.out.println( y.norm1() ) // 3
```

norm2 Second norm, a.k.a. Euclidean norm. Mathematically, for vector $x = (x_0, x_1, x_2)$, the second norm is $||x||_2 = \sqrt{x_0^2 + x_1^2 + x_2^2}$. Java example:

```
Vex x = new Vex(1,0,4), y = new Vex(0,2,-1);
System.out.println( x.norm2() ) // 4.1231...
System.out.println( y.norm2() ) // 2.2361...
```

equals Compares two vectors Vex, component by component. The result is true only if their matching components are equal. Java example:

```
Vex x = new Vex(1,2,3), y = new Vex(1,2,3), z = new Vex(3,2,1);
x == y; // false
x.equals(y); // true
x.equals(z); // false
```

clone Returns a **new** vector Vex whose components are identical to the original vector. Java example:

```
Vex x = new Vex(1,0,4);
x == x.close() // false
x.equals( x.clone() ) // true
```

1.2 Policy on null values

Your code does not need to handle `null` values.

1.3 Policy on additional methods

You may add additional methods if you think that is convenient. These methods are best left `private` however.

1.4 Submission

Only submit file **Vex.java** for this problem.

2 Master of char[]

Create a class of static methods called charMaster in file charMaster.java, with the following tools for dealing with arrays of chars (`char[]`):

charMaster.java

```
class charMaster {
  public static String char2str(char[] x) {
    // creates a new string out of the char array, for
    // example: {'h','e','l','l','o'} --> "hello"
  }

  public static char[] str2char(String x) {
    // creates a new char array out of the String, for
    // example: "hello" --> {'h','e','l','l','o'}
  }

  public static char[] reverse(char[] x) {
    // creates a new char array that contains the elements
    // of c in reverse order, for example:
    // {'h','e','l','l','o'} --> {'o','l','l','e','h'}
  }

  public static int count(char c, char[] x) {
    // returns the number of times character c appears in
    // the array of characters x
  }

  public static char[] clone(char[] x) {
    // returns a new char array with the same elements as c
  }

  public static boolean equals(char[] x, char[] y) {
    // true if and only if x and y have the same contents
  }
}
```

If you wish, you can use methods from class `String` for this. Do not use other classes to solve this problem.

2.1 Policy on null values

Your code does not need to handle `null` values.

2.2 Policy on additional methods

You may add additional methods if you think that is convenient. These methods are best left `private` however.

2.3 Submission

Only submit file **charMaster.java** for this problem.

Suggestions

You may create a Netbeans project called Homework1 (or something alike) and have a class with a `main` method in it. You can use the `main` to test the classes you are to develop for problems 1 and 2.

When creating the classes, you can copy and paste the text from this pdf and reformat everything with `alt+shift+f` (automatic formatting) in Netbeans. Then, the first thing you should do after that is to make every non-`void` method return something, even if the methods have one only instruction inside (e.g., `return null` or something just as simple). This will ensure that your code is working at least!

Comment your code. Even better, start with *pseudocode.* This will guide you throughout your work, and will facilitate the TA's grading. Buggy code that is also obscure is much less likely to be awarded points, naturally. (Don't try the *old college try.*)