IIP Second Partial - ETSInf
Date: January 17th, 2014. Time: 2 hours and 30 minutes.

1. 6.5 points We have available the well-known `CelestialBody` class, that represents different types of celestial bodies; an except of its documentation is shown below:

Field Summary

Fields

Modifier and Type	Field and Description
static int	GALAXY Constant that indicates that the CelestialBody is of type galaxy.
static int	NEBULA Constant that indicates that the CelestialBody is of type nebula.
static int	STAR Constant that indicates that the CelestialBody is of type star.

Constructor Summary

Constructors

Constructor and Description
CelestialBody(java.lang.String n, int t, double b, double d) Create a CelestialBody with a given name, type, brightness, and distance.

Method Summary

Methods

Modifier and Type	Method and Description
boolean	equals(java.lang.Object o) Checks if the current CelestialBody is equal to one given as parameter, i.e, they have the same name, type, brigthness, and distance.
int	getType() Returns type of the current CelestialBody.
int	moreBrilliant(CelestialBody other) Returns 1 when current CelestialBody is more brilliant in absolute magnitude than a given CelestialBody, 0 if they present the same absolute magnitude, and -1 when the given CelestialBody is more brilliant in absolute magnitude that the current CelestialBody.
java.lang.String	toString() Returns a String with information about the current CelestialBody with format: "name: type (brightness, distance)"; e.g., "Sirius: Star (-1.42, 8.70)".
java.lang.String	visibleWith() Returns a String that describes the form in which the current CelestialBody can be observed ("with the naked eye", "with binoculars", "with telescope", or "with large telescope").

You must: implement the class `AstronomicCatalogue` that, as its name indicates, represents a catalogue of celestial bodies by using the following components (attributes and methods):

a) (0.5 points) Attributes:

- `MAX_CELBOD`, a constant that represents the maximum number of celestial bodies in the catalogue: 120.
- `numCelBod`, an integer in the range `[0..MAX_CELBOD]` that represents the current number of celestial bodies in the catalogue.
- `catalogue`, an array of `CelestialBody`, size `MAX_CELBOD`, whose components are sequentially stored in consecutive positions (from 0 to `numCelBod-1`)
- `numStarsNakedEye`, that represents the number of stars of the catalogue that can be seen with the naked eye

b) (0.5 points) A default constructor (with no parameters) that creates an empty catalogue, with 0 celestial bodies

c) (1 point) A method with header:

```
private int positionOf(CelestialBody c)
```

that, given a `CelestialBody c`, returns its position in the catalogue, or -1 if is not present

d) (0.5 points) A method with header:

```
private boolean isNakedEyeStar(int i)
```

that, given a correct position `i` in the catalogue ($0 \leq$ `i` $<$ `numCelBod`), returns `true` when the `CelestialBody` in that position is a star that can be seen with the naked eye, and `false` otherwise

e) (1 point) A method with header:

```
public boolean add(CelestialBody c)
```

that returns `true` after adding `CelestialBody c` to the catalogue. If there is no room to store `c` or it is already present in the catalogue, the method must return `false`. Employ private methods `positionOf(CelestialBody)` (to look for the `CelestialBody`) and `isNakedEyeStar(int)` (to properly update the `numStarsNakedEye`)

f) (1 point) A method with header:

```
public CelestialBody firstMoreBrilliantThan(CelestialBody c)
```

that returns the first `CelestialBody` of the catalogue that is more brilliant in absolute magnitude that then given `CelestialBody c`, or `null` if no one accomplishes that condition

g) (1 point) A method with header:

```
public CelestialBody [] filterNakedEyeStars()
```

that returns an array of `CelestialBody` with the starts visible with the naked eye that are present in the catalogue. The length of that array must be equal to the number of stars visible with the naked eye or 0 when no one is present in the catalogue. You must employ the private method `isNakedEyeStar(int)`

h) (1 point) A method with header:

```
public CelestialBody mostBrilliant()
```

that returns the most brilliant `CelestialBody` (in absolute magnitude) of the whole catalogue, or `null` if the catalogue is empty

Solución:

```
public class AstronomicCatalogue {
    public static final int MAX_CELBOD = 120;
    private CelestialBody[] catalogue;
    private int numCelBod, numStarsNakedEye;

    public AstronomicCatalogue() {
       catalogue = new CelestialBody[MAX_CELBOD];
       numCelBod = 0; numStarsNakedEye = 0;
    }

    private int positionOf(CelestialBody c) {
        int i = 0;
        while(i<numCelBod && !catalogue[i].equals(c)) i++;
        if (i<numCelBod) return i;
        else return -1;
    }
```

```
    private boolean isNakedEyeStar(int i) {
        return catalogue[i].getType()==CelestialBody.STAR &&
               catalogue[i].visibleWith().equals("with the naked eye");
    }

    public boolean add(CelestialBody c) {
        boolean res = false;
        int pos = positionOf(c);
        if (pos==-1) {
             if (numCelBod!=MAX_CELBOD) {
                catalogue[numCelBod++] = c;
                if (isNakedEyeStar(numCelBod-1)) numStarsNakedEye++;
                res = true;
             }
        }
        return res;
    }

    public CelestialBody firstMoreBrilliantThan(CelestialBody c) {
        int i = 0;
        while(i<numCelBod && catalogue[i].moreBrilliant(c)!=1) i++;
        if (i<numCelBod) return catalogue[i];
        else return null;
    }

    public CelestialBody[]  filterNakedEyeStars() {
        CelestialBody[] aux = new CelestialBody[numStarsNakedEye];
        int k = 0;
        for(int i=0; i<numCelBod; i++)
            if (isNakedEyeStar(i)) {
                aux[k] = catalogue[i];
                k++;
            }
        return aux;
    }

    public CelestialBody mostBrilliant() {
        CelestialBody brighter = catalogue[0];
        for(int i=1; i<numCelBod; i++)
            if (catalogue[i].moreBrilliant(brighter)==1) brighter = catalogue[i];
        return brighter;
    }
}
```

2. [1.5 points] Implement a class (`static`) method that, given an integer number $n \geq 0$, shows on the standard output (screen) the letter 'Z' in n lines, as the following figure shows (for $n = 5$); notice that first and last line have as many 'Z' characters as n, whereas the rest of lines have an only 'Z' character in the secondary diagonal

```
ZZZZZ
   Z
  Z
 Z
ZZZZZ
```

Solución:

```
/** n >= 0 */
public static void printZ(int n) {
    for (int i=0; i<n; i++) System.out.print('Z');
    System.out.println();
    for (int i=0; j<n-2; i++) {
        for (int j=0; j<n-2-i; j++) System.out.print(' ');
        System.out.println('Z');
    }
    for (int i=0; i<n; i++) System.out.print('Z');
    System.out.println();
}
```

3. [2 points] The norm $\|a\|$ of a vector $a = (a_1, a_2, \ldots, a_n)$ is given by $\|a\| = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$. The scalar product $a \cdot b$ of two vectors a and b of the same dimension is calculated as $a \cdot b = a_1b_1 + a_2b_2 + \ldots + a_nb_n$. The cosine of the angle θ delimited by a and b of the same dimension can be calculated as $\cos\theta = \frac{a \cdot b}{\|a\|\|b\|}$. Supposing that vectors are represented by arrays in Java, **you must:**
   a) Implement a private static method `norm` that returns the norm of a parameter array of integer `a`, with `a.length`>0
   b) Implement a public static method `cosine` that, using the `norm` method, returns the cosine of the angle delimited by two parameter integer arrays `a` and `b`, with `a.length`>0, `b.length`>0 and `a.length = b.length`

Solución:

```
/** a.length>0 */
private static double norm(int[] a) {
        double res = 0;
        for (int i=0; i<a.length; i++) res += a[i]*a[i];
        return Math.sqrt(res);
}
/** a.length>0, b.length>0, a.length=b.length */
public static double cosine(int[] a, int[] b) {
       double num = 0;
       for (int i=0; i<a.length; i++) num += a[i]*b[i];
       double den = norm(a)*norm(b);
       return num/den;
}
```