Segon Parcial d'IIP - ETSInf
Data: 17 de gener de 2014. Duració: 2:30 hores.

1. 6.5 punts Es disposa de la classe `Astre` (que permet representar diferents tipus d'astres), ja coneguda i de la que es mostra a continuació un resum de la seua documentació:

Field Summary

Fields

Modifier and Type	Field and Description
`static int`	`ESTEL` Constant que indica que l'`Astre` és de tipus estel.
`static int`	`GALAXIA` Constant que indica que l'`Astre` és de tipus galàxia.
`static int`	`NEBULOSA` Constant que indica que l'`Astre` és de tipus nebulosa.

Constructor Summary

Constructors

Constructor and Description
`Astre(java.lang.String n, int t, double b, double d)` Crea un `Astre` amb un nom, tipus, brillantor i distància donats.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	`equals(java.lang.Object o)` Comprova si l'`Astre` en curs és igual a un altre donat; i.e. si coincideixen en nom, tipus, brillantor i distància.
`int`	`getTipus()` Torna el tipus de l'`Astre` en curs.
`int`	`mesBrillant(Astre altre)` Torna 1 si l'`Astre` en curs és més brillant en magnitud absoluta que un `Astre` donat, 0 si tenen la mateixa magnitud absoluta i -1 si l'`Astre` donat és més brillant en magnitud absoluta que l'`Astre` en curs.
`java.lang.String`	`toString()` Torna un `String` amb la informació de l'`Astre` en curs amb el següent format: "nom: tipus (brillantor, distància)"; p.e., "Sirius: Estel (-1.42, 8.70)".
`java.lang.String`	`visibleAmb()` Torna un `String` que descriu la forma en la que l'`Astre` en curs es pot observar ("a simple vista", "amb prismàtics", "amb telescopi" o "amb grans telescopis").

Es demana: implementar la classe `CatalegAstronomic` que, com el seu nom indica, representa un catàleg d'astres mitjançant els components (atributs i mètodes) que s'indiquen a continuació.

a) (0.5 punts) Atributs:

- `MAX_ASTRES`, una constant que representa el número màxim d'astres d'un catàleg: 120.
- `numAstres`, un enter en l'interval `[0..MAX_ASTRES]` que representa el número d'astres que té el catàleg en cada moment.
- `cataleg`, un array de tipus base `Astre`, de capacitat `MAX_ASTRES`, els components del qual s'emmagatzemen seqüencialment, en posicions consecutives del catàleg des de la `0` fins la `numAstres-1`.
- `numEstelsSimpleVista`, que representa el número d'astres del catàleg que són estels visibles a simple vista.

b) (0.5 punts) Un constructor per defecte (sense paràmetres) que crea un catàleg buit, amb 0 astres.

c) (1 punt) Un mètode amb perfil:

```
private int posicioDe(Astre a)
```

que, donat un `Astre a`, torna la seua posició en el catàleg, o -1 si no està.

d) (0.5 punts) Un mètode amb perfil:

```
private boolean esEstelSimpleVista(int i)
```

que, donada una posició vàlida `i` del catàleg (0≤i<`numAstres`), torna `true` si l'`Astre` d'aquesta posició és un estel visible a simple vista, o `false` si no ho és.

e) (1 punt) Un mètode amb perfil:

```
public boolean afegeix(Astre a)
```

que torna `true` després d'afegir l'`Astre a` donat al catàleg. Si `a` no cap o ja està al catàleg, el mètode torna `false` per tal d'advertir que no s'ha pogut afegir. S'han d'usar els mètodes privats `posicioDe(Astre)` (per tal de cercar l'`Astre`) i `esEstelSimpleVista(int)` (per tal d'actualitzar l'atribut `numEstelsSimpleVista` si procedeix).

f) (1 punt) Un mètode amb perfil:

```
public Astre primerMesBrillantQue(Astre a)
```

que torna el primer `Astre` del catàleg que és més brillant en magnitud absoluta que l'`Astre a` donat, o `null` si no hi ha cap.

g) (1 punt) Un mètode amb perfil:

```
public Astre[] filtraEstelsSimpleVista()
```

que torna un array d'`Astre` amb els estels visibles a simple vista que conté el catàleg. La longitud d'aquest array serà igual al número d'estels visibles a simple vista, o 0 si no hi ha cap. S'ha d'usar el mètode privat `esEstelSimpleVista(int)`.

h) (1 punt) Un mètode amb perfil:

```
public Astre brillaMes()
```

que torna l'`Astre` que és més brillant en magnitud absoluta de tots els del catàleg, o `null` si el catàleg està buit.

Solució:

```
public class CatalegAstronomic {
    public static final int MAX_ASTRES = 120;
    private Astre[] cataleg;
    private int numAstres, numEstelsSimpleVista;

    public CatalegAstronomic() {
       cataleg = new Astre[MAX_ASTRES];
       numAstres = 0; numEstelsSimpleVista = 0;
    }

    private int posicioDe(Astre a) {
        int i = 0;
        while(i<numAstres && !cataleg[i].equals(a)) i++;
        if (i<numAstres) return i;
        else return -1;
    }
```

```
    private boolean esEstelSimpleVista(int i) {
        return cataleg[i].getTipus()==Astre.ESTEL &&
               cataleg[i].visibleAmb().equals("a simple vista");
    }

    public boolean afegeix(Astre a) {
        boolean res = false;
        int pos = posicioDe(a);
        if (pos==-1) {
             if (numAstres!=MAX_ASTRES) {
                cataleg[numAstres++] = a;
                if (esEstelSimpleVista(numAstres-1)) numEstelsSimpleVista++;
                res = true;
             }
        }
        return res;
    }

    public Astre primerMesBrillantQue(Astre a) {
        int i = 0;
        while(i<numAstres && cataleg[i].mesBrillant(a)!=1) i++;
        if (i<numAstres) return cataleg[i];
        else return null;
    }

    public Astre[] filtraEstelsSimpleVista() {
        Astre[] aux = new Astre[numEstelsSimpleVista];
        int k = 0;
        for(int i=0; i<numAstres; i++)
            if (esEstelSimpleVista(i)) {
                aux[k] = cataleg[i];
                k++;
            }
        return aux;
    }

    public Astre brillaMes() {
        Astre mesBrilla = cataleg[0];
        for(int i=1; i<numAstres; i++)
            if (cataleg[i].mesBrillant(mesBrilla)==1) mesBrilla = cataleg[i];
        return mesBrilla;
    }
}
```

2. [1.5 punts] **Es demana:** implementar un mètode estàtic que, donat un valor enter $n \geq 0$, mostre en l'eixida estàndard la lletra 'Z' en n línies, tal com s'il·lustra en la següent figura per al cas en el que $n = 5$. Noteu que la primera i darrera línia tindran n caràcters 'Z', mentre que la resta de línies en tindran només uno situat en la seua diagonal secundària.

```
ZZZZZ
   Z
  Z
 Z
ZZZZZ
```

Solució:

```
/** n >= 0 */
public static void dibuixaZ(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 punts] La norma $\|a\|$ d'un vector $a = (a_1, a_2, \ldots, a_n)$ ve donada per $\|a\| = \sqrt{a_1^2 + a_2^2 + \ldots + a_n^2}$. El producte escalar $a \cdot b$ de dos vectors a i b d'igual dimensió es calcula com $a \cdot b = a_1b_1 + a_2b_2 + \ldots + a_nb_n$. El cosinus de l'angle θ que formen dos vectors a i b d'igual dimensió es pot calcular com $cos\ \theta = \frac{a \cdot b}{\|a\|\|b\|}$.

Tenint en compte que un vector es representa com un array en Java, **es demana:**

a) Implementar un mètode estàtic privat `norma` que torne la norma d'un array d'enters `a` donat, tal que `a.length`>0.

b) Implementar un mètode estàtic públic `cosinus` que, fent ús del mètode `norma`, torne el cosinus de l'angle que formen dos arrays d'enters `a` i `b` donats, tals que `a.length`>0, `b.length`>0 i `a.length = b.length`.

Solució:

```
/** a.length>0 */
private static double norma(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 cosinus(int[] a, int[] b) {
       double num = 0;
       for(int i=0; i<a.length; i++) num += a[i]*b[i];
       double den = norma(a)*norma(b);
       return num/den;
}
```