Exercicis IIP Segon Parcial
Classes Tipus de Dades amb un array com atribut (Tema 7)

1. 6 punts Es disposa de la classe `ProgramaRadio`, que representa un programa de ràdio d'una determinada cadena, i que s'emet en algun moment entre les 00:00 i les 23:59 d'un mateix dia. Entre les dades del programa, la classe inclou l'instant d'inici i final, i el tipus de programa. Se suposa que cap programa de ràdio acaba més enllà de la mitjanit del dia en què s'inicia. Aquesta classe és coneguda d'usos previs i a continuació es mostra un resum de la seua documentació, juntament amb la de la classe `TimeInstant`:

Field Summary

Fields

Modifier and Type	Field	Description
`static int`	`MAGAZINE`	Constant per a codificar el tipus de programa magazine (2)
`static int`	`MUSICA`	Constant per a codificar el tipus de programa musical (1)
`static int`	`NOTICIES`	Constant per a codificar el tipus de programa de notícies (2)

Constructor Summary

Constructors

Constructor	Description
`ProgramaRadio(int tip, java.lang.String tit, int hora, int min, int duracio)`	Crea el programa a partir del tipus, del títol, de l'hora i minuts d'inici i de la duració en minuts.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`int`	`compareTo (ProgramaRadio p)`	Retorna un valor > 2 si l'horari d'emissió de this és posterior al de p, < 2 si és anterior al de p, i 2 si coincideixen.
`TimeInstant`	`getFi()`	Retorna l'instant de finalització.
`TimeInstant`	`getIni()`	Retorna l'instant d'inici.
`int`	`getTipus()`	Retorna el tipus de programa.

Constructor Summary

Constructors

Constructor	Description
`TimeInstant(int h, int m)`	Crea un `TimeInstant` amb `h` hores i `m` minuts.

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method	Description
`int`	`compareTo (TimeInstant other)`	Compara cronològicament el `TimeInstant` en curs amb el `TimeInstant other`, tornant un valor negatiu si `this` és anterior a `other`, zero si són iguals, i un valor positiu si `this` és posterior a `altre`.
`int`	`toMinutes()`	Torna el número de minuts transcorreguts des de les 00:00 fins el `TimeInstant` en curs.

Es demana: implementar la classe tipus de dades `Graella` que representa la graella o programació diària d'una cadena de ràdio en un dia determinat, usant els següents atributs i mètodes:

a) (0,5 punts) Atributs:

- `MAX_PROGS`: atribut de classe públic constant de tipus enter, que indica el màxim número de programes que pot haver-hi en un dia de programació, tenint en compte que la duració mínima permesa d'un programa és de 15 minuts.
- `programes`: atribut d'instància privat de tipus array de `ProgramaRadio`, i que emmagatzema els programes de la graella diària de la cadena; els programes s'han de disposar en posicions contigües de l'array des de la 0 en endavant i per ordre d'emissió, sense que hi haja solapament temporal entre ells (i.e., l'instant de finalització del programa en la posició i és menor o igual que el d'inici del programa en la posició $i+1$).
- `nProg`: atribut d'instància privat de tipus enter que indica el número de programes presents en la graella, açò és, el número de programes emmagatzemats en l'array `programes` en un moment donat.

b) (0,5 punts) Un constructor per defecte (sense paràmetres) que crea l'array i inicialitza a 0 el número de programes presents.

c) (2 punts) Un mètode amb capçalera o perfil:

```
public boolean inserir(ProgramaRadio p)
```

que intenta afegir `p` a la graella (i.e., inserir-lo en l'array `programes`), considerant com a precondició que `p` no se solapa amb cap dels programes presents en l'array. No s'inseriran programes massa breus, de manera que el número de programes presents no excedirà mai el màxim de programes que caben en la programació diària. El mètode ha d'actuar de la següent forma:

- Si el programa dura menys de 15 minuts, no s'ha d'inserir.
- En cas contrari, el programa s'ha d'inserir ordenadament, desplaçant una posició cap a la dreta els programes posteriors temporalment.

Si la inserció s'ha dut a terme, el mètode ha de retornar `true`, i `false` en cas contrari.

d) (1,5 punts) Un mètode amb capçalera o perfil:

```
public int[] numDeCadaTipus()
```

que retorne un array d'`int` que compte en cada component el número de programes d'un cert tipus (`MAGAZINE`, `MUSICA` i `NOTICIES`), que apareixen en la graella. S'entén que les components de l'array resultant vénen numerades pel tipus de programa (recordar que precisament els tipus de programa que existeixen es codifiquen per enters des de 0 endavant).

Per exemple, si els tipus dels successius programes emmagatzemats en l'array són `NOTICIES`, `MUSICA`, `MUSICA`, `MAGAZINE`, `NOTICIES`, i `MUSICA` ha de retornar l'array `{1,3,2}`.

e) (1,5 punts) Un mètode amb capçalera o perfil:

```
public int progMesLlarg()
```

que retorne la posició en l'array del programa de major duració. Com a precondició, se suposarà que en la graella hi ha almenys un programa. En cas de haver-ne més d'un amb la màxima duració, es retornarà la posició del primer d'ells.

2. [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 de classe (o estàtica) 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.

3. [6.5 punts] Es disposa de la classe `Bloc` (que permet representar blocs apilables en torres d'un joc), 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`	**`BLAU`** Constant que indica que el Bloc es de color blau
`static int`	**`ROIG`** Constant que indica que el Bloc es de color roig

Constructor Summary

Constructors

Constructor and Description
`Bloc``()` Crea un Bloc de color blau que no es un comodi i la dimensio del qual es un enter aleatori dins del rang [1,50].
`Bloc``(int color, int dimensio, boolean comodi)` Crea un Bloc amb valors de color, dimensio i comodi donats.

Method Summary

Methods

Modifier and Type	Method and Description
`boolean`	**`equals`**`(java.lang.Object o)` Comprova si el Bloc en curs es igual a un altre donat; es a dir, si coincideixen en el color i la dimensio i els dos son o no comodins.
`int`	**`getColor`**`()` Torna el color del Bloc en curs.
`boolean`	**`getComodi`**`()` Comprova si el Bloc en curs es comodi.
`int`	**`getDimensio`**`()` Torna la dimensio del Bloc en curs.
`boolean`	**`potEstarDamuntDe`**`(Bloc b)` Comprova si el Bloc en curs pot estar damunt d'un Bloc donat; un Bloc a pot estar damunt d'un Bloc b si i solament si la dimensio del Bloc a es menor o igual que la del Bloc b i, be a es un comodi, o be els colors de a i b son diferents.
`java.lang.String`	**`toString`**`()` Torna un String amb la informacio del Bloc en curs en un format com el mostrat en els seguents exemples: "(Color: roig, dimensio: 22 i SI es comodi)", "(Color: blau, dimensio: 15 i NO es comodi)".

Per a formar una torre de blocs s'han de respectar les següents regles:

- Els blocs apilats en una torre han de seguir colors alterns (damunt d'un bloc blau solament pot haver-hi un bloc roig i viceversa).
- Damunt d'un bloc de dimensió x solament pot haver-hi un bloc de dimensió y, on $y \leq x$ (la torre s'estreny cap a la punta, és a dir, s'eixampla cap a la base).
- Un bloc pot ser un **comodí**, en aquest cas pot anar damunt de qualsevol altre bloc, independentment dels seus colors. Ara bé, un bloc comodí, com qualsevol altre, ha de respectar la regla de la dimensió.

Es demana: implementar la classe `TorreBlocs` que representa una torre de blocs mitjançant els components (atributs i mètodes) que s'indiquen a continuació.

Recorda que les constants de la classe `Bloc` i de la classe `TorreBlocs` s'han d'utilitzar sempre que es requerisca.

a) (0.5 punts) Atributs:

- `MAX_BLOCS`, una constant de classe (o estàtica) que representa el número màxim de blocs d'una torre: 10.
- `numBlocs`, un enter en l'interval `[0..MAX_BLOCS]` que representa el número de blocs que té la torre en cada moment.
- `torre`, un array de tipus base `Bloc`, de capacitat `MAX_BLOCS`. Els components d'aquest array s'emmagatzemen seqüencialment seguint les regles del joc, en posicions consecutives des de la `0` fins la `numBlocs - 1`, de manera que la base de la torre serà `torre[0]` i la punta serà `torre[numBlocs - 1]`.
- `numBlocsComodi`, que representa el número de blocs de la torre que són comodí.

b) (0.5 punts) Un constructor per defecte (sense paràmetres) que crea una torre buida, amb 0 blocs.

c) (1 punt) Un mètode amb perfil:

```
private int posicioDe(Bloc b)
```

que, donat un `Bloc b`, torna la posició de la primera aparició del bloc en la torre des de la base, o `-1` si no està.

d) (1 punt) Un mètode amb perfil:

```
public boolean apilar(Bloc b)
```

que torna `true` després d'apilar el `Bloc b` donat a la torre. Si `b` no cap o no pot estar damunt del darrer bloc apilat, el mètode torna `false` per tal d'advertir que no s'ha pogut apilar. S'ha d'actualitzar l'atribut `numBlocsComodi` si procedeix.

e) (1 punt) Un mètode amb perfil:

```
public Bloc primerMesGranQue(Bloc b)
```

que torna el primer `Bloc` de la torre, des de la base, que és de dimensió més gran que el `Bloc b` donat, o `null` si no hi ha cap.

f) (1 punt) Un mètode amb perfil:

```
public Bloc[] filtrarBlocsComodi()
```

que torna un array de `Bloc` amb els blocs que són comodí que formen la torre. La longitud d'aquest array serà igual al número de blocs comodí, o 0 si no hi ha cap.

g) (1.5 punts) Un mètode amb perfil:

```
public String toString()
```

que torna "`Torre buida`" si la torre està buida o, en cas contrari, torna un `String` amb la informació dels blocs que formen la torre en un format com el que es mostra en el següent exemple per a una torre amb 5 blocs:

- `torre[0]`: bloc de color roig, dimensió 15 i no és comodí.
- `torre[1]`: bloc de color blau, dimensió 10 i no és comodí.
- `torre[2]`: bloc de color blau, dimensió 7 i sí és comodí.
- `torre[3]`: bloc de color roig, dimensió 4 i no és comodí.
- `torre[4]`: bloc de color blau, dimensió 2 i no és comodí.

El `String` resultant serà:

```
BB
RRRR
CCCCCCC
BBBBBBBBB
RRRRRRRRRRRRRR
```

on "`C`" indica que el bloc és un comodí, "`B`" que és de color blau i "`R`" que és de color roig.

Cal notar que hi ha `numBlocs` línies i en cada línia apareixen tants caràcters indicant el color/comodí com la dimensió del bloc representat.