Exercici 2 - Tècnica Voraç Examen 16 Des 2016

Un llibreter acaba d'obrir un negoci de venda de llibres de diferents temàtiques, que s'ofereixen al públic ubicades en diferents prestatgeries.
Consideracions:

- Es vol optimitzar l'espai del local i per tant es pretén minimitzar el nombre de prestatgeries a usar.
- Es imprescindible que tots els llibres d'una mateixa temàtica estiguin ubicats en una mateixa prestatgeria.
- Una prestatgeria pot estar ocupada per llibres de diferents temàtiques mentre tingui espai suficient per tots els seus corresponents llibres.
- Totes les prestatgeries tenen la mateixa capacitat.
- Tots els llibres ocupen el mateix espai.
- Si es dona el cas de què una prestatgeria no té suficient capacitat per contenir tots els llibres d'una única temàtica aquesta no s'ubicarà a la botiga.

El programa en haurà d'informar, indicant les temàtiques que no s'ha pogut ubicar i per tant no es posaran a la venda.

Es vol escriure un **programa voraç** que determini la **distribució de les temàtiques** en les diferents prestatgeries, **minimitzant el nombre** d'elles.

La funció de selecció que heu d'usar és la següent:

- L'estratègia voraç ha de recorre totes les temàtiques a col·locar i ho farà de major a menor número d'unitats de llibres alhora d'assignar la prestatgeria.
- La prestatgeria seleccionada serà aquella que tingui espai lliure suficient per la temàtica completa i que a més deixi el menor espai lliure.
- Si no hi té cabuda en cap de les prestatgeries que ja tenen temàtiques assignades aleshores agafarà una prestatgeria nova (una buida, que no conté res).
- Si la temàtica necessita més espai que la capacitat màxima d'una prestatgeria aquesta temàtica no apareixerà a la solució final.

Exercici 2 - Tècnica Voraç Examen 16 Des 2016

```
import java.util.Arrays;


public class Aplicacio {

    private Tematica[] candidats;
    //Afegir atributs per emmagatzemar la solució trobada
    private Prestatgeria[] prestatgeries;
    private int quantes;
    private String noUbicats;

    public Aplicacio(int quants) {
        candidats = new Tematica[quants];
        //Crear i inicialitzar atributs afegits
        prestatgeries = new Prestatgeria[quants]; // cas pitjor: una per cads tema
        quantes = 0;
        noUbicats="";
        omplenaDades();
    }


    public static void main(String args[]) {
        System.out.println("Indica la capacitat que tenen les prestatgeries que tens");
        Prestatgeria.capacitatTotal = 30; //Keyboard.readInt();
        System.out.println("Quantes tematiques diferents vols ubicar? ");
        int quants = 8; //Keyboard.readInt();
        // Crear un objecte Aplicacio, invoca al mètode voraç i mostra la solució trobada
        Aplicacio m = new Aplicacio(quants);
        m.solucio();

        System.out.println("\nNecessitem " + m.quantes + " prestatgeries");
        System.out.println("Distribucio:\n"+ m);
        System.out.println("Tematiques no ubicades: ");
        if (m.noUbicats!="")System.out.println("\n"+ m.noUbicats);
        else System.out.println("Totes estan ubicades");
    }


    private void omplenaDades() {
        candidats[0] = new Tematica("tema 2", 20);
        candidats[1] = new Tematica("tema 1", 25);
        candidats[2] = new Tematica("tema 0", 30);
        candidats[3] = new Tematica("tema 5", 15);
        candidats[4] = new Tematica("tema 4", 15);
        candidats[5] = new Tematica("tema 3", 5);
        candidats[6] = new Tematica("tema 6", 10);
        candidats[7] = new Tematica("tema 7", 10);
    }



    public void solucio() {
/* Implementació mètode voraç. Decidiu vosaltres el retorn i paràmetres.
* En la seva execució és imprescindible que s'invoqui al mètode següent corresponent
* a la implementació de la funció de selecció, el que determina el següent candidat
* a considerar.
*/
        // ordena de menys a mes capacitat les tematiques
        Arrays.sort(candidats);
        // recorregut invers de tots els candidats
        for (int i = candidats.length - 1; i >= 0; i--) {
            int quina = funcioSeleccio(candidats[i]);
            switch (quina) {
                case -1: // no la ubiquem
                    noUbicats += candidats[i].getNom() + "\n";
                    break;
                case -2: // estanteria nova
                    prestatgeries[quantes] = new Prestatgeria();
```

```
                    prestatgeries[quantes].addTematica(candidats[i]);
                    prestatgeries[quantes].ocuparEspai(candidats[i].getNumLlibres());
                    quantes++;
                    break;
                default: // ho possem a quina
                    prestatgeries[quina].addTematica(candidats[i]);
                    prestatgeries[quina].ocuparEspai(candidats[i].getNumLlibres());
                    break;
            }
        }
    }


private int funcioSeleccio(Tematica p) {
/* Implementació del mètode privat que aplica la funció de
* selecció indicada en l'enunciat. Determineu vosaltres el
* retorn i els paràmetres necessaris.
*/

    // buscar la prestatgeria on hi cap i sobre menys espai
    if (p.getNumLlibres() > Prestatgeria.capacitatTotal) return -1;//no hi cap
    int quina = -1;
    int espaiSobrant = Prestatgeria.capacitatTotal;
    // recorregut per les prestatgeries existents per buscar espai
    for (int j = 0; j < quantes; j++) {
        int espai = prestatgeries[j].getEspaiOcupat() + p.getNumLlibres();
        if (espai <= Prestatgeria.capacitatTotal) {
            // comparar l'espai sobrant entre les prestatgeries existents
            if (Prestatgeria.capacitatTotal - espai < espaiSobrant) {
                quina = j;
                espaiSobrant = Prestatgeria.capacitatTotal - espai;
            }
        }
    }
    if (quina == -1) return -2; // una nova
    else return quina;
}

    public String toString() {
/* Ha de crear i retornar una cadena amb la solució trobada.
* Ha d'indicar el número de prestatgeries necessàries, per a cadascuna d'elles les
* temàtiques que allotja i finalment les temàtiques que no es poden ubicar
* i per tant no es posen a la venda.
*/

        String re = "";
        for (int i = 0; i < quantes; i++) {
            re += "Prestatgeria "+ i +" hi ubiquem: \n"
            + prestatgeries[i].toString() +"\n";
        }
        return re;
    }
}
```

Exercici 2 - Tècnica Voraç Examen 16 Des 2016

```java
import java.util.ArrayList;
import java.util.List;

public class Prestatgeria {
    public static int capacitatTotal;
    private int espaiOcupat;
    private List<Tematica> contingut;

    public Prestatgeria() {
        espaiOcupat = 0;
        contingut = new ArrayList<Tematica>();
    }

    public void addTematica(String tematica, int mida) {
        this.addTematica(new Tematica(tematica, mida));
    }

    public void addTematica(Tematica p) {
        contingut.add(p);
    }

    public int getEspaiOcupat() {
        return espaiOcupat;
    }

    public void ocuparEspai(int espai) {
        espaiOcupat += espai;
    }

    public String toString() {
        String r = "";
        for (int i = 0; i < contingut.size(); i++) {
            r += contingut.get(i) + "\n";
        }
        return r;
    }
}
```

Exercici 2 - Tècnica Voraç Examen 16 Des 2016

```
public class Tematica implements Comparable {
    private String nom;
    private int numLlibres;

    public Tematica(String nom, int num) {
        this.nom = nom;
        this.numLlibres = num;
    }

    public String getNom() {
        return nom;
    }

    public int getNumLlibres() {
        return numLlibres;
    }

    public int compareTo(Object o) {
        Tematica t = (Tematica) o;
        if (numLlibres < t.numLlibres) return -1;
        if (numLlibres == t.numLlibres) return 0;
        return 1;
    }

    public String toString() {
        return "Nom de la tematica " + nom + " amb: " + numLlibres;
    }
}
```