Dossier - Exercici 22 - Solució

Apartat 1

Implementeu els mètodes pendents de la classe Soci, els tres són mètodes que gestionen els rebuts del soci, un afegeix un nou rebut (addQuota), l'altre l'elimina (remQuota) i el darrer mètode QuantesQuotesPendents) determina quantes quotes li resten per pagar al soci. Recordeu que tal com s'especifica en el codi, el magatzem de quotes només emmagatzema les que estan pendents de pagament.

Apartat 1.1

```
public void addQuota(Rebut t){ // Class Soci.java
        try{quotes.encuar(t);}
        catch(Exception e){}
}
```

Apartat 1.2

```
public void remQuota(Rebut t) throws Exception{ // Class Soci.java
        boolean trobat = false;
        Cua<Rebut> c = new CuaEnll<Rebut>(); //Cua de support
        while (!quotes.cuaBuida()){ // Recurregut fins a l'ultim enllaçat
                Rebut aux= (Rebut)(quotes.desEncuar()); // element desencuat FORÇAR casting
                if (aux.equals(t)) // aplicar el mètode equals redefinit a la classe Rebut
                        trobat=true; // no encuem l'element a la cua de suport
                else c.encuar(aux); // encuem tots els altres
        }
        quotes=c; // canviem la referencia
        if (!trobat) throw new Exception("No hi és");

}
```

Apartat 1.3

```
public int quantesQuotesPendents(){ // Class Soci.java
        int cont=0;
        try{
                Cua<Rebut> c = new CuaEnll<Rebut>(); //Cua de support
                while (!quotes.cuaBuida()){ // Recurregut fins a l'ultim enllaçat
                        c.encuar(quotes.desEncuar()); // encuar l'element desencuat
                        cont++; // increment del contador
                }
                quotes = c; // canviem la referencia
        } catch(Exception e) { }
        return cont;
}
```

Apartat 2

Implementa el mètode pendent de la classe Jugador, public void addNacionalitat(String a) throws Exception, que ha d'afegir una nova nacionalitat al jugador. Cal controlar la repetició de nacionalitat, en cas de repetició cal llançar una excepció. També cal controlar la disposició d'espai dins del magatzem.

```
public void addNacionalitat(String a) throws Exception{ // Class Jugador
        int posicion = this.hiEsNacionalitat(a);
        if (posicion != -1) throw new Exception("La nacionalitat ja hi és.");

        int posicioLliure = this.posicioLliure();
        if (posicioLliure == -1) throw new Exception("La nacionalitat ja està plena.");

        this.nacionalitats[posicioLliure] = a;
}
private int hiEsNacionalitat(String nom) { // Class Jugador
        int i = 0;
        while (i < this.nacionalitats.length) {
                if (this.nacionalitats[i] == null) ++i;
                else if (this.nacionalitats[i].equals(nom)) return i;
                else ++i;
        }
        return -1;
}

private int posicioLliure() { // Class Jugador
        int i = 0;
        while (i < this.nacionalitats.length) {
                if (this.nacionalitats[i]==null) return i;
                else ++i;
        }
        return -1;
}
```

Apartat 3

Implementa tots els mètodes pendents de la classe Equip

Apartat 3.1

de moment no té jugadors assignats

```
public Equip(String nom, int codi){ // Class Equip
        this.nom = nom;
        this.codi = codi;
        jugadors = null;
        quants = 0;
}
```

Apartat 3.2

els jugadors de l' equip ens arriben dins d' un magatzem Acb, el constructor s' ha d' encarregar de posar-los en el magatzem de la classe que té aquesta funcionalitat. Som usuaris de la interfície Acb

```
public Equip(String nom, Acb<Jugador> arbreJugadors, int codi){ // Class Equip
        this(nom, codi);

        if (arbreJugadors != null) {
                AcbEnll<Jugador> acb = (AcbEnll<Jugador>)arbreJugadors;
                Cua<Jugador> cua = acb.preordre();
                try{
                        Jugador j = cua.desEncuar();
                        while(j != null) {
                                addJugador(j);
                                j = cua.desEncuar();
                        }
                } catch(Exception e){}
        }
}
```

Apartat 3.3

Afegeix un jugador a l' equip. Cal controlar la repetició

```
public void addJugador(Jugador jug) throws Exception { // Class Equip
        if (jugadors!=null && jugadors.jug.getNom().equals(jug.getNom()))
                        throw new Exception("Repe");
        Node on=null;
        if (jugadors!=null)
                on=onEs(jug); //mètode privat que afegeixo
        if (on != null)
                throw new Exception("Ja hi és");
        else { //afegim davant de tot
                jugadors = new Node(jug, jugadors);
                quants++;
        }
}
private Node onEs(Jugador j){ // Class Equip
        // la seqüència no és buida s'ha comprovat abans de fer la crida

        boolean trobat=false;
        Node aux=jugadors;
        while (!trobat & aux.seg!=null){
                trobat=aux.seg.jug.getNom().equals(j.getNom());
                if (!trobat) aux=aux.seg;
        }
        if (trobat) return aux;
        else return null;
}
```

Apartat 3.4

Sobrecarrega del mètode anterior. Ha de tenir la mateixa funcionalitat

```
public void addJugador(String nom, String nacionalitat) throws Exception { // Class Equip
        Jugador jug = new Jugador(nom);
        jug.addNacionalitat(nacionalitat);
        this.addJugador(jug);
}
```

Apartat 3.5

Ha d'esborrar del magatzem el jugador. Cal controlar la no existència

```
public void remJugador(Jugador jug) throws Exception{ // Class Equip
        if (jugadors==null) throw new Exception("No hi és");
        if (jugadors.jug.getNom().equals(jug.getNom())){ // Mirar si es el primer
                jugadors=jugadors.seg;
                quants--;
                return;
        }
        Node a = onEs(jug); // Trobar el node
        if (a!=null){
                a.seg=a.seg.seg; // desenganxar els nodes
                quants--;
        }
        else throw new Exception(""No hi és");
}
```

Apartat 4

Implementeu els mètodes pendents de la classe Secció

Apartat 4.1

```
public String equipMesJugadors(){ // Class Seccio
        return ((AcbEnll<Equip>) equips).equipMesJugadors().getNom();
}

public String nomEquipMesJugadors() {// Class AcbEnll.java
        return (this.equipMesJugadors()).getNom();
}

public Equip equipMesJugadors(){ // Class AcbEnll.java
        if (arrel == null) return null;
        Equip esqE=null, dretE=null;

        int quants=((Equip) this.arrel.inf).getQuantsJugadors();
```

```
        if (this.arrel.esq!=null)
                esqE= (Equip) this.arrel.esq.equipMesJugadors(); // crida recursiva
        if (this.arrel.dret!=null)
                dretE= (Equip) this.arrel.dret.equipMesJugadors(); // crida recursiva

        if (esqE==null && dretE==null)
                return (Equip)this.arrel.inf;
        else if (esqE==null){
                if (quants>dretE.getQuantsJugadors())
                        return (Equip)this.arrel.inf;
                else return dretE;
        }
        else if (dretE==null){
        if (quants>esqE.getQuantsJugadors())
                return (Equip)this.arrel.inf;
        else return esqE;
        }

else{ //els 3 valors, cal determinar el major
        int qdret= dretE.getQuantsJugadors(),
        qesq= esqE.getQuantsJugadors();
        if (quants>qdret && quants>qesq)
        return (Equip)this.arrel.inf;
        else if (qesq>qdret) return esqE;
        else return dretE;
        } //fi else
}
```

Apartat 4.2

```
public boolean equals(Object o){ // Class Seccio
        if (!(o instanceof Seccio)) return false;
        return nom.equals(((Seccio)o).nom) && ((AcbEnll<Equip>)equips).equals(((Seccio)o).equips);
}
public boolean equals(Object o) { // A la classe AcbEnll
        if (!(o instanceof AcbEnll)) return false;
        AcbEnll p=(AcbEnll)o;
        if (arrel==null && p.arrel== null) return true;
        if (arrel==null && p.arrel!= null || arrel!=null && p.arrel==null) return false;
        //Sabem que els 2 són diferents de null, no són buits
        if (((Exercici22.Seccio)arrel.inf).toString() != ((Exercici22.Seccio)p.arrel.inf).toString() ) return false;

                try {
                        return ((AcbEnll)fillDret()).equals(p.fillDret()) &&
((AcbEnll)fillEsquerre()).equals(p.fillEsquerre());
                } catch (ArbreException e) {
                        e.printStackTrace();
                }
                return false;
        }
}
```

Apartat 4.3

```
public String trobaNomEquipMajorCodi() throws Exception{ // Class Seccio
        return ((AcbEnll<Equip>)equips).trobaNom();
}
public String trobaNom()  throws Exception { //classe AcbEnll
        if (arrel == null) throw new Exception("No hi ha equips");
        //s'ha d'anar cap a la dreta. Així aprofitem l'ordenació!!!
        if (arrel.dret==null) return ((Equip)arrel.inf).getNom();
        AcbEnll<E> aux = arrel.dret;
        while (aux.arrel.dret!=null) aux=aux.arrel.dret;
        return ((Equip)aux.arrel.inf).getNom();
}
```

Apartat 5

Implementeu els mètodes pendents de la classe Barça

Apartat 5.1

Construcció de magatzem buits

```
public Barça(){ // Class Barça
        socis = new Acb[10];
        for (int i=0; i<10; i++)
                socis[i]=new AcbEnll<Soci>();
        seccions=new PilaEnll();
}
```

Apartat 5.2

Afegir un nou soci al magatzem. Si ja existeix es llança una excepció

```
public void addSoci(Soci p) throws Exception{ // Class Barça
        socis[(int) (p.getNumeroSoci()%10)].Inserir(p);
}
```

Apartat 5.3

Donar de baixa un soci del magatzem. Llança una excepció si no hi és

```
public void remSoci(Soci q) throws Exception{ // Class Barça
        socis[(int) (q.getNumeroSoci()%10)].Esborrar(q);
}
```