Estructures de Dades i Algorismes

Exercicis sobre "col·leccions"

Dr. Enric Sesa i Nogueras

TecnoCampus
Escola Superior Politècnica

EXERCICIS PER REALITZAR SOBRE ECLIPSE

```java
public class Element {

	private String name;
	private int value;


	// Constructor
	public Element (String name, int value) {
		this.name = name;
		this.value = value;
	}

	// setters & getter
	public String getName() {return this.name;}
	public int getValue() {return this.value;}
	public void setValue(int newVal) {this.value=newVal;}

	// equality (overriding of superclass equals)
	public boolean equals (Object o) {
		Element other;
		try {
			other = (Element)o;
			return this.name.equals(other.name);
		}
		catch(Exception e) {
			return false;
		}
	}

	// toString (overriding of superclass toString)
	public String toString () {
		return "Element["+name+" "+value+"]";
	}

}
```

```java
public class Criatura implements Comparable{

    public static final int NEN = 10;
    public static final int NENA = 20;

    public static final int MIN_EDAT=0;
    public static final int MAX_EDAT=3;

    private String nom;
    private int edat;
    private int sexe;

    public Criatura(String nom, int edat, int sexe) {
        if (edat<MIN_EDAT || edat>MAX_EDAT)
            throw new IllegalArgumentException("edat no vàlida: "+edat);
        if (sexe!=NEN && sexe!=NENA)
            throw new IllegalArgumentException("sexe no vàlid: "+sexe);

        this.nom = nom;
        this.edat = edat;
        this.sexe = sexe;
    }

    public String getNom() {return this.nom;}
    public int getEdat() {return this.edat;}
    public int getSexe() {return this.sexe;}

    // redefinició del mètode toString heretat d'Object'
    public String toString () {
        String resultat;
        resultat = "Criatura de nom: "+nom+", té "+edat+" anys. ";
        if (sexe==NEN) resultat = resultat+"Sexe: nen";
        else resultat = resultat +"Sexe: nena";
        return resultat;
    }

    // implementació de la interfície Comparable
    public int compareTo(Object o) {
        // comparació basada en l'ordre lexicogràfic sense distingir
        // majúsucules de minúscules.
        Criatura altra = (Criatura)o;
        return this.nom.compareToIgnoreCase(altra.nom);
    }

    // redefinició del mètode equals heretat d'object.
    public boolean equals (Object o) {
        // redefinició compatible amb compareTo
        try {
            return this.compareTo(o)==0;
        }
        catch(ClassCastException e) {
            return false;
        }
    }
}
```

0. Familiaritzar-se amb Collection

Trobareu aquest exercici en el paquet ex_02. S'utilitza la classe Element, el codi de la qual teniu en una pàgina anterior. Heu de completar els punts que s'indica per tal de realitzar les accions que s'especifiquen.

```
public static void main (String [] args) {

	Element [] unsElements = {
		new Element ("Alpha", 40),
		new Element ("Omega", 20),
		new Element ("Theta", 15),
		new Element ("Delta", 12),
		new Element ("Ro", 40),
		new Element ("Delta", 12),
		new Element ("Tau", 11),
		new Element ("Omega", 33),
		new Element ("Ro", 25),
		new Element ("Beta",  15),
		new Element ("Alpha", 40),
		new Element ("Epsilon", 33),
		new Element ("Gamma", 10),
		new Element ("Tau", 18),
		new Element ("Alpha", 40),
		new Element ("Ro", 23)
	};

	// Crear una primera col·lecció que contingui tots els
	//elements de les posicions parelles de la taula unsElement.
	/* COMPLETE */

	// Crear una segona col·lecció que contingui tots els
	// elements de la taula unsElements
	//que s'anomenen "Alpha" o "Omega"
	/* COMPLETE */

	// A la segona col·lecció afegir-hi l'element de la posició
	// (índex) 3 de la taula
	/* COMPLETE */

	// De la primera col·lecció eliminar els elements
	//que són com els de la segona
	/* COMPLETE */

	// Obtenir una taula amb tots els elements de la primera
	// col·lecció. Mostrar-ne el contingut
	/* COMPLETE */
}
```

Si el vostre codi és correcte, el resultat que obtindreu serà aquest:

```
Element[Theta 15]
Element[Ro 40]
Element[Tau 11]
Element[Ro 25]
Element[Gamma 10]
```

1. Practicar els mètodes de Collection i List

Trobareu aquest exercici en el paquet ex_01. En tots els apartats s'utilitza la classe Criatura, el codi de la qual teniu en una pàgina anterior. Per a cada apartat se us dóna un punt d'entrada en execució incomplet que vosaltres heu de completar. En tots els casos heu de posar el codi en el lloc indicat i executar el programa per tal de verificar que està proporcionant el resultat que se n'espera.

A. "Passar" les nenes en una col·lecció i després comptar-les (amb metodes de Collection)
Afegiu totes les nenes de la taula població a la col·lecció lesNenes. Després digueu quantes n'hi ha.

```
public static void main (String [] args) {

        Criatura [] poblacio = {
            new Criatura("NIL",0,Criatura.NEN),
            new Criatura("EVA",1, Criatura.NENA),
            new Criatura("CARLES",0, Criatura.NEN),
            new Criatura("ONA",1, Criatura.NENA),
            new Criatura("DÍDAC",0, Criatura.NEN),
            new Criatura("NIL",1, Criatura.NEN),
            new Criatura("EVA",3, Criatura.NENA),
            new Criatura("JORDI",2, Criatura.NEN),
            new Criatura("ENIA",3, Criatura.NENA)
        };


        Collection lesNenes;
        lesNenes = new LinkedList();
        int quantesNenes;

         /* APARTAT A: afegir a lesNenes totes les nenes que hi ha
             a la taula poblacio. Després "dir" quantes nenes hi ha.*/

         /* ... poseu la vostra solució a partir d'aquí */


        // mostrar el resultat
         System.out.println();
        System.out.println("A \"lesNenes\" hi ha "+quantesNenes+" nenes en total");
        System.out.println();

}
```

```
A "lesNenes" hi ha 4 nenes en total
```

B. Posar els nens en una llista i les nenes en una altra. Després calcular-ne les edats mitjanes

Poseu tots els nens de la taula població a la llista elsNens i totes les nenes a la llista lesNenes. Després calculeu l'edat mitjana dels nens i l'edat mitjana de les nenes, iterant sobre les llistes emplenades en el pas anterior. Per tal d'iterar sobre les llistes utilitzeu els seus mètodes get.

```
public static void main (String [] args) {

        Criatura [] poblacio = {
                    new Criatura("NIL",0,Criatura.NEN),
                new Criatura("EVA",1, Criatura.NENA),
                new Criatura("CARLES",0, Criatura.NEN),
                new Criatura("ONA",1, Criatura.NENA),
                new Criatura("DÍDAC",0, Criatura.NEN),
                new Criatura("NIL",1, Criatura.NEN),
                new Criatura("EVA",3, Criatura.NENA),
                new Criatura("JORDI",2, Criatura.NEN),
                new Criatura("ENIA",3, Criatura.NENA)
        };


        List elsNens, lesNenes;
        elsNens = new LinkedList();
        lesNenes = new ArrayList();


      /* APARTAT B: poseu tots els nens de la taula població a la llista
         elsNens i totes les nenes a la llista lesNenes.
         Després calculeu l'edat mitjana dels nens i l'edat mitjana de les
         nenes, iterant sobre les llistes emplenades en el pas anterior.
         Per tal d'iterar sobre les llistes, utilitzeu els seus mètodes get
      */


        double nens = 0; // per calcular la mitjana d'edat dels nens
        double nenes = 0; // per calcula la mitjana d'edat de les nenes
        Criatura c;

        /* ... poseu la vostra solució a partir d'aquí */
```

```
L'edat mitjana dels nens és: 0.6
L'edat mitjana de les nenes és: 2.0
```

C. Eliminar els nens i deixar les nenes (amb l'iterador de Collection)

Eliminar tots els nens –les nenes no- de la col·lecció llarInfants, fent ús de l'iterador proporcionat per Collection.

```
...

        /* APARTAT C: eliminar tots els nens -les nenes no- de la
           Col·lecció llarInfants. Feu-ho obtenint l'iterador
           de la col·lecció i fent amb ell tota la feina. Si us cal,
           documenteu-vos sobre el mètode iterator (de Collection) i
           sobre el darrer mètode de la interfície Iterator. */

        /* ... Poseu aquí la vostra solució */
...
```

```
contingut de la llista després de treure els nens
--------------------------------------------------
Criatura de nom: EVA, té 1 anys. Sexe: nena
Criatura de nom: EVA, té 3 anys. Sexe: nena
Criatura de nom: ENIA, té 3 anys. Sexe: nena
Criatura de nom: ONA, té 1 anys. Sexe: nena
```

2. "Millorar" MainaderiaList. Usar Iterator

Trobareu aquest exercici en el paquet ex_02. En aquest paquet trobareu la classe MainaderiaList. Aquesta classe ja la coneixeu perquè he estat utilitzada en un exemple de classe. A la versió que es va mostrar a classe, a la implementació del mètode quantsSexe no és feia ús de l'iterador de la col·lecció. L'objectiu d'aquest exercici és simple: només cal tornar a escriure el mètode fent ús de l'iterador.

```
...

    // retorna el número de criatures del sexe especificat com a paràmetre
    public int quantsSexe (int sexe) {

        /* Exercici: a la versió mostrada a classe de la implementació
           d'aquest mètode, la iteració sobre el contingut no es feia
           utilitzant l'iterador de la col·lecció.
           Torneu a escriure el codi d'aquest mètode però ara fent ús
           de l'iterador proporcionat pel mètode iterator.

           Després, podeu executar ProvaMainaderia per veure si el resultat
           que obteniu és el mateix que s'obtenia abans */
    }
...
```

3. MainaderiaDos. Un altre magatzem de criatures

En el paquet ex_03 trobareu definida la interfície MainaderiaDos que, en molts aspectes, us recordarà la interfície Mainaderia vista a classe.

```
import java.util.*;

/* una altra interficie per a definir el que volem entendre per
   mainaderia ...*/
public interface MainaderiaDos {

    /* matricula una criatura amb el nom l'edat i el sexe
     * donats com a paràmetres. Retorna true si ha pogut matricular
     * a la criatura i false si no ha pogut.
     * Els motius pels quals no es pot matricular una criatura són
     * els següents:
     * - ja hi ha una criatura igual (equals...)
     * - el sexe o l'edat no són correctes (= el constructor de
         criatura genera una excepcio) */
    public boolean matricular(String nom, int edat, int sexe);

    /* elimina de la mainaderiaDos la criatura que té el nom indicat.
     * El resultat és la criatura desmatriculada o null si aquesta no
     * existia */

    public Criatura desMatricular(String nom);

    /* retorna, sense eliminar-la, la criatura que té el nom
     * especificat. El resultat és null si no hi ha cap criatura
     * amb aquell nom */
    public Criatura buscar (String nom);

    /* calcula l'edat mitjana de les criatures de la mainaderia */
    public double edatMitjana();

    /* calcula l'edat mitjana de les criatures de la mainaderia
     * que tenen el sexe especificat */
    public double edatMitjana(int sexe);

    /* retorna una Col·lecció amb totes les nenes de la mainaderia
     * (si no hi hagués cap nena retornaria una col·lecció buida */
    public Collection lesNenes();

    /* Igual que l'anterior però amb els nens */
    public Collection elsNens();

    /* retorna un iterador que permet recórrer la mainaderia */
    public Iterator iterator();

}
```

A. Implementació basada en List

L'objectiu d'aquest apartat és fer una implementació basada en List, de manera també semblant a com es va fer en un exemple de classe. D'aquesta implementació només en teniu el principi. La resta l'heu de fer vosaltres.

```
import p0_material.Criatura;
import java.util.*;

/* implementació de la interficie MainaderiaDos basada en List */
public class MainaderiaDosList implements MainaderiaDos{

    private List infrastructura; //llista que farà, com el seu nom
    //indica, d'infrastructura per a guardar les criatures

    public MainaderiaDosList() {
        // constructor
        /* ENCARA S'HA DE COMPLETAR */
    }

    /* CAL IMPLEMENTAR TOTS ELS MÈTODES DEFINITS A LA INTERFICIE */

}
```

B. Posar a prova la implementació

Ara heu de posar a punt un petit programa que permeti una prova inicial molt superficial de la implementació que heu fet de la interfície MainaderiaDos. El programa ja està començat però vosaltres l'heu de finalitzar.

```
public class ProvaImplementacio {
    public static void main (String [] args) {
        Criatura [] poblacio = {
            new Criatura("NIL",0,Criatura.NEN),
            new Criatura("PERE",1, Criatura.NEN),
            new Criatura("NEUS",0, Criatura.NENA),
            new Criatura("ONA",1, Criatura.NENA),
            new Criatura("DÍDAC",0, Criatura.NEN),
            new Criatura("MARIONA",1, Criatura.NENA),
            new Criatura("EVA",3, Criatura.NENA),
            new Criatura("FIONA",2, Criatura.NENA),
            new Criatura("ENIA",3, Criatura.NENA)
        }; // dades per la prova. Evidentment se'n poden afegir més

        MainaderiaDosList mainaderia; // la mainaderia que es vol provar
        boolean matriculat;
        /* SI CALEN MÉS VARIABLES ES PODEN DECLARAR AQUÍ */

        mainaderia = new MainaderiaDosList(); // creacio

        //càrrega
        for (int i=0; i<poblacio.length; i++) {
            matriculat=mainaderia.matricular(poblacio[i].getNom(),
                                             poblacio[i].getEdat(),
                                             poblacio[i].getSexe()
                                            );
            if (!matriculat) {
                System.out.println("\nSembla que no hem pogut matricular a");
                System.out.println(poblacio[i]);
            }
        }

        /* CONTINUAR A PARTIR D'AQUEST PUNT */
    }
}
```

Aquesta prova hauria d'incloure els següents punts:

- Aconseguir un iterador sobre la mainaderia i amb l' ajut d'aquest mostrar-ne el contingut per a poder visualitzar que, efectivament, és el que ha de ser.
- Calcular l'edat mitjana de totes les criatures, la dels nens i la de les nenes i mostrar-les per a poder veure que el resultat és l'esperat.
- Obtenir una col·lecció amb tots els nens i mostrar-ne el contingut.
- Obtenir una col·lecció amb totes les nenes i mostrar-ne el contingut.
- Buscar alguna criatura que sí sigui a la mainaderia
- Buscar alguna criatura que no sigui a la mainaderia
- Desmatricular algunes criatures (per exemple les qu e ocupen posicions parells a la taula població)

- Tornar a aconseguir un iterador i amb el seu ajut t ornar a mostrar el contingut de la mainaderia per veure que les criatures desmatriculades ja no hi són

Serà una bona idea que aneu construint aquesta prov a de forma incremental, de manera que no codifiqueu la prova d'un punt fins que la del punt anterior no hagi produït el resultat esperat.

4. MainaderiaTres. I encara un altre magatzem de criatures.

En el paquet ex_04 trobareu definida la interfície MainaderiaTres que us recordarà a la interfície Mainaderia definida a classe i també a la interfície MainaderiaDos de l'exercici precedent. Les diferències significatives estan marcades.

```
import java.util.*;

/* una altra interficie per a definir el que volem entendre per
   mainaderia ... Aquesta utilitza la notació <E> per garantir que
   les col·leccions que retorna ho són de només criatures*/
public interface MainaderiaTres {

    /* matricula una criatura amb el nom l'edat i el sexe
     * donats com a paràmetres. Retorna true si ha pogut matricular
     * a la criatura i false si no ha pogut.
     * Els motius pels quals no es pot matricular una criatura són
     * els següents:
     * - ja hi ha una criatura igual (equals...)
     * - el sexe o l'edat no són correctes (= el constructor de criatura
     *   genera una excepcio) */
    public boolean matricular(String nom, int edat, int sexe);

    /* elimina de la mainaderiaDos la criatura que té el nom indicat.
     * El resultat és la criatura desmatriculada o null si aquesta no
     * existia */

    public Criatura desMatricular(String nom);

    /* retorna, sense eliminar-la, la criatura que té el nom
     * especificat. El resultat és null si no hi ha cap criatura
     * amb aquell nom */
    public Criatura buscar (String nom);

    /* calcula l'edat mitjana de les criatures de la mainaderia */
    public double edatMitjana();

    /* calcula l'edat mitjana de les criatures de la mainaderia
     * que tenen el sexe especificat */
    public double edatMitjana(int sexe);

    /* retorna una Col·lecció amb totes les nenes de la mainaderia
     * (si no hi hagués cap nena retornaria una col·lecció buida */
    public Collection<Criatura> lesNenes();

    /* Igual que l'anterior però amb els nens */
    public Collection<Criatura> elsNens();

    /* retorna un iterador que permet recórrer la mainaderia */
    public Iterator<Criatura> iterator();

}
```

A. Implementació basada en Set

L'objectiu d'aquest apartat és fer una implementació de la interfície basada en Set tot aprofitant, en la mesura del possible els "avantatges" que ofereixen les versions de java a partir de la 1.5. Aquesta implementació està començada i vosaltres l'heu d'acabar.

```
import java.util.*;

/* implementació de la interficie MainaderiaTres basada en Set */
public class MainaderiaTresSet implements MainaderiaTres,
Iterable<Criatura>{

    private Set<Criatura> infrastructura; //Conjunt que farà, com el
   // seu nom indica, d'infrastructura per a guardar les criatures

    public MainaderiaTresSet() {
        // constructor
        /* ENCARA S'HA DE COMPLETAR */

    }

    /* CAL IMPLEMENTAR TOTS ELS MÈTODES DEFINITS A LA INTERFICIE */
    // Excepte iterator que ja el teniu implementat de regal

    /* retorna un iterador que permet recórrer la mainaderia */
    public Iterator<Criatura> iterator() {
        return infrastructura.iterator();
    }

}
```

C. Posar a prova la implementació

Ara heu de posar a punt un petit programa que permeti una prova inicial molt superficial de la implementació que heu fet de la interfície MainaderiaTres. El programa ja està començat però vosaltres l'heu de finalitzar.

Aquesta prova hauria d'incloure els següents punts:

- Recórrer la mainaderia per a visualitzar-ne el contingut i, així, poder veure que, efectivament conté el que ha de contenir. Fer servir una iteració for-each.
- Calcular l'edat mitjana de totes les criatures, la dels nens i la de les nenes i mostrar-les per a poder veure que el resultat és l'esperat.
- Obtenir una col·lecció amb tots els nens i mostrar-ne el contingut (iterant amb for-each)
- Obtenir una col·lecció amb totes les nenes i mostrar-ne el contingut (iterant amb for-each)
- Buscar alguna criatura que sí sigui a la mainaderia
- Buscar alguna criatura que no sigui a la mainaderia
- Desmatricular algunes criatures (per exemple les que ocupen posicions parells a la taula posició, com s'ha fet a l'execució que es mostra).
- Recórrer la mainaderia per a tornar-ne a mostrar el contingut de tal manera que es vegi que les criatures desmatriculades ja no hi són.

Serà una bona idea que aneu construint aquesta prova de forma incremental, de manera que no codifiqueu la prova d'un punt fins que la del punt anterior no hagi produït el resultat esperat.

```
Contingut de la mainaderia
 Criatura de nom: DÍDAC, te 0 anys. Sexe: nen
 Criatura de nom: ENIA, te 3 anys. Sexe: nena
 Criatura de nom: EVA, te 3 anys. Sexe: nena
 Criatura de nom: FIONA, te 2 anys. Sexe: nena
 Criatura de nom: MARIONA, te 1 anys. Sexe: nena
 Criatura de nom: NEUS, te 0 anys. Sexe: nena
 Criatura de nom: NIL, te 0 anys. Sexe: nen
 Criatura de nom: ONA, te 1 anys. Sexe: nena
 Criatura de nom: PERE, te 1 anys. Sexe: nen

L'edat mitjana de les nenes és: 1.6666666666666667

L'edat mitjana dels nens és: 0.3333333333333333

L'edat mitjana de totes les criatures és: 1.2222222222222223

Els nens de la mainaderia són:
 Criatura de nom: DÍDAC, te 0 anys. Sexe: nen
 Criatura de nom: NIL, te 0 anys. Sexe: nen
 Criatura de nom: PERE, te 1 anys. Sexe: nen

Les nenes de la mainaderia són:
 Criatura de nom: ENIA, te 3 anys. Sexe: nena
 Criatura de nom: EVA, te 3 anys. Sexe: nena
 Criatura de nom: FIONA, te 2 anys. Sexe: nena
 Criatura de nom: MARIONA, te 1 anys. Sexe: nena
 Criatura de nom: NEUS, te 0 anys. Sexe: nena
 Criatura de nom: ONA, te 1 anys. Sexe: nena

A la mainaderia hi ha una criatura de nom EVA
  Criatura de nom: EVA, te 3 anys. Sexe: nena

A la mainaderia no hi ha cap RAMONA

Després de la desmatriculació, el contingut de la mainaderia es:
  Criatura de nom: FIONA, te 2 anys. Sexe: nena
  Criatura de nom: MARIONA, te 1 anys. Sexe: nena
  Criatura de nom: ONA, te 1 anys. Sexe: nena
  Criatura de nom: PERE, te 1 anys. Sexe: nen
```

5. El diccionari de sinònims. Implementació basada en Map.

En el paquet ex_05 trobareu definida la interfície DiccionariSinònims

```
import java.util.*;

public interface DiccionariSinonims {

    /* afegir un sinònim d'una paraula. Retorna true si efectivament
     * s'ha pogut afegir i false si no s'ha pogut fer -perquè el sinònim
     * ja era conegut per a aquella paraula. Les implementacions d'aquest
     * mètode no haurien de ser sensibles a la distinció entre majúscules
     * i minúscules */
    boolean afegir(String paraula, String sinonim);

    /* afegeix a la paraula tots els sinònims del conjunt. Retorna true
     * si algun dels sinònims s'ha pogut afegir a la paraula i false en
     * cas contrari Les implementacions d'aquest mètode no haurien de ser
     * sensibles a la distinció entre majúscules i minúscules */
    boolean afegir(String paraula, Set<String> sinonims);

    /* recupera els sinònims coneguts de la paraula donada com a paràmetre.
     * La taula ha de tenir la mida exacta (tantes posicions com sinònims
     * té la paraula. Si la paraula no té cap sinònim -no és coneguda- llavors
     * la taula ha de tenir mida (length) 0. Les implementacions d'aquest
     * mètode no haurien de ser sensibles a la distinció entre majúscules
     * i minúscules */
    String [] recuperar (String paraula);

    /* Retorna una taula amb totes les paraules de les quals se'n tenen
     * sinònims. La taula ha de tenir la mida exacta (tantes posicions com
     * parules tinguin sinònims. Si el diccionari es buit llavors la taula
     * ha de tenir mida (length) 0 */
    String [] paraulesConegudes ();

}
```

L'objectiu d'aquest exercici és fer-ne una implementació que estigui basada en la interfície Map de java.util i, en concret, en la classe TreeMap que és una de les implementacions de Map.

Aquesta implementació ja està "començada" i la vostra feina consisteix en completar-la. En el codi que se us subministra i trobareu tot un seguit de comentaris /* COMPLETAR ... */ que indiquen en quin lloc heu d'afegir codi.

Per tal que una vegada hàgiu completat el codi el pugueu posar a prova, se us subministra un petit programa que permet de manipular un diccionari de sinònims. Utilitzeu aquest programet per a veure si la vostra implementació sembla fer el que d'ella se n'espera.

```java
import java.util.*;

/* Implementació de la interfície DiccionariSinonims basada en un map */

public class DiccioAmbMap implements DiccionariSinonims{

    private Map<String, Set<String>> infrastructura;

    /* constructor */
    public DiccioAmbMap () {
        // només cal crear la infrastructura
        /* COMPLETAR: useu una instància de la classe TreeMap */
    }

    public boolean afegir(String paraula, String sinonim) {
        String pMaj = paraula.toUpperCase();
        String sMaj = sinonim.toUpperCase();
        // si la paraula pMaj existeix, en recuperem el conjunt de
        // sinònims i provem d'afegir-hi el que ens han donat sMaj
        if (/* COMPLETAR: pMaj és una clau a la infrastructura? */) {
            Set<String> sinonims = /* COMPLETAR: recupetar els sinònims */
            /* COMPLETAR: afegir sMaj als sinònims i retornar */
        }
        else {
            // si no existeix li vinculem un nou conjunt de sinònims que
            // de moment només conté el sinònim donat
            Set<String> nou = new TreeSet<String>();
            /* COMPLETAR: vincular pMaj amb nou */
            return nou.add(sMaj);
        }
    }

    public boolean afegir(String paraula, Set<String> sinonims) {
        // Proveu d'implementat aquest mètode sense fer ús de l'anterior.
        // una iteració for-each sobre el segon paràmetre us serà útil.
        // Recordeu que la paraula (primer paràmetre) pot ser nova o no.
        /* COMPLETAR */
    }

    public String[] recuperar(String paraula) {
        String pMaj = paraula.toUpperCase();
        String [] resultat;
        // si la paraula hi és, recuperem el seu conjunt de sinònims i el
        // "volquem" en una taula de nova creació
        if (this.infrastructura.containsKey(pMaj)) {
            Set<String> sinonims = this.infrastructura.get(pMaj);
            /* COMPLETAR: crear la taula resultat i omplir-la amb el contingut
             * de sinonims */
         }
         else {
            // i si no hi és ens limitem a crear una taula de mida 0
            resultat = new String[0];
         }
         return (resultat);
    }

    public String[] paraulesConegudes() {
        /* COMPLETAR: declarar la variable paraules de tal manera que pugui
         * referenciar un conjunt de claus -Strings- */
        String [] resultat;
        paraules = /* COMPLETAR: recuperar el conjunt de claus */
        /* COMPLETAR: crear la taula resultat i omplir-la amb el contingut
         * del conjunt de claus */

        return(resultat);
    }
}
```

EXERCICIS
“CURTS” (A)

0. Collection...

Pel que fa al programa que teniu a continuació digueu quin resultat tindrà quan arribi al punt marcat com a `/* POINT 1 */`. El mateix quan arribi al punt marcat `/* POINT 2 */` .

```
public static void main (String [] args) {

	Element [] unsElements = {
		new Element ("Alpha", 40),
		new Element ("Omega", 20),
		new Element ("Theta", 15),
		new Element ("Delta", 12),
		new Element ("Ro", 40),
		new Element ("Delta", 12),
		new Element ("Tau", 11),
		new Element ("Gamma", 10),
		new Element ("Omega", 18),
		new Element ("Alpha", 40),
		new Element ("Ro", 23),
		new Element ("Theta", 15)
	};

	Collection elsUns, elsAltres;
	Object [] contingut;

	elsUns = new LinkedList();
	elsAltres = new ArrayList();

	for (int i=1; i<unsElements.length; i=i+3) { // beware: 1, 4, 5, ...
		elsUns.add(unsElements[i]);
	}

	for (int i=0; i<unsElements.length; i=i+2) { //beware 0, 2, ...
		elsAltres.add(unsElements[i]);
	}

	elsAltres.add(unsElements[0]);
	elsAltres.add(unsElements[1]);
	elsAltres.add(unsElements[2]);
	elsAltres.removeAll(elsUns);

	contingut = elsAltres.toArray();
	for (int i=0; i<contingut.length; i++) {
		System.out.println(contingut[i]);
	}
	/* POINT 1 */

	elsUns.remove(new Element("Omega", 33));
	System.out.println(elsUns.size());
	/* POINT 2 */
}
```

Solució:

en arribar al primer punt haurà escrit

```
Element[Alpha 40]
Element[Theta 15]
Element[Tau 11]
Element[Alpha 40]
Element[Theta 15]
```

en arribar al segon haurà escrit el número **3**

1. Collection

Considereu que disposeu d'una col·lecció anomenada total

```
Collection total;
```

Considereu també que ja està creada i emplenada

Doneu el codi necessari per a

a) Comptar el número d'objectes de la classe Animal (o derivades) que conté.
    a. Feu-ho amb el seu iterador
    b. Feu-ho amb una iteració for-each

b) Eliminar tots els objectes de la classe Animal (o derivades) que pugui contenir.

c) Proporcionar una col·lecció que contingui, sense repeticions, tots els animals (objectes de la classe animal o derivades) que hi ha a total (sense eliminar-los de total)

d) Igual que l'anterior però eliminant-los de total. No és tan senzill com sembla. Mireu-vos la documentació de remove...

2. List<t>

Considereu que disposeu de la següent llista anomenada parcInfantil

```
List<Criatura> parcInfantil;
```

Considereu també que ja està creada i emplenada.

Doneu el codi necessari per a

a) Intercanviar les criatures que hi ha a la primera i a la darrera posició

b) Comptar quantes nenes hi ha a les 20 darreres posicions

c) Determinar la posició del nen de més edat (la primera posició, si n'hi ha més d'un).

d) Suposem ara que la llista està ordenada (ordre natural de les criatures). "Arriba" una nova criatura **c**. Col·locar-la en el lloc de la llista que li pertoca. Ull, la llista pot ser buida.

3. Set<t> i SortedSet<t>

Considereu que disposeu dels següents conjunts (considereu també que ja estan creats i emplenats):

```
Set<Criatura> pati;
SortedSet<Criatura> fila;
```

Doneu el codi necessari per a

a) Afegir a pati totes les criatures que hi ha a fila (sense que a pati hi quedin criatures repetides)

b) Eliminar les cinc primeres criatures de fila i posar-les a pati. Ull, pot ser que fila contingui menys de cinc criatures.

c) Determinar quines són les criatures que són a pati i a fila simultàniament (sense alterar cap del dos conjunt)

d) Determinar quines són les criatures que són a pati però no són a fila (sense alterar cap dels dos conjunts)

4. Map<t,v>

Considereu que disposeu del següent map, anomenat serveiPediatria. Les claus representen pediatres (els seus noms) i els valors representen totes les criatures que són ateses per ell/a.

```
Map<String, Set<Criatura>> serveiPediatria;
```

Considereu també que ja està creat i emplenat.

Doneu el codi necessari per a

a) Comptar el número total de criatures que són ateses en el servei

b) Comptar quants pediatres atenen a més de 50 criatures

c) Proporcionar totes les nenes (els nens no) que són ateses per Dr. “Vilajoliu”

d) Donada una criatura **c** determinar quin és el pediatra que l’atén.

e) Proporcionar tots els pediatres que només atenen nens (no atenen a cap nena)

f) Determinar el pediatra (un de sol) que atén un número més gran de criatures

g) Determinar els pediatres (pot haver n’hi més d’un) que atenen un número més gran de criatures.

h) Fer que totes les nenes que són ateses per la Dra. “Sanchís” passin a ser ateses per la Dra. “Robert”.

EXERCICIS
“CURTS” (B)

```
public class Llibre extends ... implements ... {

	...
	public int getPags() {...}
	public void setPags(int pags) {...}
	public int getPopularitat(){...}
	public void setPopularitat(int popularitat) {...}
	public String getISBN(){...}
	...
}
```

1. Collection

Considereu que disposeu d'una col·lecció anomenada estanteria...

```
Collection estanteria;
```

Considereu també que ja està creada i emplenada amb objectes de diferents classes entre les quals hi ha la classe Llibre.

a) Doneu el codi necessari per comptar els llibres que hi ha a estanteria

b) Doneu el codi necessari per eliminar d'estanteria tot el que no siguin llibres

c) Doneu el codi necessari per eliminar d'estanteria tot el que no siguin llibres de més de 100 pàgines

d) Doneu el codi necessari per a comptar les aparicions del(s) llibre(s) que té(nen) l'ISBN "0023-3344-PL"

e) Doneu el codi necessari per a incrementar en 10 unitats la popularitat dels llibres que la tenen per sota de 5.

2. Collection<t>

Ara la declaració d'estanteria la restringeix a només poder contenir llibres

```
Collection<Llibre> estanteria;
```

a) Té algun sentit plantejar-se d'eliminar d'estanteria els objectes que no sigui llibres?

b) Doneu el codi necessari per a comptar quants llibres hi ha a estanteria

c) Doneu el codi necessari per a obtenir d'estanteria (sense modificar-ne el contingut) tots els llibres que tenen més de 100 pàgines. Sense repeticions

d) Igual que abans però ara cal eliminar d'estanteria aquests llibres (i si estan repetits cal eliminar-ne totes les aparicions)

3. SortedSet<t>

Considereu ara

```
SortedSet<Llibre> topVendes;
```

a) Doneu el codi necessari per a eliminar els cinc primers llibres de topVendes

b) Doneu el codi necessari per a obtenir (sense modificar topVendes) una col·lecció que contingui el top-ten (els 10 primers llibres de topVendes)

c) Té sentit plantejar-se d'intercanviar el primer i el darrer llibre de topVendes? Si la resposta és afirmativa doneu-ne el codi necessari. Si és negativa raoneu-ne el perquè.

4. Map<t,v>

Considereu que disposeu del següent map, que aparella autors (objectes de la classe Autor) amb els seus llibres.

```
Map<Autor, Set<Llibre>> biblioteca
```

Considereu també que ja està creat i emplenat.

Doneu el codi necessari per a

a) Comptar quants autors diferents hi ha a la biblioteca

b) Comptar quants llibres diferents hi ha a la biblioteca

c) Determinar l'autor que té més llibres

d) Determinar el llibre que té més autors

EXERCICIS
“TEST”

Tots els exercicis es basen en aquesta classe. És important tenir clar com els NamedCounters es comparen entre ells i quan dos NamedCounters es consideren iguals...

```
public class NamedCounter implements Comparable<NamedCounter> {

	private int value;
	private String name;

	public NamedCounter(int value, String name) {
		this.value = value;
		this.name = name;
	}

	public int getValue () {return this.value;}
	public String getName () {return this.name;}

	public int compareTo (NamedCounter other) {
		return this.value==other.value ?
			this.name.compareToIgnoreCase(other.name) :
			this.value - other.value;
	}

	public boolean equals (Object o) {
		try {
			return this.compareTo((NamedCounter)o) == 0;
		}
		catch (ClassCastException cce) {
			return false;
		}
	}

	public void plus () {
		this.value++;
	}

	public void minus () {
		this.value--;
	}

	public String toString () {
		return this.name.toUpperCase()+"("+this.value+")";
	}

}
```

1.

Considereu el següent codi

```
List firstCol = new LinkedList();
List secondCol = new Vector();

firstCol.add(new NamedCounter(0,"Zero"));
firstCol.add(new NamedCounter(1,"One"));
firstCol.add(new NamedCounter(2,"Two"));
firstCol.add(new NamedCounter(3,"Three"));
firstCol.add(new NamedCounter(4,"four"));

secondCol.add(new NamedCounter(1,"One"));
secondCol.add(new NamedCounter(2,"Two"));
secondCol.add(firstCol.get(3));
secondCol.add(new NamedCounter(3, "FOUR"));

for (Object o : secondCol) {
      ((NamedCounter) o).plus();
}

firstCol.retainAll(secondCol);

int total = 0;

for (Object o : firstCol) {
      total = total + ((NamedCounter) o).getValue();
}

System.out.println(total);
```

Què escriurà aquest fragment de codi (quin serà el valor de la variable total)?

a) Aquest codi no escriurà res perquè no es pot compilar ja que la invocació del mètode retainAll no és correcta
b) 6
c) 7
d) 8
e) 0
f) Cap de les anteriors

2.

Considereu el següent codi

```
NamedCounter one = new NamedCounter(1, "one");
NamedCounter two = new NamedCounter(2, "two");
NamedCounter three = new NamedCounter(3, "three");

List<NamedCounter> oneList = new Vector<NamedCounter>();
List<NamedCounter> anotherList = new Vector<NamedCounter>();

oneList.add(one);
oneList.add(three);
anotherList.addAll(oneList);
oneList.add(two);
anotherList.add(oneList.remove(0));
oneList.get(1).minus();

for (NamedCounter p : oneList) {
      System.out.print(p+" ");
}
for (NamedCounter p : anotherList) {
      System.out.print(p+" ");
}
```

a) Aquest codi no escriurà res perquè no es pot compilar ja que el valor retornat pel mètode remove no és del tipus escaient
b) THREE(3) TWO(1) ONE(1) THREE(3) TWO(1)
c) THREE(3) TWO(2) ONE(1) THREE(3) TWO(2)
d) THREE(3) TWO(1) ONE(1) THREE(3) TWO(2)
e) THREE(3) TWO(1) ONE(1) THREE(3) ONE(1)
f) THREE(3) TWO(2) ONE(1) THREE(3) ONE(1)
g) THREE(2) TWO(2) ONE(1) THREE(2) ONE(1)
h) El codi escriu alguna cosa, però no és cap de les anteriors

3.

Considereu el següent fragment de codi que pot ser compilat i executat sense cap problema:

```
Map<String, SortedSet<NamedCounter>> aMap;
aMap = new HashMap<String, SortedSet<NamedCounter>>();

TreeSet<NamedCounter> s1 = new TreeSet<NamedCounter>();
TreeSet<NamedCounter> s2 = new TreeSet<NamedCounter>();

s1.add(new NamedCounter(1, "one"));
s1.add(new NamedCounter(2, "two"));

s2.add(new NamedCounter(1, "oNE"));
s2.add(new NamedCounter(3, "three"));
s2.addAll(s1);

aMap.put("First", s1);
aMap.put("Second", s2);
aMap.put("Third", aMap.get("Second"));

for (String k: aMap.keySet()) {
      for(NamedCounter nc : aMap.get(k)) {
            nc.plus();
      }
}

int total = 0;
for (SortedSet<NamedCounter> ts : aMap.values()) {
      for (NamedCounter nc : ts) {
            total = total + nc.getValue();
      }
}

System.out.println(total);
```

Què escriurà aquest fragment de codi (quin serà el valor de la variable total)?

a) 33
b) 20
c) 30
d) 17
e) 31
f) 18
g) Cap de les anteriors no és correcta.