UD8.2- ESTRUCTURES DE DADES DINÀMIQUES
Col·leccions (Conjunts)

Programació – **1er DAW/DAM**

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

CP FP BATOI

C/ Societat Unió Musical, 8 - 03802 Alcoi
Tel. 966 52 76 60 - Fax 966 52 76 61
03012165.secret@gva.es
www.cipfpbatoi.es

GOBIERNO DE ESPAÑA
MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

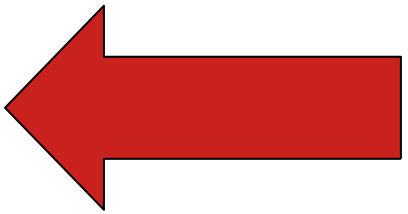
UNIÓ EUROPEA

CONTINGUTS

- **Set (HashSet)**
  - Detecció d'elements repetits
  - Recorregut
  - Ordre dels elements

1. La interfície `Set`

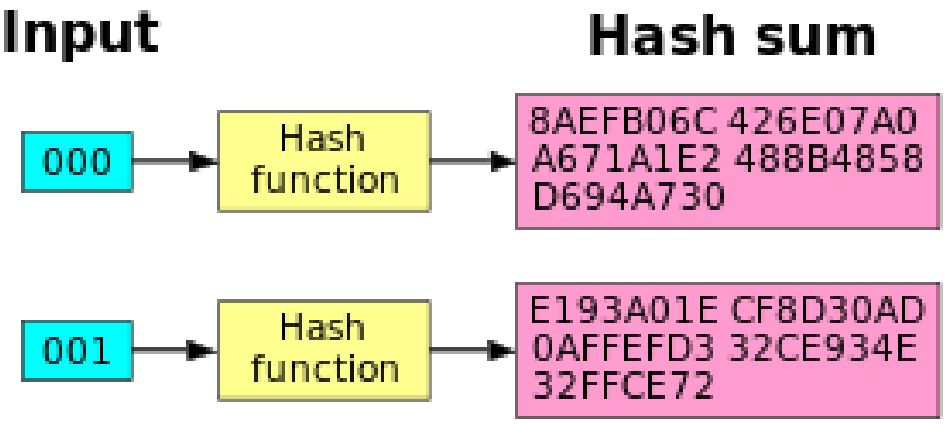
- Representa un conjunt de **tipus de col·leccions** que no permeten tenir **elements duplicats**
  - Cada **element** ha de ser **únic** per la qual cosa si al intentar afegir un element, aquest ja existeix no s'afegirà.
- La API de Java ofereix **2 implementacions:**
  - **HashSet**: Conjunt no ordenat d'elements.
  - **TreeSet:** Conjunt ordenat d'elements.



2. La classe `HashSet`

Com assegurar la unicitat dels elements?

- Basa el seu funcionament en les **taules de dispersió (hashing).**
- Per **cada objecte** a **Java** es genera un codi que l'identifica dins del sistema **hashCode.**
  - El **hashCode** és **generat** per defecte mitjançant el mètode `hashCode()` de la classe `Object`



3. Detecció d'elements repetits

A l'hora d'inserir un nou objecte sol·licita el codi de dispersió (mètode **hashCode**):

1. Si el **codi** de dispersió **no és a la taula**, a les hores s'**insereix l'objecte** (no està repetit).
2. Si el **codi és a la taula** (col·lisió), **consulta si** l'objecte amb el mateix codi de dispersió que ja és a la taula **és igual** (mètode `equals`) que l'objecte que es vol inserir.
3. En cas de ser **iguals**, es **descarta** per ser repetit.

Proporciona un accés **eficient** als **elements** de la **col·lecció**

3. Detecció d'elements repetits

Classes envolvents (Wrapper Class)

- Si els elements són **objectes de les classes envolvents (wrapper class)** de tipus primitius no ens hem de preocupar.
  - La implementació, per defecte, dels mètodes `hashCode` i `equals` serà suficient.

```
public static void main(String[] args) {
    // > Creació d'un conjunt buit de sencers
    HashSet<Integer> conjunt =new HashSet<>();
    // > Inserció d'elements
    conjunt.add(2);
    conjunt.add(3);
    conjunt.add(2); // Element repetit -> Es descarta
    System.out.println(conjunt.size()); //La mida serà 2
}
```

3. Detecció d'elements repetits

Classes Genèriques

- Per a qualsevol altre tipus d'objectes, haurem de dur a terme una implementació consistent dels mètodes `hashCode` i `equals`.
  - `hashCode` i `equals` han de tornar **true / false** per a les mateixes situacions.
  - La classe `Objects` ofereix un mètode estàtic `hash(Object... values)`que ens permet generar el `hashCode` ràpidament.
  - L'entorn de desenvolupament ens proporciona una implementació robusta d'ambdós mètodes

3. Detecció d'elements repetits

Exemple `hashCode` i `equals`

```
public class Persona {

    private String dni;

    private String name;

    private boolean isMarried;
```

```
    @Override
    public boolean equals(Object obj) {
        if(this == obj) {
            return true;
        }
        if(!(obj instanceof Persona)) {
            return false;
        }
        Persona persona = (Persona)obj;
        return isMarried == persona.isMarried
            && dni.equals(persona.dni)
            && name.equals(persona.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(dni, name, isMarried);
    }

}
```

4. Recorregut

- A diferència de les llistes, els **elements** d'un conjunt **no** es poden **recuperar per posició.**
- Per recuperar elements d'un conjunt **cal** utilitzar **un iterador o** la seva versió alternativa mitjançant **un** recorregut **for each.**

foreach	iterador

`// > Creació d'un conjunt`
`HashSet<Integer> conjunt =new HashSet<>();`
`// > Inserció d'elements`
`conjunt.add(2);`
`conjunt.add(3);`
 	
`for(Integer element: conjunt) {`
`  System.out.println(element);// 2,3`
`}`
	`// > Creació d'un conjunt buit`
	`HashSet<Integer> conjunt =new HashSet<>();`
	`// > Inserció d'elements`
	`conjunt.add(2);`
	`conjunt.add(3);`
	
	`Iterator iter = conjunt.iterator();`
	`while(iter.hasNext()) {`
	`  System.out.println(iter.next());//2,3`
	`}`

5. Ordre dels elements

- **L'ordre** en què es **recuperen** els objectes d'un conjunt durant el recorregut **no necessàriament coincideix** amb l'**ordre** en què van ser **inserits**.
  - L'**ordre** es defineix per la seva posició a la **taula de dispersió (hash)**.

```
public static void main(String[] args) {
    // Creació d'un conjunt buit
    HashSet<Integer> conjunt = new HashSet<>();

    // Inserció d'elements
    conjunt.add(10);
    conjunt.add(3);
    conjunt.add(8);
    conjunt.add(34);

    Iterator iterator = conjunt.iterator();
    while (iterator.hasNext()) {
        System.out.println(iterator.next()); // 34,3,8,10
    }
}
```

Activitat Prèvia

- **Activitat 6.** Crea un programa que demane a l'usuari noms de persones fins que introduïsca la **cadena "FIN"**. En acabar hauràs de mostrar tots els noms d'usuari (sense repeticions). Utilitza una estructura `HashSet`

Nota: Els noms de persona es sol·licitaran a una única línia separats per espais en blanc.

Exemple d'execució

```
Introdueix noms de persones separats per espais
Alex Pepe Juan Sergio
Maria Pepe Julia

Ana FIN Tomás

Julia
Alex
Ana
Joan
Sergio
Maria
Pepe
```

Activitat Prèvia

- **Activitat 7.** Donades les següents taules que **mostren el conjunt de participants** i i els resultats obtinguts en diferents proves de classificació per als jocs olímpics, crea una classe `Participant` i **2 llistes** (1 per a cada taula) que ens permeten representar la següent informació .

Prova 1		
Nom	**Dni**	**Temps**
Toni Garcia	64112243L	10,12
Elena Compte	72363370B	12,23
Maria Perez	56099532W	15,30
Juan Magan	89367935D	18,15

Prova 2		
Nom	**Dni**	**Temps**
Toni Garcia	64112243L	19,30
Ernesto Compte	39059576X	20,10
Mari Carmen Ruíz	26723726A	21,10
Elena García	87787367R	20,05

Utilitzant un `HashSet` crea un **nou llistat** que continga tots els **participants** que s'han presentat a **qualsevol de les 2 proves.** Investiga el mètode `addAll`.

Activitat Prèvia

Exemple d'Execució

```
 ==== Participants Prova 1 ====
nomComplet: Elena Compte, dni: 72363370B
nomComplet: Maria Perez, dni: 56099532W
nomComplet: Juan Magán, dni: 89367935D
nomComplet: Toni García, dni: 64112243L
Total Participants: 4

 ==== Participants Prova 2 ====
nomComplet: Mari Carmen Ruíz, dni: 26723726A
nomComplet: Elena García, dni: 87787367R
nomComplet: Ernesto Compte, dni: 39059576X
nomComplet: Toni García, dni: 64112243L
Total Participants: 4

 ==== Participants únics ====
nomComplet: Elena Compte, dni: 72363370B
nomComplet: Maria Perez, dni: 56099532W
nomComplet: Mari Carmen Ruíz, dni: 26723726A
nomComplet: Elena García, dni: 87787367R
nomComplet: Juan Magán, dni: 89367935D
nomComplet: Ernesto Compte, dni: 39059576X
nomComplet: Toni García, dni: 64112243L
Total Participants: 7
```

Activitat Formativa

- **Realitzar Activitat 8** (Parelles) - *Disponible a Aules*

- Això és tot... de moment :-)