UD8.1- ESTRUCTURES DE DADES DINÀMIQUES - Col·leccions (Llistes)

Programació – **1er DAW/DAM**

GENERALITAT VALENCIANA
Conselleria d'Educació, Cultura i Esport

CPIFP 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

- **List (ArrayList)**
  - Declaració i creació
  - Inserció, modificació i eliminació d'elements
  - Recorregut

1. Introducció

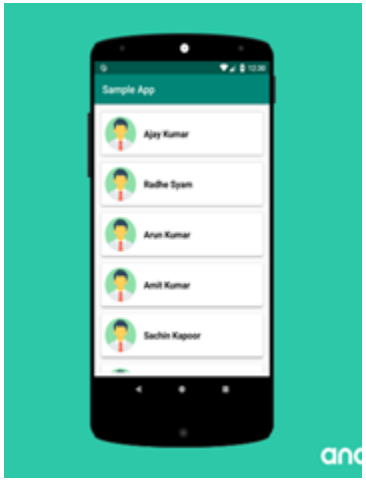
En tots els llenguatges de programació existeixen estructures per a gestionar **col·leccions** de dades. Podem diferenciar:

- Estructures **ESTÀTIQUES**
  - Hem de conèixer el nombre d'elements que formen part de la col·lecció en el moment de la compilació.
  - Exemple: arrays i matrius

```
/*
 * La llista següent conté 50 elements
 * durant tot el seu cicle de vida
 */
Persona[] llistat = new Persona [50];
```

1. Introducció

- Estructures **DINÀMIQUES**
  - El **número d'elements** es decideix i **modifica en temps d'execució**.
  - El número d'elements és **il·limitat** (o limitat a la quantitat de memòria disponible en la màquina)
  - Ex. Llistes, Piles, Coles, Llistes Enllaçades, etc.



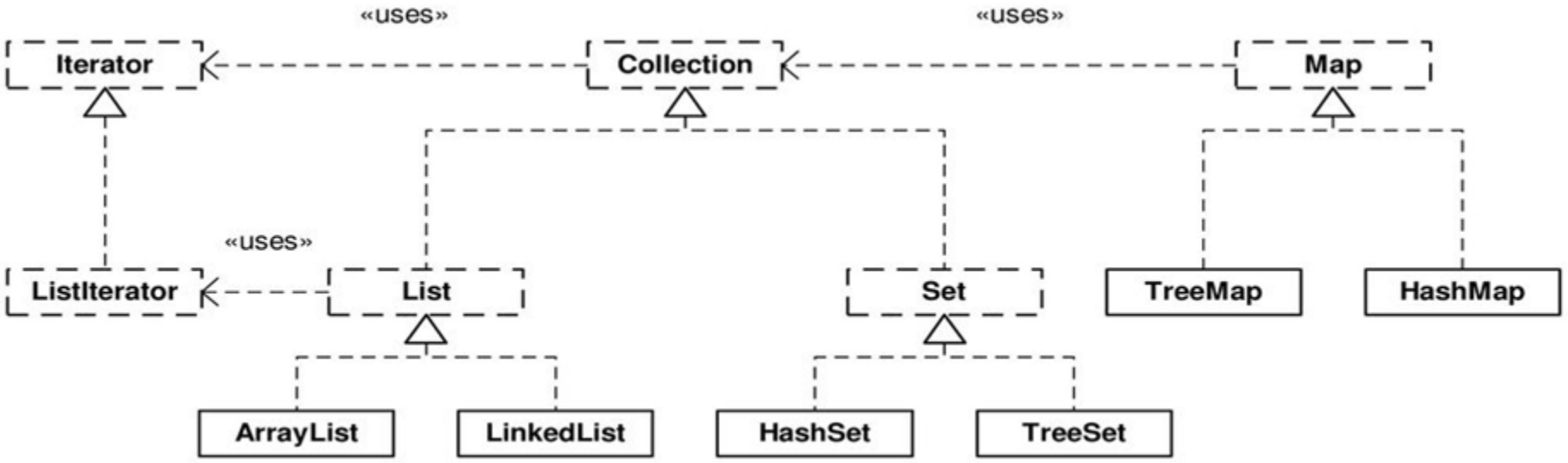
```
Persona juan = new Persona();

ArrayList<Persona> llistat = new ArrayList<>();

llistat.add(juan);
```

Quan es crea la **llista** té **0 elements**

La **llista** té **1 element**

2. Col·leccions en Java

- **Java proporciona una** àmplia **biblioteca** per treballar amb col·leccions de dades, disponibles al paquet `java.util`
  - Totes les **col·leccions** són **genèriques i estan** definides a partir de **interfícies.**
  - Representa un gran exemple de **reutilització** de codi**.**



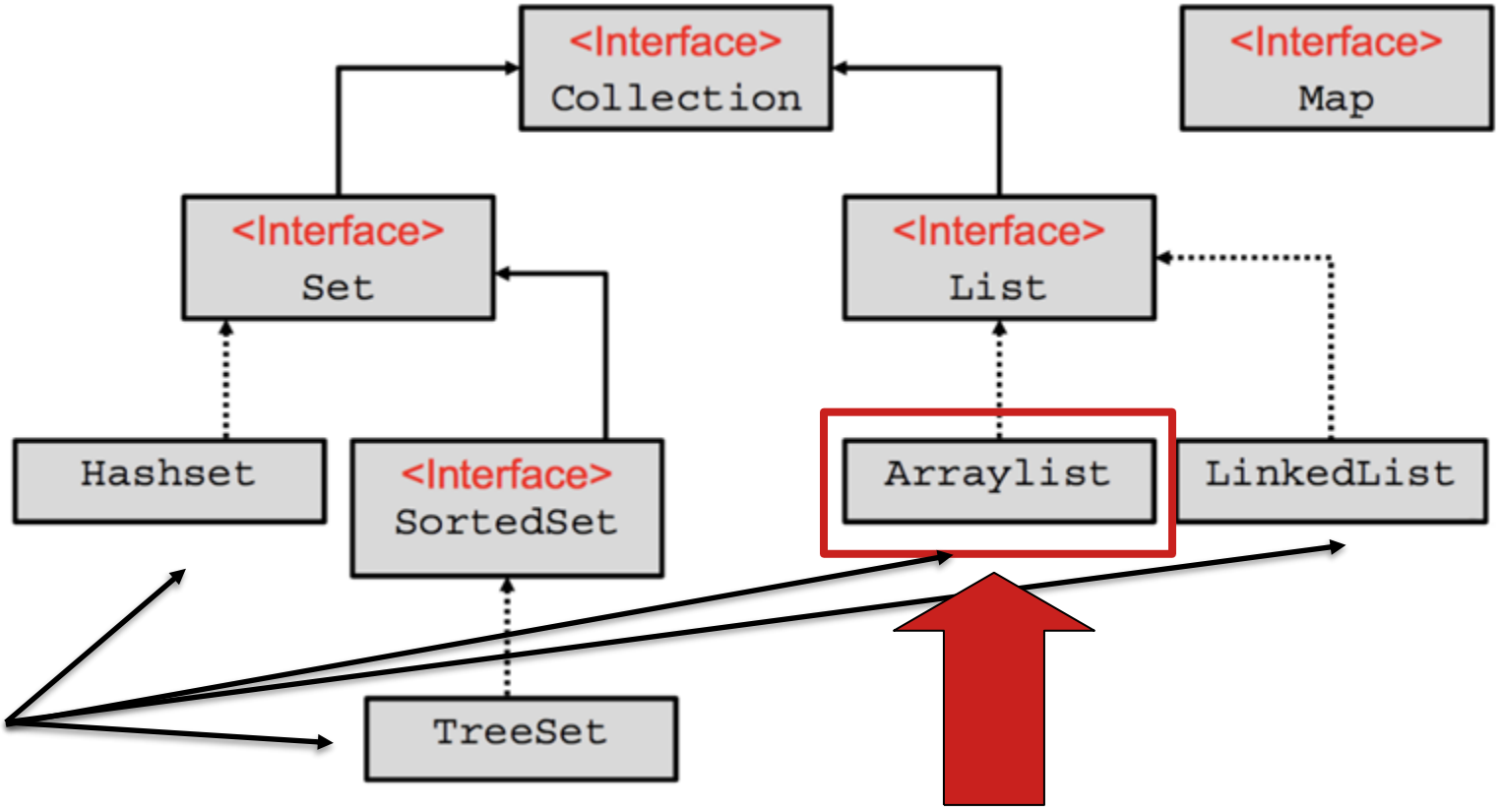
2.1 La interfície `Collection`

- La interfície `Collection` defineix operacions comunes a tots els tipus de col·leccions java.
  - El comportament vindrà donat pel tipus de col·lecció que fem servir. Podem diferenciar **2 tipus**:

List	Col·lecció d'objectes amb una **seqüència determinada**.
Set	Col·lecció d'objectes que no admet **duplicats**.

2.1 La interfície `Collection`

2.1.1 Implementacions (poden ser instanciades)



Farem servir la implementació **ArrayList** per exemplificar el seu funcionament

3. La classe *ArrayList*

- La classe `ArrayList` permet emmagatzemar dades en memòria, de forma similar als `arrays` però amb un gran avantatge: **la quantitat d'elements que pot contindre és dinàmic.**

- Hem de tenir en compte que la classe `ArrayList` s'implementa internament mitjançant un array **estàtic.**

  *- Quan el **array s'ompli**, es crea un nou array de major **longitud** i la informació antiga es transfereix a la nova matriu (similar al que hem fet amb* `Arrays.copyOf`*)*

```
public class ArrayList... {

   private Object[] elementData;
...
}
```

3.1 Declaració d'un `ArrayList`

- Un `ArrayList` es declara de la mateixa forma que qualsevol altra classe, excepte que hem d'especificar el tipus de dates que gestionarà la llista.

```
ArrayList<Tipus_base> llistat  = new ArrayList<>();
```

- Si volem mantenir un **tipus primitiu** (`char`, `int`, `double`, …), haurem d'especificar la classe **envoltori** associada: `Character`, `Integer`, `Double`, `Float`, *etc.*

- **Exemple:**

```
ArrayList<String> paisatges;
ArrayList<Integer> edats;
```

3.2 Creació d'un objecte *ArrayList*

```
ArrayList<NomClasse> nomLlista;
nomLlista = new ArrayList<>();
```

Declaració e inicialització en **2 passos**

```
ArrayList<NomClasse> nomLlista = new ArrayList<>();
```

Declaració e inicialització

Exemples:

```
ArrayList<String> paisos = new ArrayList<>();
ArrayList<Alumne> curs = new ArrayList<>();
```

3.2 Creació d'un objecte *ArrayList*

- Disposa de **3 constructors**:
  - `ArrayList()`: Constructor per defecte. Crea una llista buida.
  - `ArrayList(int capacitatInicial)`: Crear una llista amb una capacitat inicial.
  - `ArrayList(Collection c)`: Crea una llista d'elements a partir d'un altra col·lecció.

```
HashSet<Integer> puntuacions = new HashSet<>();

puntuacions.add(12);
puntuacions.add(24);

ArrayList<Integer> nomLlista = new ArrayList<>(puntuacions);
```

3.3 Afegir elements al final de la llista

- `boolean add(Object elementAInsertar)`
  - Permet **inserir** nous elements al *ArrayList*.
  - El primer element es col·loca a la posició 0, el segon a la posició 1, etc.
  - El boolean retornat sempre serà true.

Exemples:

```
ArrayList<String> paisos = new ArrayList<>();
paisos.add("Espanya");  // Ocupa la posició 0
paisos.add("França");   // Ocupa la posició 1
paisos.add("Portugal"); // Ocupa la posició 2

// Es pot crear un ArrayList per desar dades numèriques
ArrayList<Integer> edats =new ArrayList<>();
edats.add(22);
edats.add(31);
edats.add(18);
```

3.4 Afegir elements a una posició determinada

- `void add(int posició, Object elementAInserir)`
  - **Insereix** l'element en la **posició indicada** movent tots els elements una posició cap a la dreta.
  - Si voleu inserir en una posició que **no existeix**, produirà l'excepció `IndexOutOfBoundsException`.

Exemple:

```
ArrayList<String> paisos = new ArrayList<>();
paisos.add("Espanya");
paisos.add("França");
paisos.add("Portugal");
// El orden fins ara és: Espanya, França, Portugal
paisos.add(1,"Itàlia");
// La comanda ara és: Espanya, Itàlia, França, Portugal
```

3.5 Consulta d'un element d'una llista

- `Object get(int index)`
  - Permet **obtenir** l'element d'un índex concret.

Exemple:

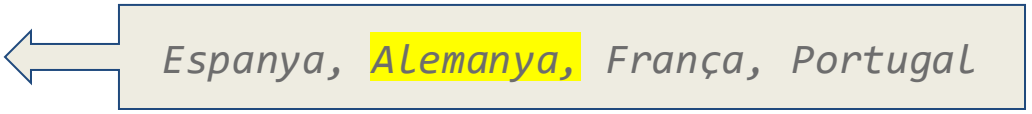
```
/**
 *  Seguint amb l'exemple anterior on tenim:
 *   Espanya, Itàlia, França i Portugal
 *  mostra: Portugal
 **/
System.out.println(paisos.get(3));
```

3.6 Modificar un element de la llista

- `Object set(int index, Object nouElement)`
  - Permet **modificar** (substituir) un element que prèviament ha sigut guardat a la llista.
    - El primer paràmetre indica el índex de la posició que ocupa l'element a modificar.
    - El segon paràmetre indica el nou element que es guardarà a la posició indicada.
    - Retorna una referència a l'objecte substituït

Exemple:

```
paisos.set (1,"Alemanya");
```



3.7 Cercar un element

- La classe `ArrayList` ens proporciona mètodes per a saber si un element está en el llistat o no.

```
public boolean contains(Object object)
```

```
ArrayList<String> paisos = new ArrayList<>();
paisos.add("Espanya");
paisos.add("França");
paisos.add("Portugal");
if (paisos.contains("Espanya")){
   System.out.println("El element es troba en l'array");
}
```

3.7 Cercar un element

- Quan realitzem la cerca de l'element en el `ArrayList`, internament s'utilitza el mètode `equals()` de la classe `Object`

```
public class Pelicula {
  ...
  public boolean equals(Object object) {
      if (!(obj instance of Pelicula)) {
          return false;
      }
      Pelicula otraPeli = (Pelicula) obj;
      return this.getAutor().equalsIgnoreCase(otraPeli.getAutor())
                    && getTitle().equalsIgnoreCase(otraPeli.getTitle());
   }
   ...
}
```

3.8 Recòrrer una llista

- `public int size()`
  - Retorna el **número d'elements** de la llista.

Exemple:

```
for (int i = 0; i < paisos.size() ; i++) {
   System.out.println(paisos.get(i));
}
```

```
for (String pais: paisos) {    // Ací no cal fer ús de size()
   System.out.println(pais);
}
```

3.9 Altres mètodes de classe `ArrayList`

- `boolean remove(Object o)`
  - **Eliminar** de la col·lecció l'objecte indicat.
- `void clear()`
  - **Esborra** tot el contingut de la llista.
- `Object clone()`
  - Retorna una **còpia** de la llista.
- `int indexOf(Object o)`
  - Retorna **l'índex** de la **primera ocurrència** de l'element especificat en la llista (Internament fa ús del mètode `equals` definit a la classe `Object`).
- `boolean isEmpty()`
  - Torna true **si** la llista **està buida**.
- `Object[] toArray()`
  - **Converteix** la llista a un **array estàtic**.

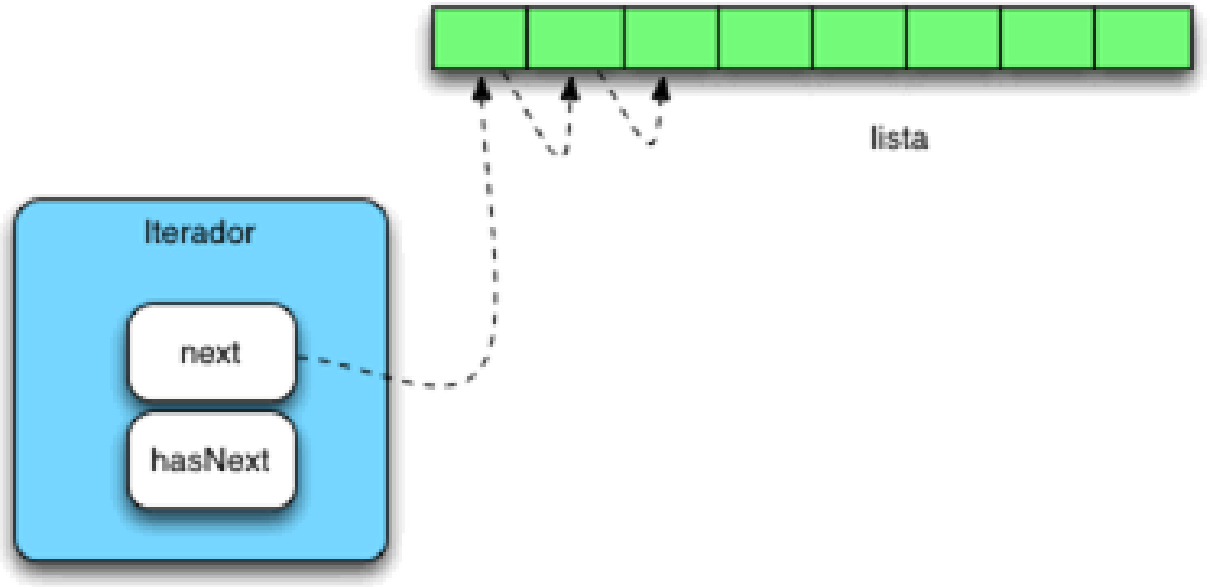
Activitat prèvia

- **Activitat 1.-** Escriu un programa en Java que declare e inicialitze un `ArrayList` amb els **colors de l'arc de Sant Martí**. Seguidament heu d'indicar per pantalla:
  - El **nombre d'elements** que conté el llistat.
  - La posició que ocupa el **color vermell.**
  - **Recorre** la llista i mostra tots els elements.
  - Crea un mètode que demane a l'usuari un color i indique si aquest color es troba a la llista; en cas contrari haureu d'afegir aquest color al llistat.

3.10 Recòrrer una llista II

- Interfície `Iterator`
  - Totes les col·leccions implementen la interfície `Iterable`
  - `Iterable` utilitza un `Iterator` internament.
  - Aquesta interfície ens proporciona la capacitat de **iterar** sobre els **elements** de la **col·lecció**.
  - Amb el mètode `iterator()` de qualsevol col·lecció podem obtindre el iterador associat que ens permetrà utilitzar els métodes:
    - `hasNext()`: ens indica si hi ha més elements a la llista
    - `next()`: mou l'iterador al següent element de la llista

3.10 Recòrrer una llista II




```
Iterator iterator = paisos.iterator();
// True si hi ha més elements
while (iterator.hasNext ()) {
   // Accés a l'element i apunta al següent
   System.out.println(iterator.next());
}
```

3.10 Recòrrer una llista II

Exemple:

```
// ArrayList
List<String> llista2 =new ArrayList<>();

// Afegim nodes i creem un Iterator
llista2.add("Madrid");
llista2.add("Sevilla");
llista2.add("València");
Iterator iterador2 = llista2.iterator();

// Recorrem i mostrem la llista
while(iterador2.hasNext()) {
    // Hem de fer un càsting al tipus d'element
    String element = (String) iterador2.next();
    System.out.print(element +" ");
}
System.out.println("--ArrayList--");
```

Activitat Prèvia

- **Activitat 2.** Crea un `ArrayList` amb els noms de 6 companys de classe. A continuació, mostra aquests noms per pantalla. Utilitza per fer-ho un **Iterator**.
  - Tot seguit escriu un programa que pregunte a l'usuari per un nom i indique si aquesta persona és company de classe o no. Utilitza per això el mètode `contains()`.

Activitat Prèvia

- **Activitat 3.-** Realitza un programa que introduïsca 50 **valors aleatoris** (entre 0 i 100) en un `ArrayList` i que després calcule la *suma*, la *mitjana*, el *màxim* i el *mínim* d 'aquests números.

Activitat Prèvia

Activitat 4.- Crea una classe anomenada `VideoJoc`. Per a cada videojoc, emmagatzemarem el seu títol, gènere, preu i si és multijugador o no.

- Seguidament crea una clase `TestVideoJoc` on declares un `ArrayList` amb el següent llistat de videojocs i seguidament escriu un programa que pregunte a l'usuari pel nom d'un videojoc que vullga llogar e indiqueu per pantalla si se'n disposa o no d'ell.
- **Nota. Hauràs de sobreescriure el mètode `equals`.**

Título	Género	Precio	Multijugador
Fortnite	Acción	40€	Si
Fifa	Deportes	50€	SI
Gran Theft Auto	Acción	80€	Si
MineCraft	Simulación	60€	Si
AnimalCrossing	Simulación	30€	Si

Activitat Formativa

- **Realitzar Activitat 5** (Parelles) *Disponible a aules*