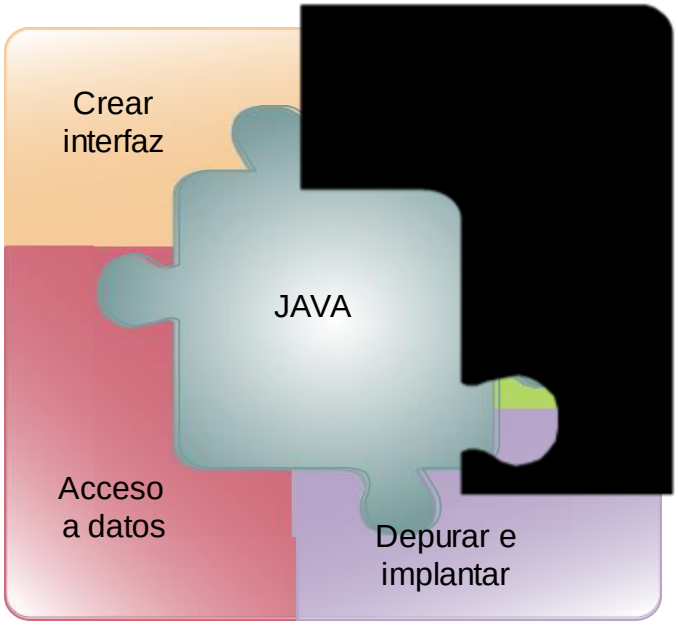
Unidad 6-Extra
ArrayLists

Descripción



- **Estructuras Dinámicas vs Estructuras Estáticas**
- **Colecciones**
- **ArrayList**
  - Declaración y creación
  - Añadir
  - Consultar
  - Modificar
  - Buscar
  - Recorrer
  - Otros elementos
- **Array vs ArrayList**

Estructuras Dinámicas –Estructuras Estáticas

En prácticamente todos los lenguajes de computación existen estructuras para almacenar colecciones de datos:

Una serie de datos agrupados a los que se puede hacer referencia con un único nombre

- **Estructuras estáticas**
  - *Se* debe saber el número de elementos que formarán parte de esa colección en tiempo de compilación
  - Un ejemplo son los arrays
- **Estructuras dinámicas**
  - El número de elementos integrantes se decide y modifica en tiempo de ejecución
  - El número de elementos es ilimitado
  - Son ejemplos las colas, pilas, listas enlazadas, árboles, grafos, etc

ARRAYLIST

ArrayList

- **La clase ArrayList permite el almacenamiento de datos en memoria de forma similar a los *arrays* convencionales pero con la gran ventaja de que el número de elementos que puede almacenar es dinámico**
  - La cantidad de elementos que puede almacenar un *array* convencional está limitado por el número que se indica en el momento de crearlo o inicializarlo
  - Los ArrayList pueden almacenar un número variable de elementos sin estar limitados por un número prefijado
- Está pensada para crear **listas en las cuales se aumenta el final de la lista frecuentemente**
- **Forma parte del paquete java.util.ArrayList**

Declaración de una variable ArrayList

- **De forma genérica:**

```
ArrayList nombreDeLista;
```

  - De esta manera no se especifica el tipo de datos

    Esto permite listas heterogéneas a cambio de ser necesario el uso de *casting en la* recuperación de los elementos

  - Es recomendable especificar el tipo de datos que va a contener la lista

    Así se emplearán las operaciones y métodos adecuados para el tipo de datos manejado

- **Para especificar el tipo de datos:**

```
ArrayList<nombreClase> nombreDeLista;
```

  - **En caso de almacenar datos de un tipo básico de Java (char, int, double, etc…) se debe especificar el nombre de la clase asociada (Wrapper class): Character, Integer, Double, etc…**

Declaración de una variable ArrayList

- **Para especificar el tipo de datos ejemplos:**

```
ArrayList<String> listaPaises;
ArrayList<Integer> edades;
```

Primitive Data Type	Wrapper Class
char	Character
byte	Byte
short	Short
int	Integer
long	Long
float	Float
double	Double
boolean	Boolean

Creación de un objeto ArrayList

```
ArrayList<nombreClase> nombreDeLista;
nombreDeLista = new ArrayList( );
```

- **Se puede declarar la lista a la vez que se crea:**

```
ArrayList<nombreClase> nombreDeLista = new ArrayList( )
```

- **Ejemplo:**

```
ArrayList<String> listaPaises = new ArrayList(  );

ArrayList<Alumnos>   curso6J = new ArrayList(  );
```

Creación de un objeto ArrayList

- **Posee tres constructores:**
  - ArrayList( )

    Constructor por defecto. Crea un ArrayList vacío
  - ArrayList(int capacidadInicial)

    Crea una lista con una capacidad inicial indicada
  - ArrayList(Collection c)

    Crea una lista a partir de los elementos de la colección indicada

Añadir elementos al final de la lista

- Método: **boolean add(*Object elementoAInsertar*)**
  - Los elementos que se van añadiendo se colocan después del último elemento
  - El primer elemento que se añada se colocará en la posición 0
- **Ejemplos:**

```
ArrayList<String> listaPaises = new ArrayList();
listaPaises.add("España");   //Ocupa la posición 0
listaPaises.add("Francia");  //Ocupa la posición 1
listaPaises.add("Portugal"); //Ocupa la posición 2

//Se pueden crear ArrayList para guardar datos numéricos
ArrayList<Integer> edades = new ArrayList();
edades.add(22);
edades.add(31);
edades.add(18);
```

Insertar elementos en una determinada posición

- Método: **void add(int *posición*, Object *elementoAInsertar*)**
  - inserta un elemento en una determinada posición desplazando el elemento que se encontraba en esa posición y todos los siguientes, una posición más
  - Si se intenta insertar en una **posición que no existe** se produce una excepción (IndexOutOfBoundsException)
- **Ejemplo:**

```
ArrayList<String> listaPaises = new ArrayList();
listaPaises.add("España");
listaPaises.add("Francia");
listaPaises.add("Portugal");
//El orden hasta ahora es: España, Francia, Portugal
listaPaises.add(1, "Italia");
//El orden ahora es: España, Italia, Francia, Portugal
```

Consultar un elemento de la lista

- Método: **Object get (int *posición*)**
  - permite obtener el elemento almacenado en una determinada posición
- **Ejemplo**

```
System.out.println(listaPaises.get(3));
//Siguiendo el ejemplo anterior, mostraría: Portugal
```

Modificar un elemento de la lista

- Método:

  Object set (int *posición*, Object *nuevoElemento*)

  - Permite modificar un elemento que previamente ha sido almacenando en la lista

    El primer parámetro indica la posición que ocupa el elemento a modificar

    El segundo parámetro especifica el nuevo elemento que ocupará dicha posición sustituyendo al elemento anterior

```
listaPaises.set(1, "Alemania");
```

Buscar un elemento

- Método: **int indexOf(Object *elementoBuscado*)**
  - Retorna la posición que ocupa el elemento que se indique por parámetro
  - Si el elemento se encuentra en más de una posición retorna la posición del primero
  - El método lastIndexOf obtiene la posición del último encontrado
- **Ejemplo**

```
String paisBuscado = "Francia";
int pos = listaPaises.indexOf(paisBuscado);
if(pos!=-1)
   System.out.println(paisBuscado + “  en la posición: "+pos);
else
   System.out.println(paisBuscado + " no se ha encontrado");
```

Recorrer el contenido de la lista

- **size( )**
  - Devuelve el número de elementos que contiene la lista

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

```
for(String pais:listaPaises)
     System.out.println(pais);
```

Otros métodos

```
// Declaración de un ArrayList de "String". Puede ser de cualquier otro Elemento u Objeto (float, Boolean, Object, ...)

ArrayList<String> nombreArrayList = new ArrayList<String>();

// Añade el elemento al ArrayList

nombreArrayList.add("Elemento");

// Añade el elemento al ArrayList en la posición 'n'

nombreArrayList.add(n, "Elemento 2");

// Devuelve el numero de elementos del ArrayList

nombreArrayList.size();

// Devuelve el elemento que esta en la posición '2' del ArrayList

nombreArrayList.get(2);

// Comprueba se existe del elemento ('Elemento') que se le pasa como parametro

nombreArrayList.contains("Elemento");

// Devuelve la posición de la primera ocurrencia ('Elemento') en el ArrayList

nombreArrayList.indexOf("Elemento");
```

Otros métodos

```
// Devuelve la posición de la última ocurrencia ('Elemento') en el ArrayList
nombreArrayList.lastIndexOf("Elemento");
// Borra el elemento de la posición '5' del ArrayList
nombreArrayList.remove(5);
// Borra la primera ocurrencia del 'Elemento' que se le pasa como parametro.
nombreArrayList.remove("Elemento");
//Borra todos los elementos de ArrayList
nombreArrayList.clear();
// Devuelve True si el ArrayList esta vacio. Sino Devuelve False
nombreArrayList.isEmpty();
// Copiar un ArrayList
ArrayList arrayListCopia = (ArrayList) nombreArrayList.clone();
// Pasa el ArrayList a un Array
Object[] array = nombreArrayList.toArray();
```

Arrays Vs. ArrayList

arrays	*List*
`String[] x;`	`ArrayList<String> x;`
`x = new String[1000];`	`x = new ArrayList<String>();`
`... = x[20];`	`... = x.get(20);`
`x[20] = "1492";`	`x.set(20, "1492");`
	`x.add("2001");`