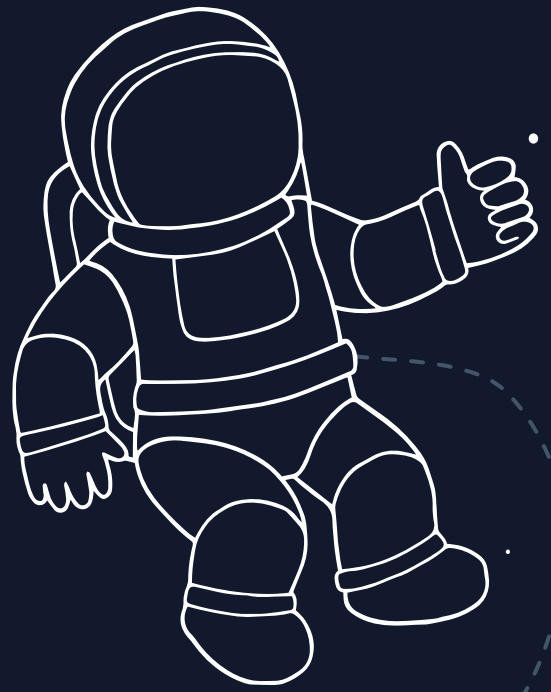
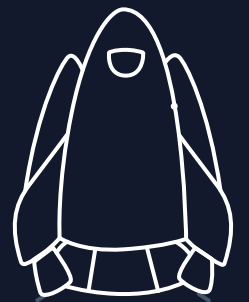


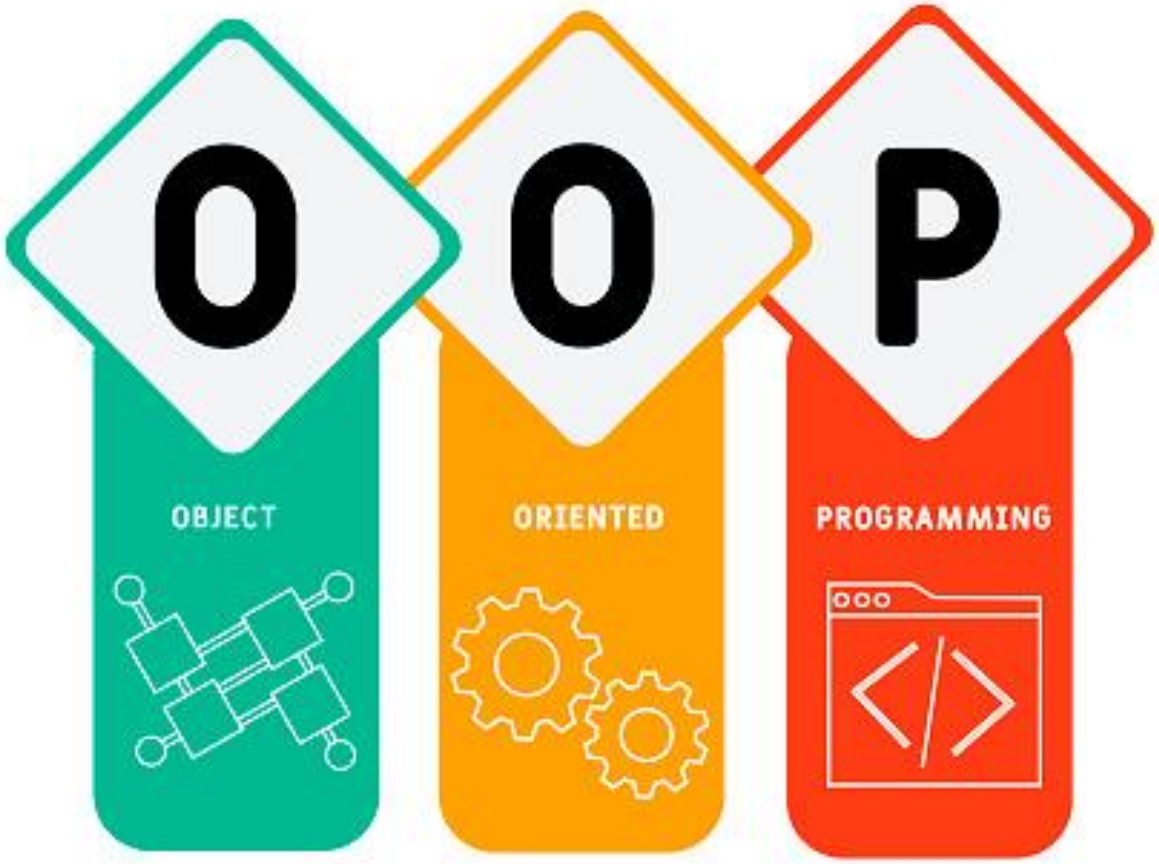
Programa académico CAMPUS

MODULO JAVA
Sesión 8
Arreglos, Listas y Clases genéricas

Trainer Carlos H. Rueda C.



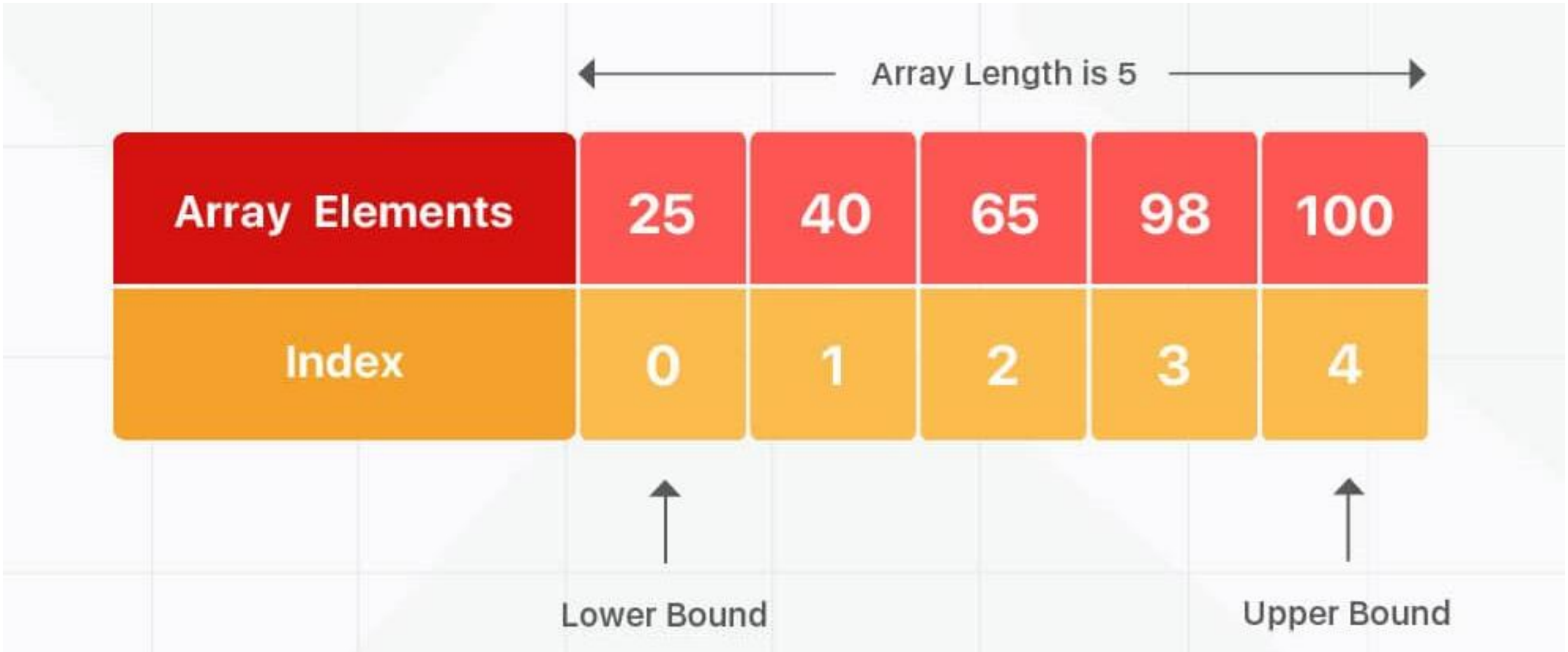

Java™

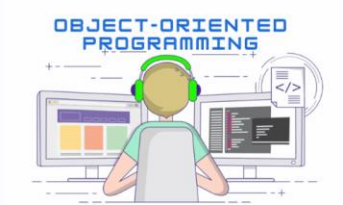

O
O
P
OBJECT
ORIENTED
PROGRAMMING

Arreglos

- Es una estructura de datos
- Posibilita la agrupación de elementos del **mismo tipo** en una sola variable.
- Se organizan en memoria de forma secuencial. Empiezan en 0
- Los arreglos son considerados objetos en Java
- Una vez creados, tienen un **tamaño fijo**.
- El tamaño del arreglo se establece durante la declaración **no puede modificarse durante la ejecución del programa**

Arreglos


Array Length is 5
Array Elements
25
40
65
98
100
Index
0
1
2
3
4
Lower Bound
Upper Bound


OBJECT-ORIENTED
PROGRAMMING

Arreglos

Sintaxis

Para **declarar** un arreglo en Java, se utiliza la siguiente sintaxis:

```
tipoDeDatos[] nombreDelArreglo;
```

Por ejemplo, para declarar un arreglo de enteros:

```
int[] numeros;
```

Arreglos

Sintaxis

Después de **declarar** un arreglo, se necesita **inicializarlo** para asignar memoria y especificar el tamaño del arreglo (El tamaño de los arreglos se declara en un primer momento y **no puede cambiar** en tiempo de ejecución como puede producirse en otros lenguajes).

```
nombreDelArreglo = new tipoDeDatos[tamaño];
```

Ejemplo:

```
numeros = new int[5];
```

En una sola línea

```
int[] numeros = new int[5];
```

Asignación de valores a un arreglo

Se pueden asignar valores a los elementos del arreglo utilizando el índice del arreglo.

En la declaración:

```
int[] numeros = {10, 20, 30, 40, 50};
int numeros2[] = {10, 20, 30, 40, 50};
int[] numeros3 = new int[] {2, 4, 5, 8};
```

Asignar valores:

```
int[] numeros = new int[5];

numeros[0] = 10;
numeros[1] = 20;
numeros[2] = 30;
numeros[3] = 40;
numeros[4] = 50;
```

Acceso a elementos del arreglo

Se puede acceder a los elementos del arreglo utilizando sus índices. Por ejemplo, para imprimir el primer elemento del arreglo:

```
System.out.println(numeros[0]); // Esto imprimirá 10
```

Obtener el tamaño del arreglo

Se puede acceder al tamaño del arreglo por medio del atributo `length`, Este atributo devuelve el número de elementos que posee el array. Hay que tener en cuenta que es una variable de solo lectura, es por ello que no es posible realizar una asignación a dicho atributo.

```
System.out.println(numeros.length); // Esto imprimirá 5
```

Iteración de un arreglo

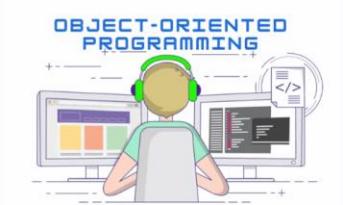
Se puede utilizar estructuras repetitivas para iterar a través de los elementos de un arreglo.

```
for (int i = 0; i < numeros.length; i++) {
    System.out.println(numeros[i]);
}
```

Iteración de un arreglo

Otra alternativa es la estructura **for-each** la cual, en lugar de trabajar con índices, procesa directamente cada elemento del array. Sin embargo, ***no se tiene acceso al índice del elemento actual***.

```
for (int numero : numeros) {
    System.out.println("Elemento: " + numero);
}
```



Arreglos multidimensionales

Java también permite crear arreglos multidimensionales. Por ejemplo, una matriz bidimensional se puede declarar e inicializar de la siguiente manera:

```java
// Asignación de valores en la declaración
int[][] matriz = {{1, 2}, {3, 4}, {5, 6}};

// int[][] matriz = new int[3][2];
// Asignación de valores
matriz[0][0] = 1;
matriz[0][1] = 2;
matriz[1][0] = 3;
matriz[1][1] = 4;
matriz[2][0] = 5;
matriz[2][1] = 6;
```

toString()

El método `toString()` convierte un arreglo en una representación de cadena legible. Este método es útil cuando se requiere imprimir o mostrar el contenido de un array.

```
System.out.println(Arrays.toString(numeros));
// Esto imprimirá [10, 20, 30, 40, 50]
```

Método sort()

El método `sort()` ordena los elementos de un arreglo en orden ascendente (siempre y cuando los elementos se puedan ordenar).

```
import java.util.Arrays;

class Main {
    public static void main(String[] args) {
        String[] names = {"Oscar", "America", "Ana"};

        Arrays.sort(names);

        for (String name : names)
            System.out.println(name);
    }
}
```

Método sort()

```java
import java.util.Arrays;
import java.util.Comparator;

class Main {
    public static void main(String[] args) {
        String[] names = {"Oscar", "America", "Ana"};

        // Ordenar por longitud de caracteres
        Arrays.sort(names, new Comparator<String>() {
            @Override
            public int compare(String o1, String o2) {
                //<0: menor || 0: igual || >0: mayor
                return o1.length() - o2.length();
            }
        });

        for (String name : names)
            System.out.println(name);
    }
}
```

Método binarySearch()

```
String[] names = {"Oscar", "America", "Ana"};

int pos = Arrays.binarySearch(names, "Ana");
if (pos >= 0)
    System.out.println("Se encontró. Posición: " + pos);
else
    System.out.println("No se encontró en el array.");
```

Método equals()

Este método devuelve ***verdadero*** si ambos arrays contienen los mismos elementos en el mismo orden.

```
String[] names = {"Oscar", "America", "Ana"};
String[] names2 = {"Oscar", "Ana", "America"};

System.out.println("Comparando Arrays : " + Arrays.equals
    (names, names2));
```

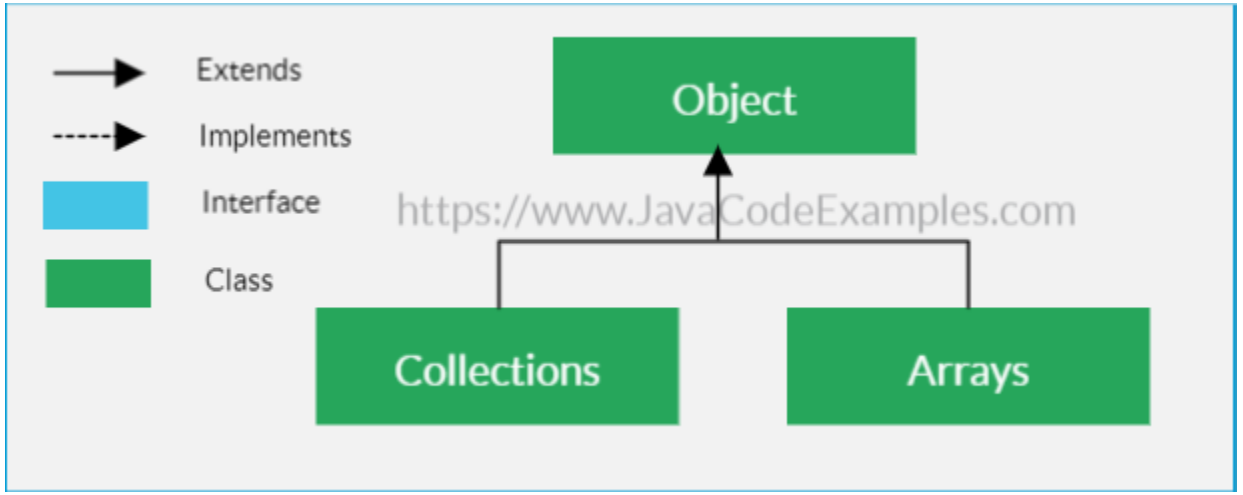
Método fill()

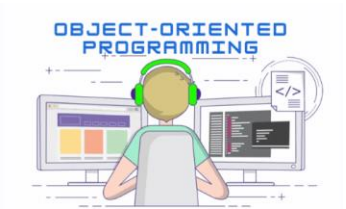
Este método asigna el valor especificado a cada elemento del rango especificado

```
int[] numeros = new int[10];

Arrays.fill(numeros, 1);

for (int elem : numeros)
  System.out.print(elem + " ");
```

Colecciones en Java

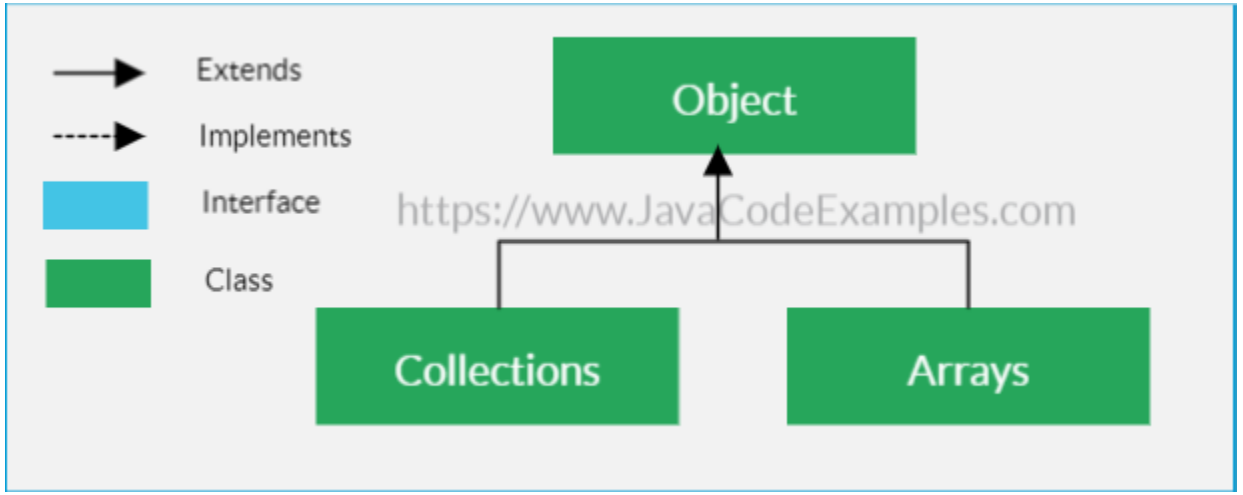
Cualquier conjunto de objetos individuales que se representan como una única entidad se conoce como una colección de objetos. En Java, se ha definido un marco separado llamado el "***Framework de Colecciones***" en JDK 1.2, que contiene todas las clases e interfaces de colección en él. En Java, la interfaz `Collection` (`java.util.Collection`) y la interfaz `Map` (`java.util.Map`) son las dos principales interfaces "raíz" de las clases de colección en Java.

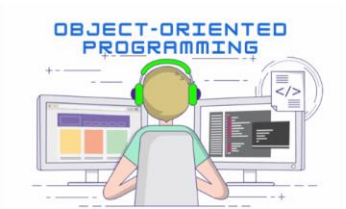




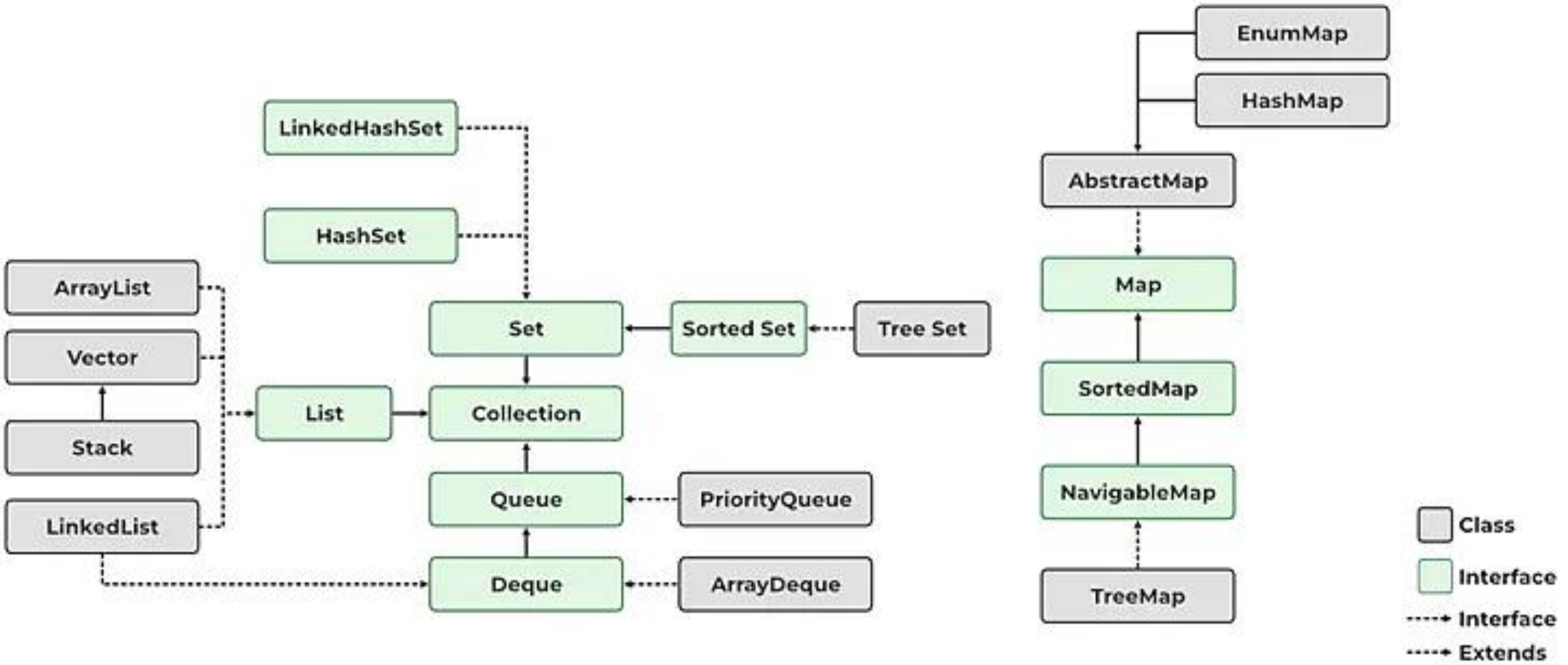
Colecciones en Java

Cualquier conjunto de objetos individuales que se representan como una única entidad se conoce como una colección de objetos. En Java, se ha definido un marco separado llamado el "***Framework de Colecciones***" en JDK 1.2, que contiene todas las clases e interfaces de colección en él. En Java, la interfaz `Collection` (`java.util.Collection`) y la interfaz `Map` (`java.util.Map`) son las dos principales interfaces "raíz" de las clases de colección en Java.



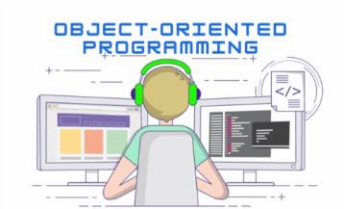


Jerarquía del framework de colecciones



Métodos de la Interfaz Collection

- **boolean add(Object obj)**
  - Adds the **obj** to the invoking collection.
- **boolean addAll(Collection c)**
  - Adds all the elements of **c** to the invoking collection.
- **boolean remove(Object obj)**
  - Removes the **obj** from the invoking collection.
- **boolean removeAll(Collection c)**
  - Removes all the elements of **c** from the invoking collection.
- **boolean retainAll(Collection c)**
  - Removes all the elements from the invoking collection except elements in **c**.
- **void clear( )**
  - Removes all elements from the invoking collection.
- **boolean contains(Object obj)**
  - Finds the **obj** to the invoking collection.
- **boolean containsAll(Collection c)**
  - Finds all elements of **c** in the invoking collection.
- **boolean equals(Object obj)**
  - Returns true if the **obj** and the invoking collection are equal, else returns false.
- **boolean isEmpty( )**
  - Returns true if the invoking collection is empty, else returns false.
- **int size( )**
  - Returns total number of elements in the invoking collection.
- **int hashCode( )**
  - Returns hash code for the invoking collection.
- **Object[ ] toArray( )**
  - Returns an array that contains all elements of the invoking collection.
- **Object[ ] toArray(Object obj)**
  - Returns an array that contains only those elements whose type matches the array type.
- **Iterator iterator( )**
  - Returns an iterator for the invoking collection.



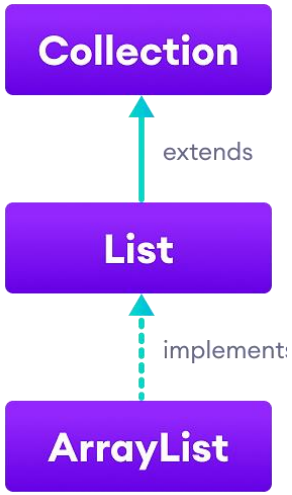
List

La interfaz `List` en Java proporciona una manera de almacenar una colección ordenada. Es una interfaz secundaria de `Collection`. Consiste en una colección ordenada de objetos en la que **se pueden almacenar valores duplicados**. Dado que `List` mantiene el orden de inserción, permite el acceso posicional y la inserción de elementos.

La interfaz `List` se encuentra en el paquete `java.util` y hereda de la interfaz `Collection`. Además, es una fábrica de la interfaz `ListIterator`. A través del `ListIterator`, se puede recorrer la lista en direcciones hacia adelante y hacia atrás. Las clases de implementación de la interfaz `List` son `ArrayList`, `LinkedList`, `Stack` y `Vector`. `ArrayList` y `LinkedList` son ampliamente utilizadas en programación Java. *Desde Java 5, la clase* `Vector` *está obsoleta.*

ArrayList

`ArrayList` en Java es una parte del marco de trabajo de colecciones de Java y es una clase del paquete `java.util` y proporciona arreglos dinámicos en Java. Aunque puede ser más lento que los arreglos estándar, es útil en programas donde se necesita mucha manipulación en el arreglo. Las principales ventajas de `ArrayList` en Java son que si se declara un arreglo, es necesario mencionar su tamaño, pero en `ArrayList`, no es necesario especificar el tamaño.



Métodos de ArrayList – add()

```java
ArrayList<String> strArrayList = new ArrayList<String>();

// Añade el elemento al ArrayList
strArrayList.add("Elemento 1");
strArrayList.add("Elemento 2");

// Añade el elemento al ArrayList en la posición '1'
strArrayList.add(1, "Elemento 3");

for(String elem : strArrayList) {
    System.out.println(elem);
}
```

Métodos de ArrayList – size()

```
System.out.println("Tamaño ArrayList: " + strArrayList.size());
```

Métodos de ArrayList – get()

```
// Devuelve el elemento que esta en la posición '2' del
    ArrayList
System.out.println("Elemento en posicion 2: " +strArrayList.get
    (2));
```

Métodos de ArrayList – get()

```
// Comprueba se existe del elemento ('Elemento') que se le pasa
    como parametro
System.out.println("Está 'Elemento 4' en el Array?: "
    +strArrayList.contains("Elemento 4"));
```

Métodos de ArrayList – indexOf()

```
// Devuelve la posición de la primera ocurrencia ('Elemento') en
    el ArrayList
System.out.println("Posición de 'Elemento 4' en el Array?: " +
    strArrayList.indexOf("Elemento"));
```

Métodos de ArrayList – lastIndexOf()

```
// Devuelve la posición de la última ocurrencia ('Elemento') en
    el ArrayList
System.out.println("Posición de 'Elemento 2' en el Array?: " +
    strArrayList.lastIndexOf("Elemento 2"));
```

Métodos de ArrayList – remove()

```
// Borra el elemento de la posición '5' del ArrayList
    // Devuelve el elemento eliminado
System.out.println(strArrayList.remove(0));

// Borra la primera ocurrencia del 'Elemento' que se le pasa
    como parametro.
// Devuelve verdadero si esta lista contiene el elemento
    especificado
System.out.println(strArrayList.remove("Elemento 4"));
```

Métodos de ArrayList – clear() y isEmpty()

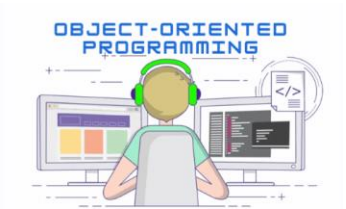
```
//Borra todos los elementos de ArrayList
strArrayList.clear();

// Devuelve true si esta lista no contiene elementos
strArrayList.isEmpty();
```

Métodos de ArrayList – clone() y toArray()

```java
// Copiar un ArrayList
ArrayList strArrayListCopia = (ArrayList) strArrayList.clone();

// Pasa el ArrayList a un Array
Object[] vstr = strArrayList.toArray();
for(Object elem : vstr) {
    System.out.println((String) elem);
}
```

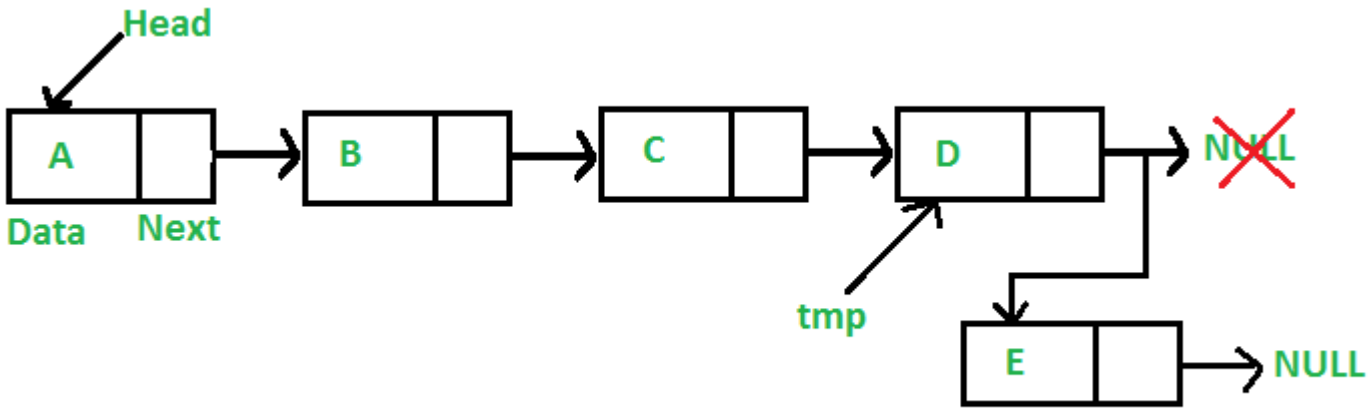
Recorre de ArrayList

```java
ArrayList<Integer> miLista = new  ArrayList<>();


//Recorrer con for utilizando los índices
for (int i = 0; i < miLista.size(); i++) {
  System.out.println(miLista.get(i));
}

//Recorrer con foreach a través de los elementos
for (Integer elemento : miLista) {
  System.out.println(elemento);
}

//Recorrer utilizando el iterador de la lista
Iterator iterador = miLista.iterator();
while(iterador.hasNext()){
  System.out.println(iterador.next());
}
```

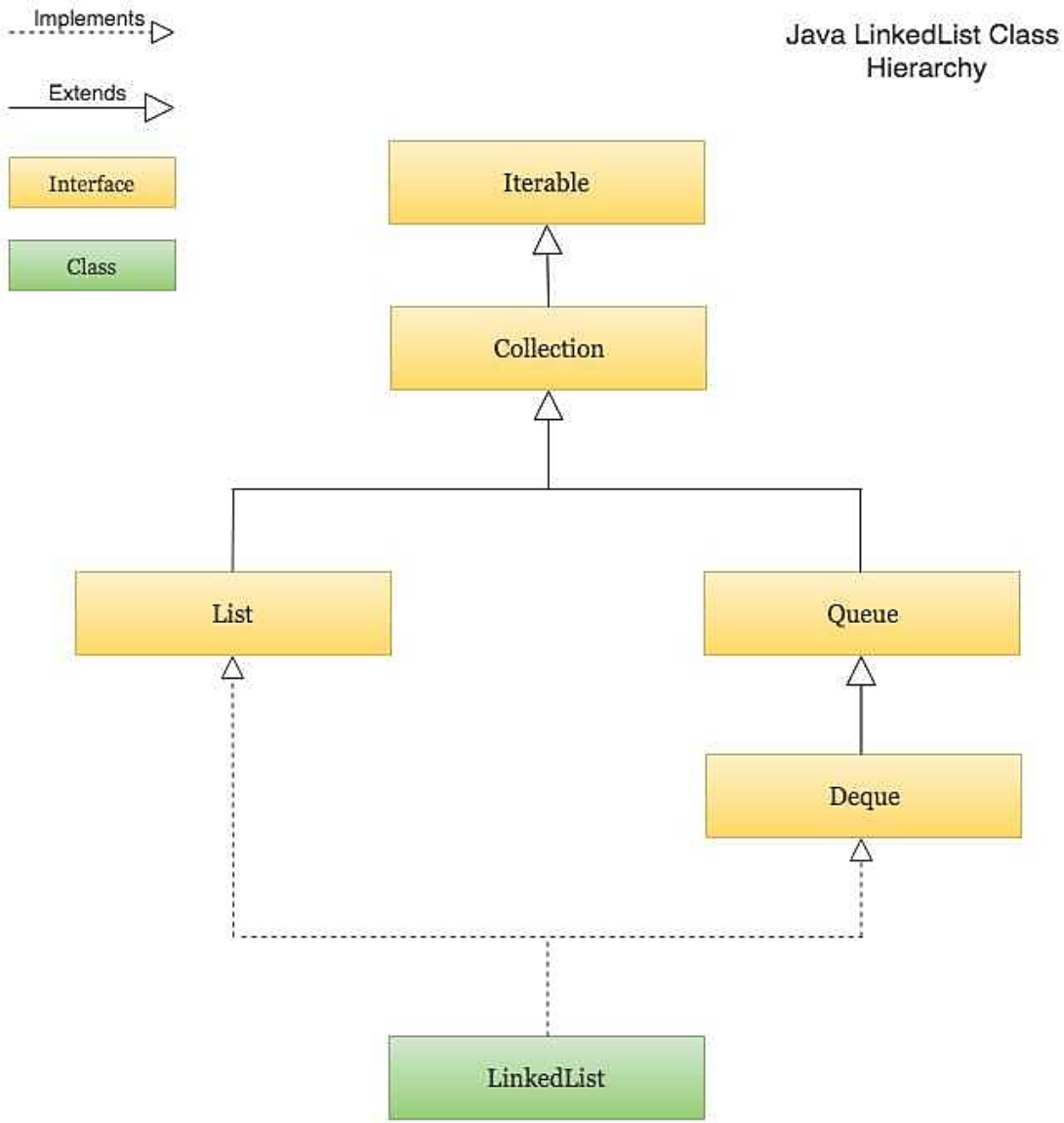


LinkedList

`LinkedList` es una clase que hace parte del framework de colecciones que se encuentra en el paquete `java.util`. Esta clase es una implementación de la estructura de datos de *lista enlazada*, que es una estructura de datos lineal donde los elementos no se almacenan en ubicaciones contiguas, sino que cada elemento es un objeto separado con una parte de datos y una parte de dirección. Los elementos están enlazados mediante punteros y direcciones. Cada elemento se conoce como un nodo.



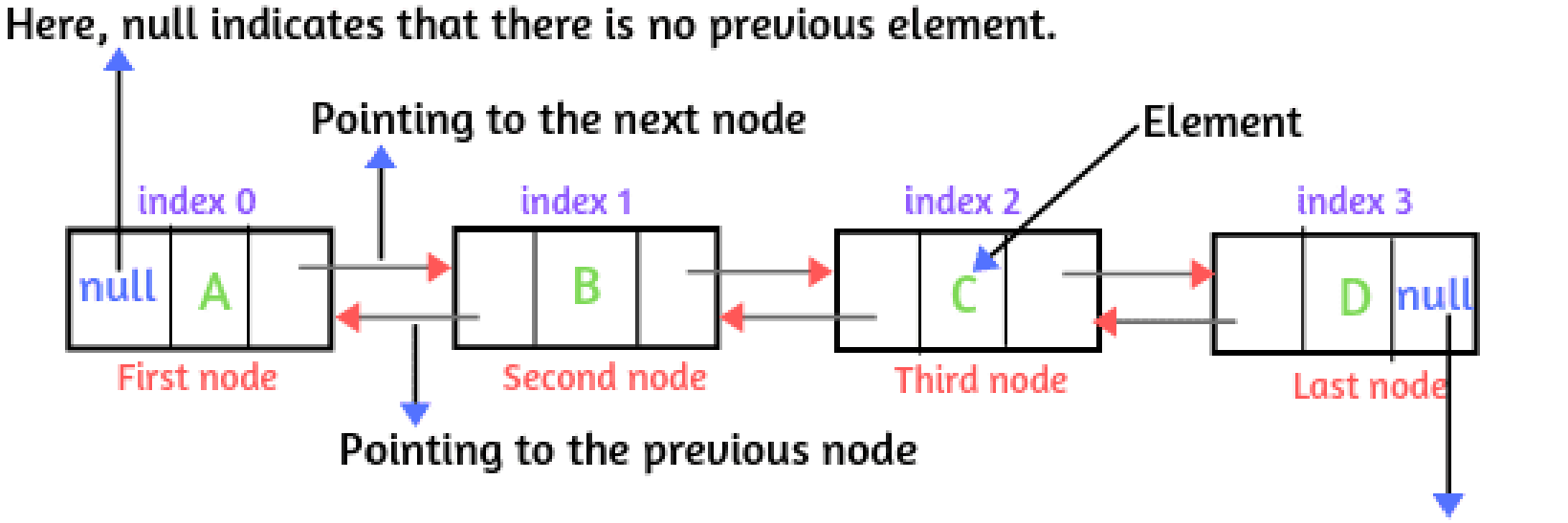
LinkedList


Implements
Extends
Interface
Class
Java LinkedList Class Hierarchy
Iterable
Collection
List
Queue
Deque
LinkedList


OBJECT-ORIENTED PROGRAMMING

LinkedList

Representation of Java LinkedList Node



A array representation of linear Doubly LinkedList in Java

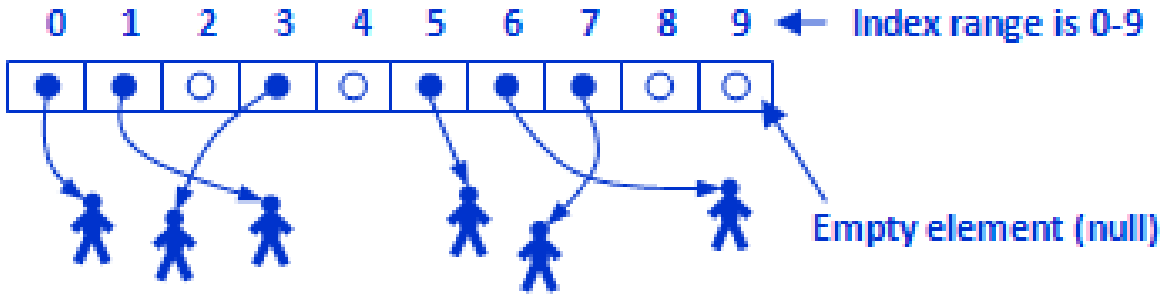
Métodos de LinkedList?

La principal diferencia entre una lista enlazada normal y una lista enlazada doblemente es que una lista enlazada doblemente contiene un puntero adicional, comúnmente llamado puntero anterior, además del puntero siguiente y los datos que se encuentran en la lista enlazada simple.

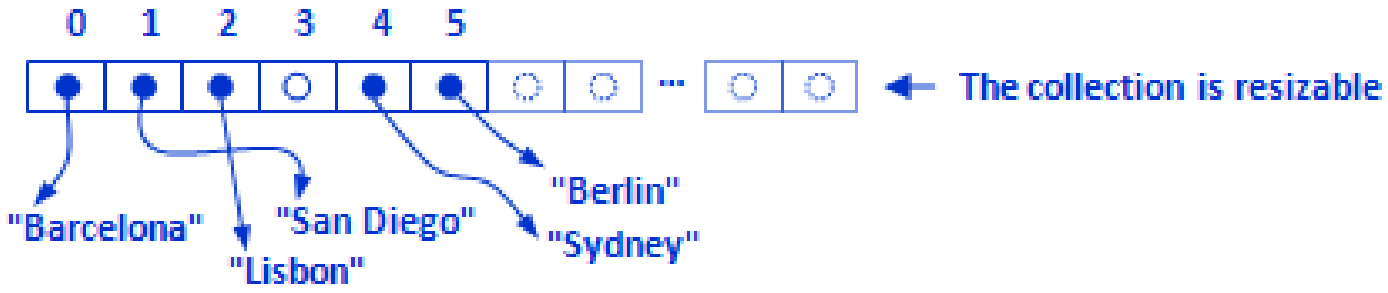
> Los métodos y operaciones para `ArrayList` y `LinkedList` no varían en sus resultados pues ambos implementan la interfaz `List`. Las diferencias entre una u otra van relacionadas con su rendimiento.

Array of 5 integers

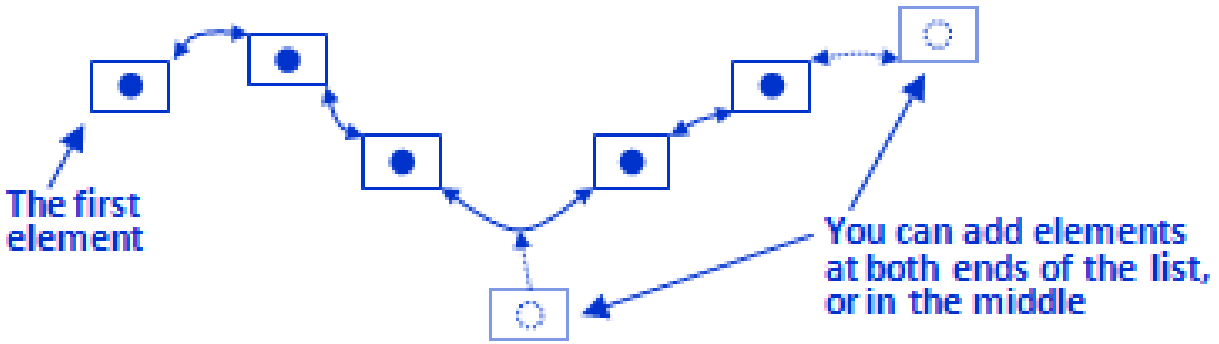
Array of 10 agents



ArrayList (collection) of strings, currently contains 6 elements



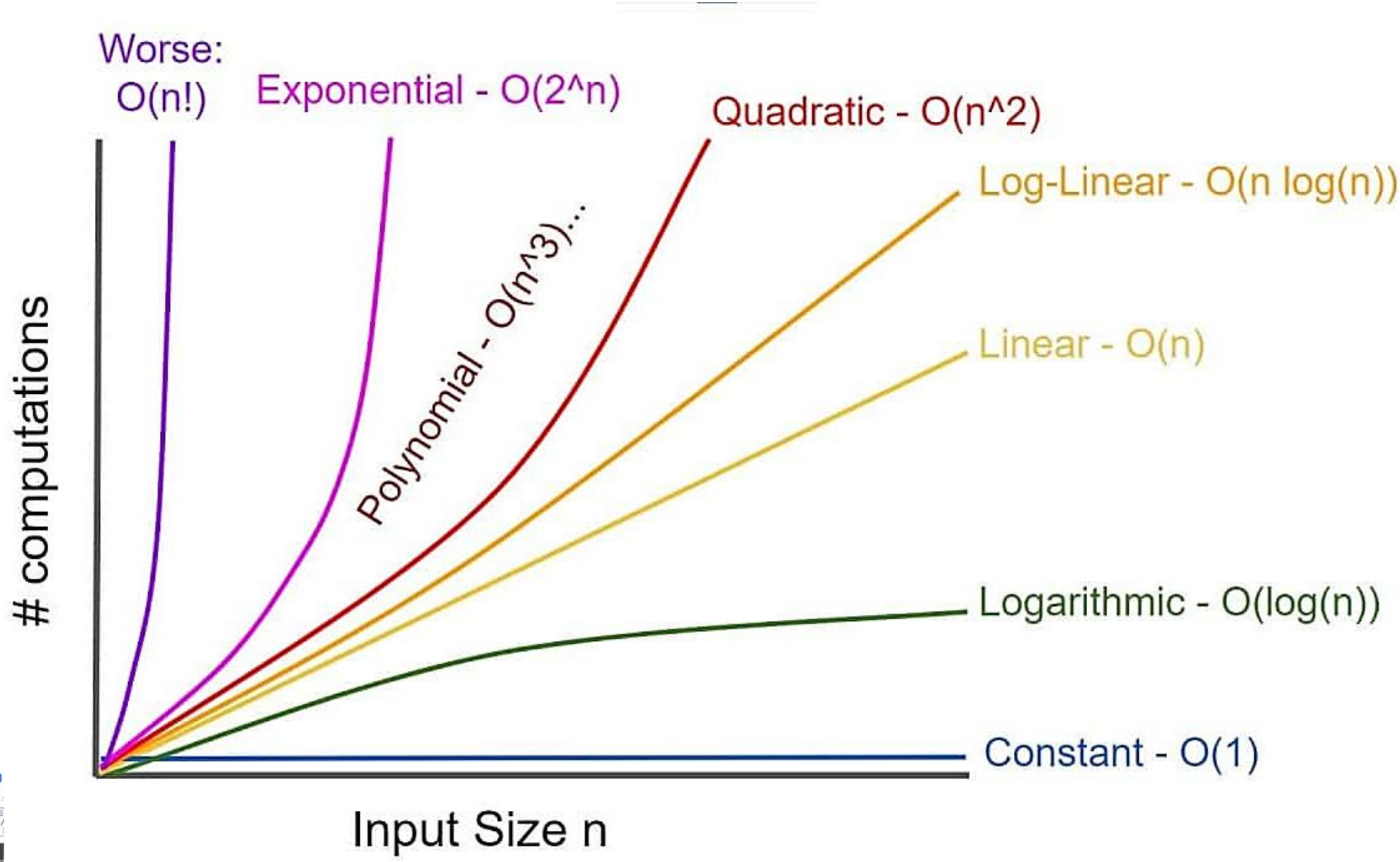
LinkedList (collection)



¿Cuándo usar ArrayList y LinkedList?

- En `LinkedList`, las inserciones o eliminaciones se realizan en tiempo constante utilizando iteradores, pero solo se permite el acceso secuencial a los elementos. Esto significa que se puede recorrer la lista hacia adelante o hacia atrás, pero encontrar un elemento específico lleva tiempo proporcional al tamaño de la lista.
- `ArrayList` permite un acceso aleatorio rápido para la lectura, lo que significa que se puede acceder a cualquier elemento en tiempo constante. Sin embargo, agregar o eliminar elementos desde cualquier posición que no sea el final requiere desplazar todos los elementos restantes. Además, si se agregan más elementos de los que la matriz subyacente puede contener, se asigna una nueva matriz (1.5 veces el tamaño de la anterior) y se copian los elementos antiguos a la nueva matriz.

Complejidad computacional en Collections



Complejidad computacional en Collections

List	Add	Remove	Get	Contains	Next	Data Structure
ArrayList	O(1)	O(n)	O(1)	O(n)	O(1)	Array
LinkedList	O(1)	O(1)	O(n)	O(n)	O(1)	Linked List
CopyOnWriteArrayList	O(n)	O(n)	O(1)	O(n)	O(1)	Array

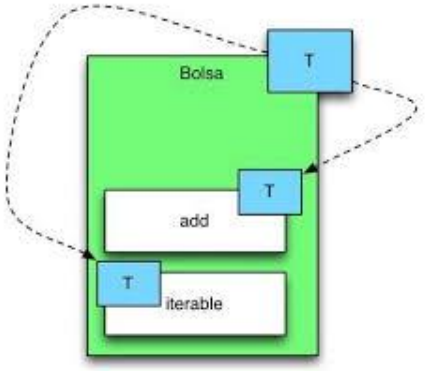
Set	Add	Remove	Contains	Next	Size	Data Structure
HashSet	O(1)	O(1)	O(1)	O(h/n)	O(1)	Hash Table
LinkedHashSet	O(1)	O(1)	O(1)	O(1)	O(1)	Hash Table + Linked List
EnumSet	O(1)	O(1)	O(1)	O(1)	O(1)	Bit Vector
TreeSet	O(log n)	O(log n)	O(log n)	O(log n)	O(1)	Redblack tree
CopyOnWriteArraySet	O(n)	O(n)	O(n)	O(1)	O(1)	Array
ConcurrentSkipListSet	O(log n)	O(log n)	O(log n)	O(1)	O(n)	Skip List

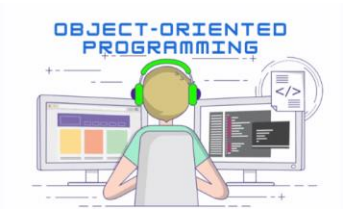
Map	Put	Remove	Get	ContainsKey	Next	Data Structure
HashMap	O(1)	O(1)	O(1)	O(1)	O(h / n)	Hash Table
LinkedHashMap	O(1)	O(1)	O(1)	O(1)	O(1)	Hash Table + Linked List
IdentityHashMap	O(1)	O(1)	O(1)	O(1)	O(h / n)	Array
WeakHashMap	O(1)	O(1)	O(1)	O(1)	O(h / n)	Hash Table
EnumMap	O(1)	O(1)	O(1)	O(1)	O(1)	Array
TreeMap	O(log n)	O(log n)	O(log n)	O(log n)	O(log n)	Redblack tree
ConcurrentHashMap	O(1)	O(1)	O(1)	O(1)	O(h / n)	Hash Tables
ConcurrentSkipListMap	O(log n)	O(log n)	O(log n)	O(log n)	O(1)	Skip List

Queue	Offer	Peak	Poll	Remove	Size	Data Structure
PriorityQueue	O(log n)	O(1)	O(log n)	O(n)	O(1)	Priority Heap
LinkedList	O(1)	O(1)	O(1)	O(1)	O(1)	Array
ArrayDequeue	O(1)	O(1)	O(1)	O(n)	O(1)	Linked List
ConcurrentLinkedQueue	O(1)	O(1)	O(1)	O(n)	O(1)	Linked List
ArrayBlockingQueue	O(1)	O(1)	O(1)	O(n)	O(1)	Array
PrioriritiyBlockingQueue	O(log n)	O(1)	O(log n)	O(n)	O(1)	Priority Heap
SynchronousQueue	O(1)	O(1)	O(1)	O(n)	O(1)	None
DelayQueue	O(log n)	O(1)	O(log n)	O(n)	O(1)	Priority Heap
LinkedBlockingQueue	O(1)	O(1)	O(1)	O(n)	O(1)	Linked List

Clases Genéricas

Como se pudo observar, al momento de crear listas se usa `<tipoDeDato>` para indicar el tipo de dato de los elementos de la colección, porque el concepto de una estructura de datos, se puede entender independientemente del tipo de elemento que manipula. Esto, se debe a que estas clases son creadas bajo el concepto de clases genéricas. Las clases genéricas permiten definir el concepto de cualquier otra clase de manera independiente de su tipo. De esta manera, se puede crear instancias de objetos con tipos específicos usando la clase genérica. Esto proporciona flexibilidad y reutilización del código, ya que se puede crear clases y métodos que trabajan con diferentes tipos de datos sin tener que duplicar código.





Clases Genéricas

```java
public class Caja<T> {

    private T contenido;

    public void ponerContenido(T objeto) {
        this.contenido = objeto;
    }

    public T obtenerContenido() {
        return contenido;
    }
}
```

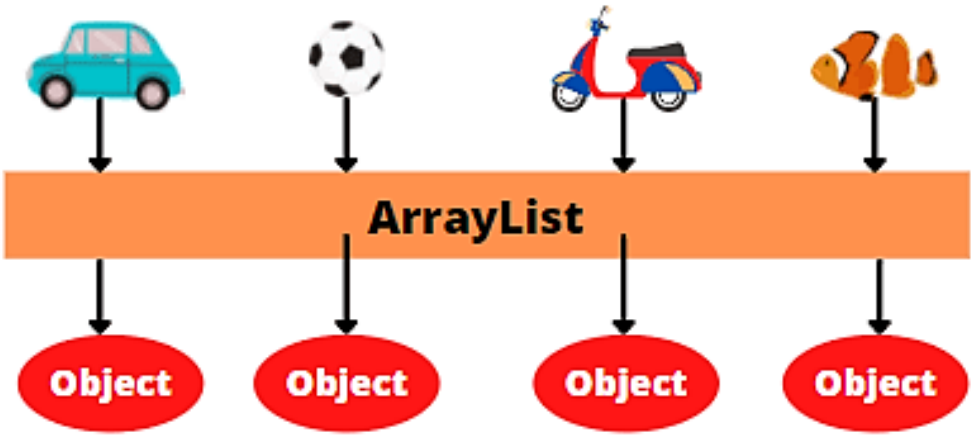
```java
// Crear una caja para enteros
Caja<Integer> cajaDeEnteros = new Caja<>();
cajaDeEnteros.ponerContenido(42);
int entero = cajaDeEnteros.obtenerContenido();
System.out.println("Contenido de la caja de enteros: " + entero
    );

// Crear una caja para cadenas de texto
Caja<String> cajaDeTexto = new Caja<>();
cajaDeTexto.ponerContenido("Hola, mundo!");
String texto = cajaDeTexto.obtenerContenido();
System.out.println("Contenido de la caja de texto: " + texto);
```

Clases Genéricas

WITHOUT GENERICS

Objects go IN as a reference to Car, Football, Scooter, and Fish objects

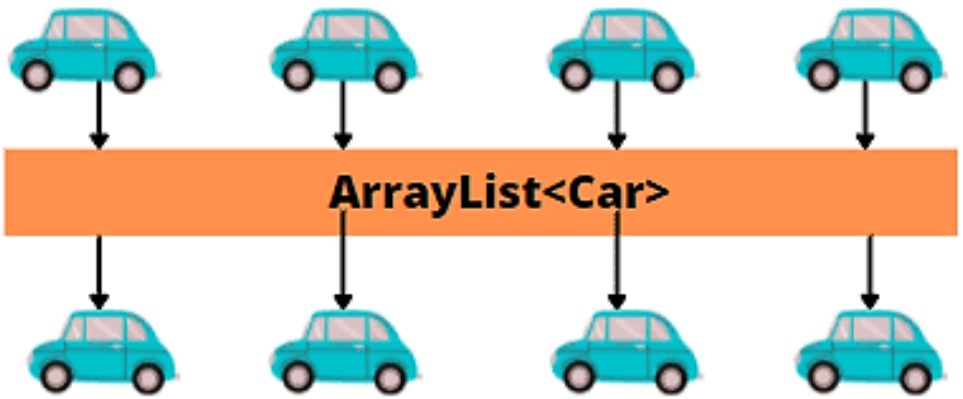
And come OUT as a reference of type Object.



WITH GENERICS

Objects go IN as a reference to only Car objects

And come OUT as a reference of type Car.



Clases Genéricas

```java
// Without Generics
List list = new ArrayList();
list.add("Hello");
String str = (String) list.get(0); // Needs casting

// With Generics
List<String> list = new ArrayList<>();
list.add("Hello");
String str = list.get(0); // No casting needed
```

Ejercicios

Calificación de estudiantes

Dada la siguiente información sobre las calificaciones de estudiantes de una institución educativa:

- Código
- Nombre
- Nota 1 (Peso de 30%)
- Nota 2 (Peso de 30%)
- Nota 3 (Peso de 40%)

El proceso se termina cuando el usuario indica que no hay más estudiantes.

Se pide calcular:

La nota definitiva de cada estudiante e indicar con un mensaje si aprobó o reprobó, utilizando funciones

Para aprobar, la nota deber ser mayor o igual a 3.0 y la información en su totalidad se debe almacenar en diccionarios

Para este programa cree la clase Estudiante

Ejercicios

A continuación se presenta el Informe Semanal de las ventas diarias realizadas por la Compañía **SweetCO** de sus cinco (5) principales productos.

VENTAS DE LA SEMANA

Producto	L	M	W	J	V	S	D
1	100	88	92	94	85	110	118
2	30	42	31	32	38	40	37
3	23	35	39	45	55	60	61
4	45	50	56	65	47	57	68
5	18	25	33	21	22	28	32

PRECIOS DE PRODUCTOS

1	2	3	4	5
1500	5000	6500	2500	22500

Usted ha sido contratado para desarrollar un programa que reciba la matriz de Ventas Semanales y calcule los ingresos de la compañía a partir del vector de precios de sus Productos.

Responda las siguientes preguntas del negocio:

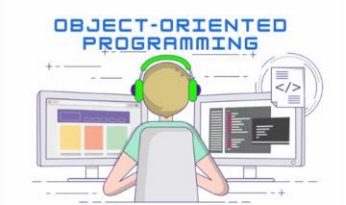
- Producto que genera mas ingresos en la semana
- El día de la semana con mayor ingresos por ventas

Ejercicios

Calcula la letra de un DNI, pediremos el DNI por teclado y nos devolverá el DNI completo. Para calcular la letra, cogeremos el resto de dividir nuestro DNI entre 23, el resultado debe estar entre 0 y 22. Haz un método donde según el resultado de la anterior formula busque en un array de caracteres la posición que corresponda a la letra.
Por ejemplo, si introduzco 70588387, el resultado será de 7 que corresponde a 'F'.

El listado de la tabla está en la siguiente tabla

Posicion	Letra	Posicion	Letra	Posicion	Letra
0	T	11	B	21	K
1	R	12	N	22	E
2	W	13	J		
3	A	14	Z		
4	G	15	S		
5	M	16	Q		
6	Y	17	V		
7	F	18	H		
8	P	19	L		
9	D	20	C		
10	X				



Ejercicios

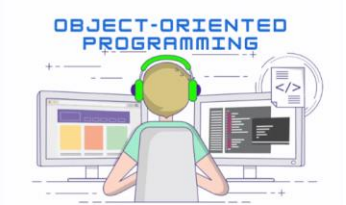
Cree un programa en el cual almacenará el listado de los departamentos de Colombia, posteriormente realizar las siguientes acciones:

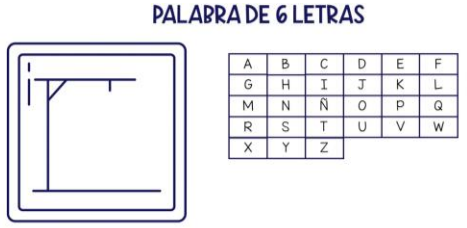
1. Mostrar en pantalla el primer y último departamento almacenado en el vector.
2. Conocer en qué posición se encuentra determinado departamento en el vector (Con su nombre).
3. Se debe poder eliminar una ciudad del vector, sin perder el orden del mismo.
4. Mostrar todos los departamentos almacenados en el vector.

La diferentes opciones deben poder accederse a través de un menú. Valide las opciones e información ingresada por el usuario.

Ejercicios

Crea el juego del ahorcado por consola. Al inicio del programa, pedirá los nombres de los dos jugadores. El primer turno será para el jugador 1.
Uno escribe una palabra y el otro la adivina, si este la adivina obtendrá un punto y escribirá una palabra, sino lo acierta el jugador actual suma un punto y escribe de nuevo otra palabra.
Cuando el jugador inserte su palabra también deberá añadir una pequeña pista, por ejemplo si la palabra es Madrid, la pista puede ser «Ciudad».
El jugador que tenga que acertar la palabra tendrá seis oportunidades (cabeza, cuerpo, brazos y piernas). No es necesario dibujar nada por pantalla, solo mostrar el número de oportunidades restantes.
Deberá mostrase los caracteres que el usuario inserta para esa palabra, para evitar que las repita, en caso de que lo haga, avisarle y no contar como error.
Cada vez que acierte o no la palabra, deberá mostrar la puntuación de ambos. El primero que llegue a 3 puntos gana.





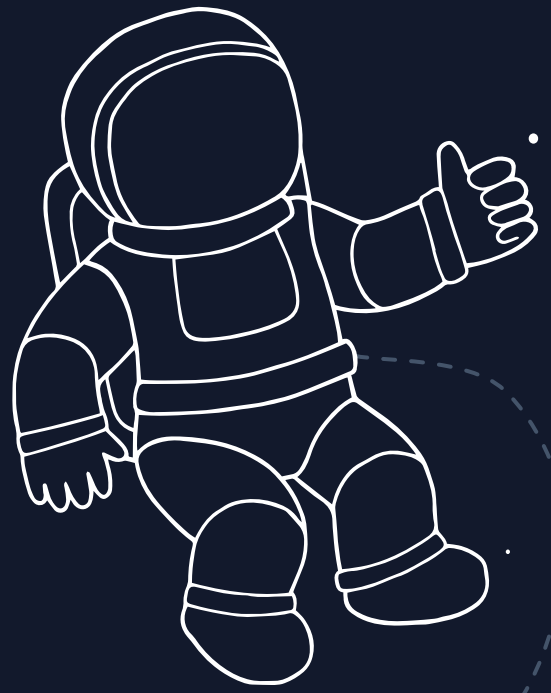
Ejercicios

Se quiere simular un juego en el que participan N jugadores y otra persona que hace de árbitro. Cada jugador elige 4 números en el rango [1, 10], pudiendo estar repetidos. A continuación, el árbitro, sin conocer los números que ha elegido cada jugador, selecciona 2 números A y B.

El programa debe ser capaz de calcular cuántos números de los seleccionados por cada jugador están comprendidos entre los valores A y B. Ganará el jugador que más números tenga en dicho intervalo.

Se pide implementar un programa modular que simule el juego para 3 jugadores, teniendo en cuenta que:

- Tanto los 4 datos de cada jugador, como los valores para A y B se introducirán por teclado. En todos los casos, el programa detectará la entrada de números erróneos, solicitando nuevamente el dato hasta que sea válido.
- Se deben mostrar por pantalla no solo los aciertos de cada jugador sino los datos que ha introducido cada jugador y los que ha seleccionado el árbitro. Por último, hay que imprimir la media aritmética de los aciertos de todos los jugadores



Programa académico CAMPUS

Módulo JAVA