Estructuras de almacenamiento. Colecciones

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Java ofrece dos tipos de estructuras para el almacenamiento de objetos:

- **Arrays**. Permiten el almacenamiento de tipos básicos y objetos. Es una estructura sencilla de longitud fija que solo permite datos de un mismo tipo.

- **Colecciones**. Son estructuras complejas que permiten almacenar solo objetos y cuyo tamaño puede variar a lo largo de la ejecución del programa.

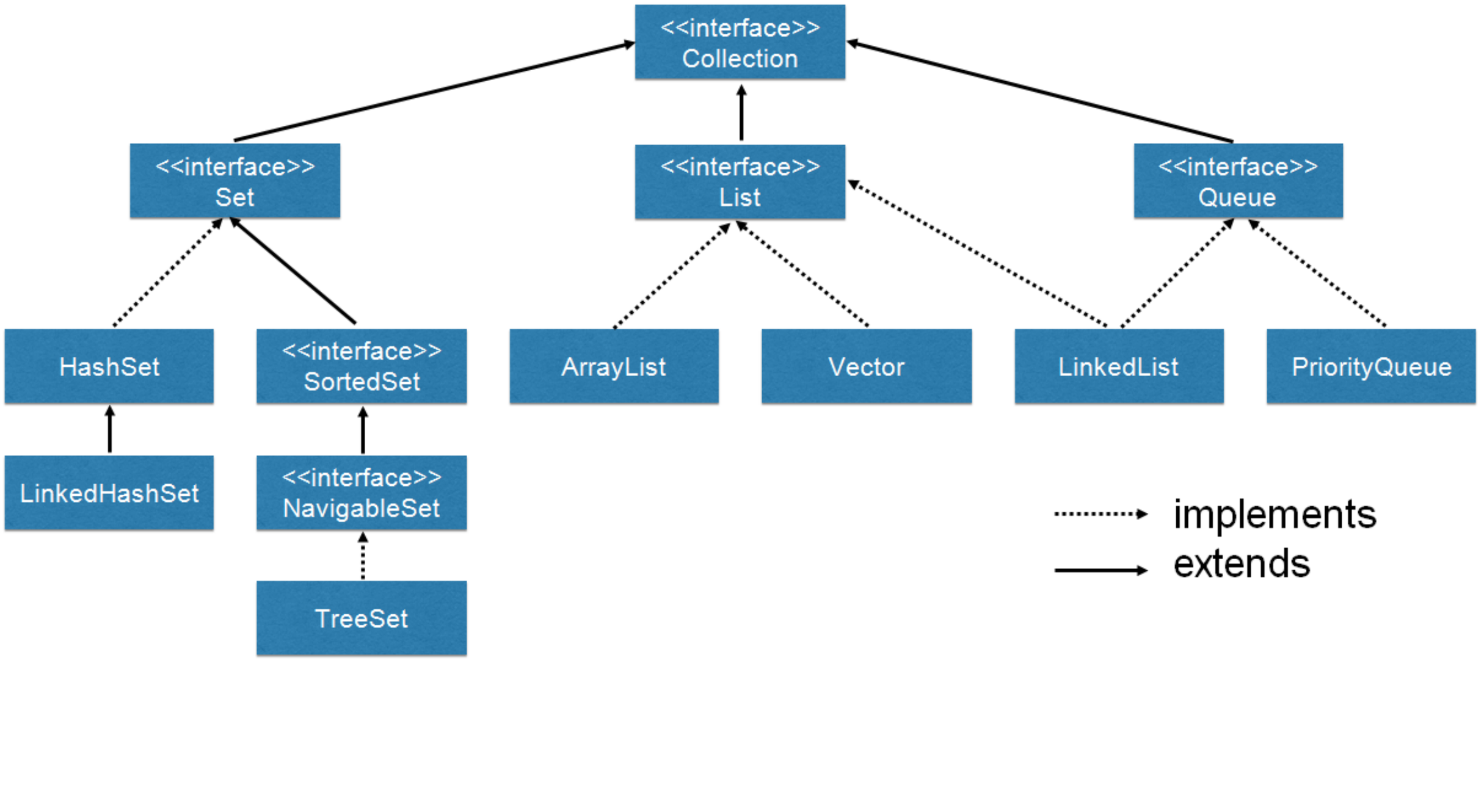
Colecciones

Interface Collection: define métodos que permiten añadir, eliminar y recorrer la estructura gracias a un Iterator.

Interface List: define métodos que permiten manejar colecciones de objetos con una secuencia determinada. El uso de este interface tiene control preciso sobre dónde es insertado cada elemento en la lista. El usuario puede acceder a los elementos por su índice entero y buscar por elementos en la lista.

Interface Set: colección de objetos donde no se admiten duplicados.

Collection Interface



Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

En cualquier clase de colección, debe haber una forma de introducir datos y otra de eliminarlos. Dependiendo del tipo de colección, existen distintos métodos para añadir y eliminar objetos.

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

En cualquier clase de colección, debe haber una forma de introducir datos y otra de eliminarlos. Dependiendo del tipo de colección, existen distintos métodos para añadir y eliminar objetos.

Busca en la documentación de Java los métodos definidos en la interfaz Collection

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Métodos de la interfaz Collection:

Devuelve el número de elementos de la colección:

int size();

Devuelve true si la colección está vacía:

boolean isEmpty();

Devuelve true si la colección contiene el elemento indicado:

boolean contains(Object element);

Devuelve true si la colección contiene todos los elementos de la colección c:

boolean containsAll(Collection c);

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Métodos de la interfaz Collection:

Devuelve true si la colección cambió como resultado de una llamada a este método.
Devuelve false si la colección no permite duplicados y ya contiene el elemento:

boolean add(Object element);

Añade todos los elementos de la colección c:

boolean addAll(Collection c);

Elimina el elemento:

boolean remove(Object element);

Elimina los elementos de la colección c contenidos en la colección:
boolean removeAll(Collection c);

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Métodos de la interfaz Collection:

Retiene solo los elementos de la colección c contenidos en la colección, es decir, elimina de esta colección los elementos restantes

boolean retainAll(Collection c);

Elimina todos los elementos de la colección:

void clear();

Devuelve un array con los elementos de la colección

Object[] toArray();

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

ArrayList

La clase **ArrayList** implementa una lista, internamente utiliza los arrays que hemos estudiado. Cuando se añaden elementos, la capacidad crece automáticamente.

Este array comienza con un tamaño determinado que podemos especificar en el constructor y va creciendo dinámicamente a medida que añadimos elementos.

La clase ArrayList implementa la interfaz Collection.

Revisa la documentación de la clase ArrayList, observa cómo implementa los métodos de la interfaz Collection que hemos visto y mira en detalle otros como el método get(índice). Observa si implementa o hereda toString()

https://docs.oracle.com/javase/8/docs/api/java/util/ArrayList.html

Colecciones

A partir de la versión J2SE5, podemos declarar los vectores y otros elementos de colecciones utilizando la notación de tipos genéricos, donde se indica el tipo de objeto entre los símbolos “< >”:

ArrayList <Integer> lista = new ArrayList<Integer>();

De esta forma, al indicar el tipo de objeto, la recuperación de los datos no necesita casting.

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Observa el siguiente **ejemplo**.

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

```java
import java.util.ArrayList;


public class EjArrayList1 {

    public static void main(String[] args) {

        String[] equipos = { "R. Vallecano", "Numancia", "Leganés", "Getafe","Betis","Valladolid", "Celta" };
        String[] equipos2 = { "Numancia", "Leganés", "Getafe"};

        ArrayList< String > listaEquipos = new ArrayList< String >();
        ArrayList< String > listaEquiposABorrar = new ArrayList< String >();

        for ( String equipo : equipos )
        {
            listaEquipos.add(equipo ); // añade un equipo al final de la lista
        }

        for ( String equipo : equipos2 )
        {
            listaEquiposABorrar.add( equipo );
        }

         // Imprimimos la lista
        System.out.println( "ArrayList: " );

        System.out.println(listaEquipos);

        //Borramos de listaEquipos aquellos que estén en la lista listaEquiposABorrar
        listaEquipos.removeAll(listaEquiposABorrar);

        System.out.println(listaEquipos);
    }
}
```

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Realiza una app que gestione una lista de Videojuegos y tenga el siguiente menú:

```
1. Insertar videojuego
2. Borrar videojuego
3. Mostrar lista de videojuegos
4. Borrar lista de videojuegos
5. Comprobar si un elemento está en la lista
6. Salir
```

```java
import java.util.ArrayList;
import java.util.Scanner;

public class EjArrayList2 {

    private static Scanner lectorEntrada;

    public static void main(String[] args) {

        lectorEntrada = new Scanner(System.in);
        ArrayList< String > listaVideojuegos = new ArrayList< String >();
        int opcion = 0;

        do
        {
            System.out.println("1. Insertar videojuego\n"
                    + "2. Borrar videojuego \n"
                    + "3. Mostrar lista de videojuegos\n"
                    + "4. Borrar lista de videojuegos \n"
                    + "5. Comprobar si un elemento está en la lista \n"
                    + "6. Salir\n");

            System.out.print("Introduce opción: ");
            opcion = lectorEntrada.nextInt();

            switch(opcion)
            {
                case 1:
                    insertarVideoJuego(listaVideojuegos);
                    break;
                case 2:
                    borrarVideojuego(listaVideojuegos);
                    break;
                case 3:
                    System.out.println(listaVideojuegos);
                    break;
                case 4:
                    listaVideojuegos.clear();
                    break;
                case 5:
                    comprobarVideojuegoEnLista(listaVideojuegos);
                    break;
                case 6:
                    System.out.println("Saliendo...");
                    break;
                default:
                    System.out.println("Opción errónea");
            }
        }while(opcion != 6);

        lectorEntrada.close();
    }
```

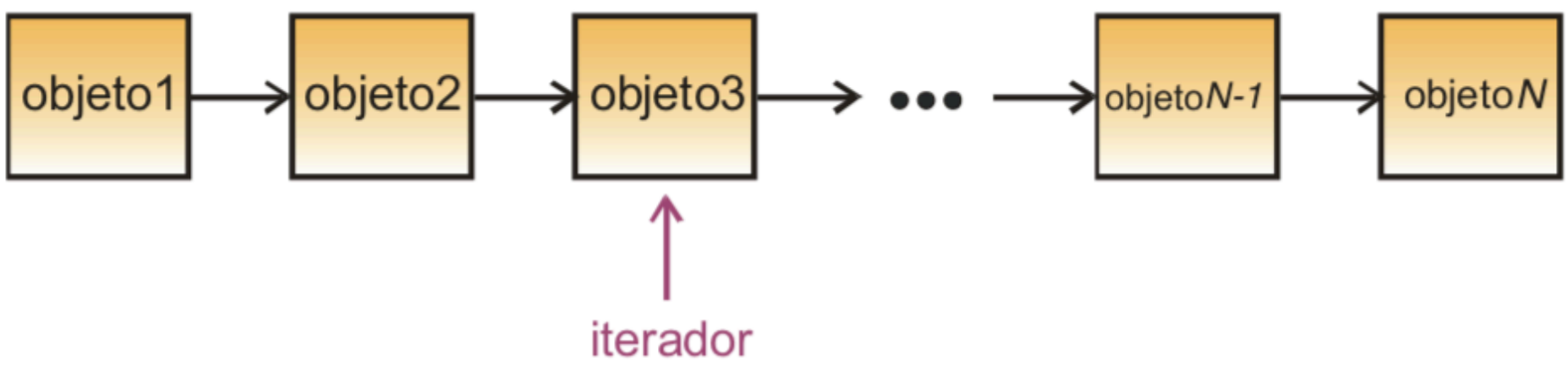
```java
private static void insertarVideoJuego(ArrayList< String > listaVideojuegos)
{
    String nombreVideojuego = "";

    System.out.println("Introduczca el nombre del videojuego");

    nombreVideojuego = lectorEntrada.next();

    if (listaVideojuegos.add(nombreVideojuego))
    {
        System.out.println("Videojuego " + nombreVideojuego + " insertado en la lista");
    }
    else
    {
        System.out.println("Error al insertar en la lista");
    }
}

private static void borrarVideojuego(ArrayList< String > listaVideojuegos)
{
    String nombreVideojuego = "";

    System.out.println("Introduczca el nombre del videojuego");

    nombreVideojuego = lectorEntrada.next();

    if (listaVideojuegos.remove(nombreVideojuego))
    {
        System.out.println("Videojuego " + nombreVideojuego + " borrado de la lista");
    }
    else
    {
        System.out.println("Error al borrar de la lista");
    }
}

private static void comprobarVideojuegoEnLista(ArrayList< String > listaVideojuegos)
{
    String nombreVideojuego = "";

    System.out.println("Introduczca el nombre del videojuego");

    nombreVideojuego = lectorEntrada.next();

    if (listaVideojuegos.contains(nombreVideojuego))
    {
        System.out.println("Videojuego " + nombreVideojuego + " está en la lista");
    }
    else
    {
        System.out.println("No está en la lista");
    }
}
```

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Iterator

En ocasiones necesitamos recorrer un ArrayList y en función de determinada situación borrar un elemento o finalizar el recorrido sin llegar al final de la lista. Para estos casos utilizaremos un bucle while y un i**terador**

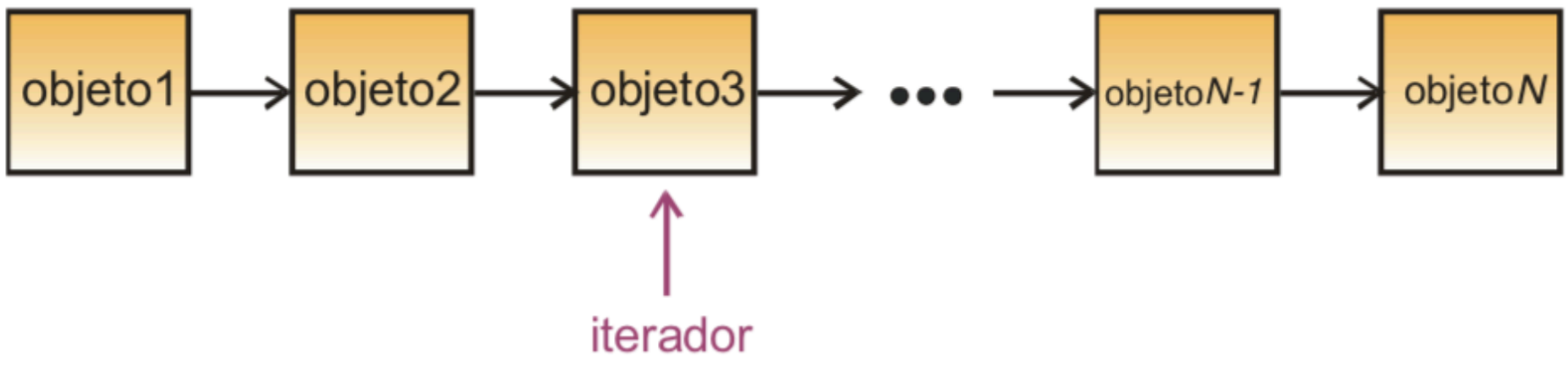


Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Iterator

El concepto de **iterador** nos permite movernos a través de una secuencia de objetos y seleccionar los adecuados sin que sea necesario conocer la estructura de la secuencia de almacenamiento.



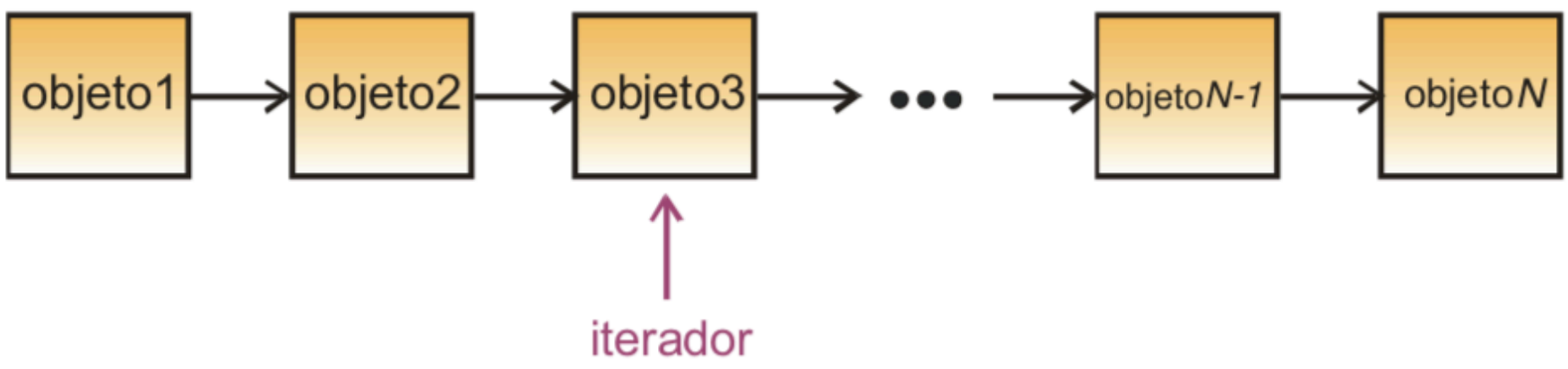
Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Iterator

Java incluye el **Interface Iterator**, nos permite recorrer los elementos de la colección y añade la operación de borrado.

Los iteradores son necesarios a la hora de manipular colecciones y estructuras de datos complejas donde no sea obvia la forma de desplazarse por ellas



Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

Colecciones

Iterator

Busca en la documentación de Java los métodos definidos en la interfaz Iterator

Colecciones

Métodos interfaz Iterator

El siguiente método devuelve true si hay más elementos:

Boolean hasNext();

El siguiente método devuelve el siguiente elemento de la colección, lanzando una Excepción NoSuchElementException si no existen más elementos:

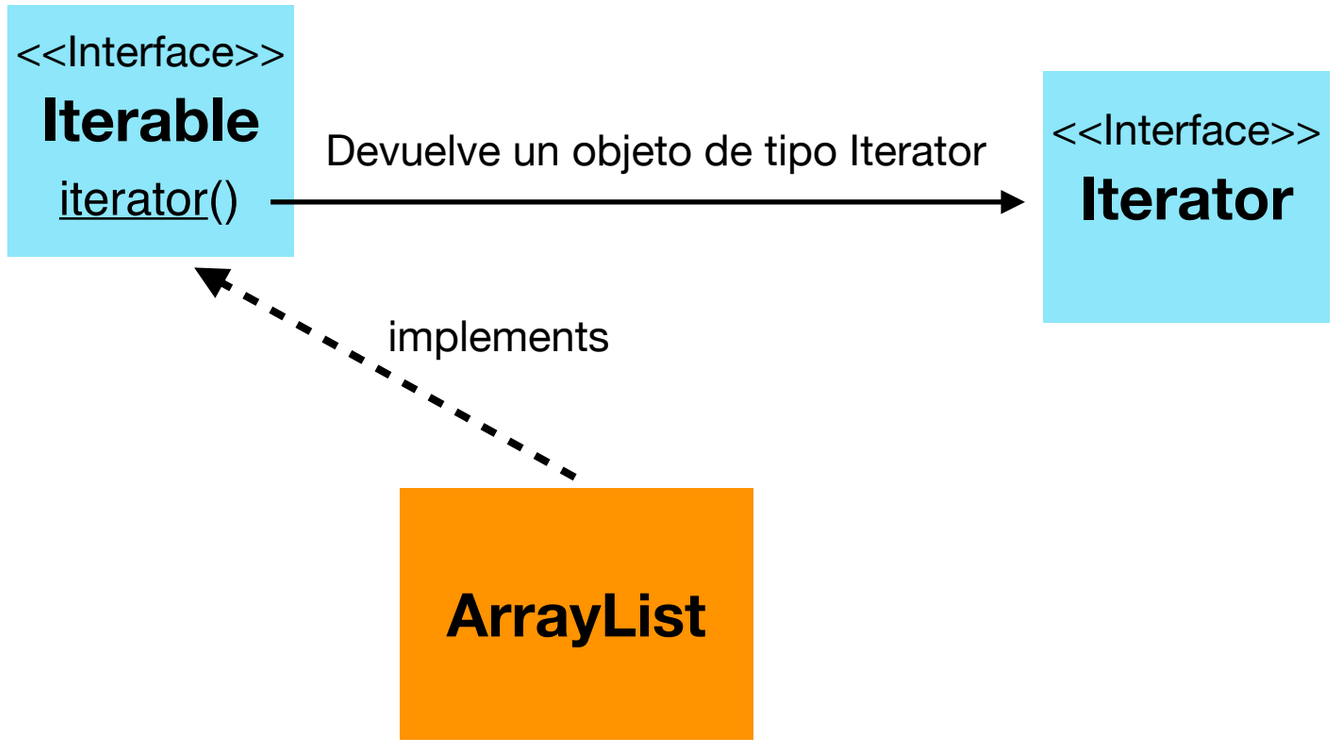
Object next();

El siguiente método elimina el último elemento devuelto:

void remove();

Colecciones

Observa en la documentación que ArrayList implementa la interfaz Iterable y ésta a su vez tiene un método iterator() que devuelve un objeto de tipo Iterator(interfaz)



Colecciones

Métodos interfaz Iterator

Ejemplo: En el ejemplo siguiente vamos recorriendo un array con un iterador y preguntando al usuario si desea borrarlo. Si borramos un elemento del arraylist mientras lo estamos recorriendo con un for nos daría problemas

```java
import java.util.ArrayList;

public class EjArrayList3 {

    public static void main(String[] args) {

        String[] equipos = { "R. Vallecano", "Numancia", "Leganés", "Getafe","Betis","Valladolid", "Celta" };

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

        for ( String equipo : equipos )
        {
            listaEquipos.add(equipo ); // añade un equipo al final de la lista
        }

        borrarEquipos(listaEquipos);

        System.out.println(listaEquipos);
    }
```

```java
private static void borrarEquipos(Collection <String>coleccionOrigen)
{
    Scanner lectorEntrada;
    lectorEntrada = new Scanner(System.in);

    boolean finBorrar = false;

    // Utilizamos un iterador
    Iterator< String > miIterador = coleccionOrigen.iterator();

    // recorremos la lista, cuando borremos uno salimos
    while ( miIterador.hasNext() && !finBorrar)
    {
        System.out.println("¿Desea borrar el equipo: " + miIterador.next() + " ? s/n");

        if (lectorEntrada.next().equals("s"))
        {
            miIterador.remove();
            finBorrar = true;
        }
    }
    lectorEntrada.close();
}
```

Colecciones

Métodos interfaz Iterator

Ejercicio: Debemos gestionar la venta de entradas de una sala de cine. Para ello manejaremos una lista de entradas y otra de entradas vendidas. De las entradas se almacenará el código de entrada que tendrá el formato “ENT-X” siendo X entre 1 y 20, el precio, el título de la película y la hora del pase (string). Inicialmente el sistema crea una lista de entradas con 20 entradas.

Cuando vendemos una entrada la sacamos de la lista de entradas y la pasamos a la de vendidas. Si se devuelve una entrada hacemos el proceso inverso. Se mostrarán las dos listas. No es necesario pedir datos al usuario.

```java
import java.util.ArrayList;
import java.util.Iterator;


public class Cine {
    static ArrayList <EntradaCine> entradasVendidas;
    static ArrayList <EntradaCine> entradas;
    static String nombreSala;

    static int NUM_LOCALIDADES = 10;


    public static void main(String[] args)
    {

        entradasVendidas = new ArrayList<EntradaCine>();
        entradas = new ArrayList<EntradaCine>();
        nombreSala = "dream cinema;";
        CrearEntradas();

        System.out.println("Entradas a la venta: ");
        System.out.println(entradas);

        venderEntrada("ENT-4");

        System.out.println("Entradas 4 vendida");

        System.out.println("Entradas a la venta: ");
        System.out.println(entradas);

        System.out.println("Entradas vendidas: ");
        System.out.println(entradasVendidas);

        devolverEntrada("ENT-4");

        System.out.println("Entrada 4 devuelta");

        System.out.println("Entradas a la venta: ");
        System.out.println(entradas);

        System.out.println("Entradas vendidas: ");
        System.out.println(entradasVendidas);
    }
```

```java
private static void CrearEntradas()
{
    final Double PRECIO = 15.0;
    final String TITULO_PELICULA = "Matrix";
    final String HORA_PELICULA = "18:30";

    String codEntrada;
    EntradaCine miEntrada;

    for (int i = 1; i <= NUM_LOCALIDADES; i++)
    {
        codEntrada = "ENT-" + i;
        miEntrada = new EntradaCine (codEntrada, PRECIO, TITULO_PELICULA, HORA_PELICULA);
        entradas.add(miEntrada);
    }
}

private static void venderEntrada(String codigoEntrada)
{
    boolean salir = false;
    EntradaCine entAux;

    // Utilizamos un iterador
    Iterator<EntradaCine> miIterador = entradas.iterator();

    // recorremos la lista hasta encontrar la que buscamos
    while ( miIterador.hasNext() && !salir)
    {
        entAux = miIterador.next();
        if (entAux.getCodEntrada().equals(codigoEntrada)) {
            miIterador.remove();

            entradasVendidas.add(entAux);

            salir = true;
        }
    }
}
```

```java
private static void devolverEntrada(String codigoEntrada)
{
    boolean salir = false;
    EntradaCine entAux;

    // Utilizamos un iterador
    Iterator<EntradaCine> miIterador = entradasVendidas.iterator();

    // recorremos la lista hasta encontrar la que buscamos
    while ( miIterador.hasNext() && !salir)
    {
        entAux = miIterador.next();
        if (entAux.getCodEntrada().equals(codigoEntrada)) {
            miIterador.remove();
            System.out.println(entAux);

            entradas.add(entAux);

            salir = true;
        }

    }
}
```

```java
public class Entrada {
    protected String codEntrada;
    protected Double precio;

    Entrada(String inCodEntrada, Double inPrecio)
    {
        codEntrada = inCodEntrada;
        precio = inPrecio;
    }

    Entrada(String inCodEntrada)
    {
        codEntrada = inCodEntrada;
        precio = 0.0;
    }

    public String getCodEntrada() {
        return codEntrada;
    }
    public void setCodEntrada(String codEntrada) {
        this.codEntrada = codEntrada;
    }
    public Double getPrecio() {
        return precio;
    }
    public void setPrecio(Double precio) {
        this.precio = precio;
    }
}
```

```java
public class EntradaCine extends Entrada
{
    private String tituloPelicula;
    private String horaPase;

    EntradaCine(String inCodEntrada, Double inPrecio, String inTituloPelicula, String inHoraPase)
    {
        super(inCodEntrada,inPrecio);
        tituloPelicula = inTituloPelicula;
        horaPase = inHoraPase;

    }

    EntradaCine(String inCodEntrada)
    {
        super(inCodEntrada);
    }
    public String getTituloPelicula() {
        return tituloPelicula;
    }
    public void setTituloPelicula(String tituloPelicula) {
        this.tituloPelicula = tituloPelicula;
    }

    public String getHoraPase() {
        return horaPase;
    }
    public void setHoraPase(String horaPase) {
        this.horaPase = horaPase;
    }

    public String toString()
    {
        return "Entrada: \n" +
                "titulo Pelicula: "+tituloPelicula+"\n" +
                "hora pase: "+horaPase+"\n" +
                "cod entrada: "+ codEntrada+ "\n" +
                "precio: "+ precio;
    }
```

```java
public boolean equals (Object objeto)
{
    boolean igual = false;

    if ((objeto!=null) && (objeto instanceof EntradaCine))
    {
        if (((EntradaCine)objeto).codEntrada.equals(this.codEntrada))
        {
            igual = true;
        }
    }

    return igual;
}
```

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

ArrayList

Ejercicio: Realizad una app que cree un ArrayList con varios vehículos. Que los muestre por pantalla, los ordene y los vuelva a mostrar por pantalla. Posteriormente borrará uno de los vehículos de la lista a partir de su matrícula. Ver programa principal.

Utilizad la clase vehículo de anteriores ejercicios.

¿Qué tenemos que hacer para ordenar los vehículos?

Cooperativa de Enseñanza
JOSÉ RAMÓN OTERO

ArrayList

¿Qué tenemos que hacer para ordenar los vehículos?

Revisar el método sort de la interfaz Collections: https://docs.oracle.com/javase/7/docs/api/java/util/Collections.html

¿Qué interfaz implementa?

https://docs.oracle.com/javase/7/docs/api/java/lang/Comparable.html

¿Qué interfaz deberías implementar en tu clase vehículo? para poder ordenar nuestra lista de vehículos

```
public static void main (String[] args)
{
    listaVehiculos = new ArrayList<Vehiculo>();

    Vehiculo vehic1 = new Vehiculo("4546-AAA", "rojo", 1998);
    Vehiculo vehic2 = new Vehiculo("222-BBB", "verde", 2008);
    Vehiculo vehic3 = new Vehiculo("2761-TFR", "azul", 1990);

    listaVehiculos.add(vehic1);
    listaVehiculos.add(vehic2);
    listaVehiculos.add(vehic3);


    System.out.println(listaVehiculos);

    Collections.sort(listaVehiculos);

    System.out.println("\n*** Después de ordenar ***\n");

    System.out.println(listaVehiculos);

    System.out.println("\n*** Vamos a eliminar el elemento a partir de una matrícula ***\n");

    borrarVehiculo("2761-TFR");

    System.out.println(listaVehiculos);
}
```

```java
private static void borrarVehiculo(String matricula)
{
    boolean salir = false;
    Vehiculo vehicAux;

    // Utilizamos un iterador
    Iterator<Vehiculo> miIterador = listaVehiculos.iterator();

    // recorremos la lista hasta encontrar la que buscamos
    while ( miIterador.hasNext() && !salir)
    {
        vehicAux = miIterador.next();
        if (vehicAux.getMatricula().equals(matricula)) {
            miIterador.remove();
            salir = true;
        }
    }
}
```

```java
package coleccion.arraylist;


public class Vehiculo implements Comparable<Vehiculo>{
    private String matricula="";
    private String color="";
    private int anio=0;


    /**
     * Constructor por defecto
     */

    public Vehiculo()
    {

        matricula = "--------";
        color = "---";
        anio = 0;

    }

    public Vehiculo(String inMatricula, String inColor, int inAnio)
    {

        matricula = inMatricula;
        color = inColor;
        anio = inAnio;
    }

    public String getMatricula()
    {
        return matricula;
    }

    public int getAnio()
    {
        return anio;
    }

    public void setMatricula(String matricula) {
        this.matricula = matricula;
    }

    public void setAnio(int anio) {
        this.anio = anio;
    }
```

```java
    public void setAnio(int anio) {
        this.anio = anio;
    }

    public boolean equals(Object inObjeto)
    {
        boolean iguales= false;



        if ((inObjeto != null) && (inObjeto instanceof Vehiculo))
        {
            iguales = ( (matricula == ((Vehiculo)inObjeto).getMatricula()) &&
                    (anio == ((Vehiculo)inObjeto).getAnio()  ) );
        }

        return iguales;
    }

    /**
     * Implementamos el interfaz comparable, será llamado internamente por una colección cuando
     * invoquemos al método sort()
     * @param vehic
     * @return  a negative integer, zero, or a positive integer as this object is less than,
     * equal to, or greater than the specified object.
     */
    public int compareTo(Vehiculo vehic)
    {
        int resultado = 0;

        if (vehic != null)
        {
            if(vehic.getAnio()< anio)
            {
                resultado = -1;
            }
            else if(vehic.getAnio() == anio)
            {
                resultado = 0;
            }else
            {
                resultado = 1;
            }
        }

        return resultado;
    }

    public String toString()
    {
        return "Vehículo: \n" +
                "Matrícula: "+matricula+"\n" +
                "Color: "+color+"\n" +
                "Año: "+ anio;
    }

}
```