
# Activad UT6.1

# Programación

*Alfonso García Jorge*

*1ºDAWNA*

1.- Implemente una claseArrayList, que permita mediante el método add(tipo_primitivo) agregar a la lista elementos de tipos primitivos int, float y double, y que automáticamente los cargue con sus respectivas “clases envolventes”.

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

no usages
public class Main {
    no usages
    public static void main(String[] args) {
        ArrayList<Object> misPrimitivos = new ArrayList();
        int miInt = 10;
        float miFloat = 34f;
        double miDoble = 23.8;
        char miChar = 'A';

        misPrimitivos.add(miInt);
        misPrimitivos.add(miFloat);
        misPrimitivos.add(miDoble);
        misPrimitivos.add(miChar);

        Iterator<Object> miIterador = misPrimitivos.iterator();
        while(miIterador.hasNext()){
            Object miObjeto = miIterador.next();
            System.out.println(miObjeto.toString() + " su tipo es: " + miObjeto.getClass());
        }
    }
}
```

```
Main
C:\Users\alfon\.jdks\openjdk-19.0.2\bin\java.exe "-javaagen
10 su tipo es: class java.lang.Integer
34.0 su tipo es: class java.lang.Float
23.8 su tipo es: class java.lang.Double
A su tipo es: class java.lang.Character

Process finished with exit code 0
```

2.- Realiza un programa que introduzca valores aleatorios (entre 0 y 100) en un ArrayList y que luego calcule la suma, la media, el máximo y el mínimo de esos números. El tamaño de la lista también será aleatorio y podrá oscilar entre 10 y 20 elementos ambos inclusive.

Main.java

```java
import java.util.ArrayList;

no usages
public class Main {
    no usages
    public static void main(String[] args) {

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

        int suma = 0;
        int maximo = 0;
        int minimo = 100;

        int tam = (int) (Math.random() * 11 + 10);

        for (int i = 0; i < tam; i++) {
            numAleatorios.add((int) (Math.random() * 101));
        }
        System.out.println("Numeros aletorios: " + numAleatorios);

        for (int n : numAleatorios) {
            suma += n;
        }
        System.out.println("La suma es: " + suma);
        System.out.println("La media es: " + (suma) / (tam));

        for (int n : numAleatorios) {
            if (n > maximo) {
                maximo = n;
            }

            if (n < minimo) {
                minimo = n;
            }
        }
        System.out.println("El número mínimo del arraylist es: " + minimo);
        System.out.println("El número máximo del arraylist es: " + maximo);
    }
}
```

```
C:\Users\alfon\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBr
Numeros aletorios: [73, 90, 3, 79, 98, 19, 82, 3, 89, 51, 38, 31, 57, 56]
La suma es: 769
La media es: 54
El número mínimo del arraylist es: 3
El número máximo del arraylist es: 98

Process finished with exit code 0
```

3.- Escribir una clase, de nombre PilaPalabras, para gestionar una estructura de pila que permita apilar y desapilar objetos de la clase String. La clase implementará el método apilarPalapra para poner una palabra encima de la pila, el método desapilarPalabra para quitar el elemento de la cima de la pila de devolviéndolo y el método obtenerPalabra para obtener la palabra situada en la cima de la pila sin quitarla de ella. También implementará el método pilaPalabrasVacia para determinar si la pila está o no vacía. Los métodos deben implementarse utilizando la clase LinkedList

Pila.java Main.java

```
no usages
public class Main {
    no usages
    public static void main(String[] args) {
        Pila miPila = new Pila();

        miPila.apilar( palabra: "nose");
        miPila.apilar( palabra: "que");
        miPila.apilar( palabra: "poner");
        miPila.apilar( palabra: "joer");
        System.out.println(miPila.toString());

        System.out.println("Mostrar palabra = " + miPila.obtener());
        System.out.println("Quitar una palabra = " + miPila.desapilar());
        System.out.println("Quitar otra palabra = " + miPila.desapilar());

        System.out.println(miPila.toString());

    }
}
```

Pila.java Main.java

```
import java.util.LinkedList;

2 usages
public class Pila {
    5 usages
    LinkedList<String> milinked;

    1 usage
    public Pila(){
        milinked = new LinkedList<>();
    }

    4 usages
    public void apilar(String palabra){
        milinked.add( index: 0,palabra);
    }

    2 usages
    public String desapilar(){
        return milinked.poll();

    }

    1 usage
    public String obtener(){
        return milinked.get(0);
    }

    @Override
    public String toString() {
        return "Pila= " + milinked.toString();
    }
}
```

Main

```
C:\Users\alfon\.jdks\openjdk-19.0.2\bin\java.
Pila= [joer, poner, que, nose]
Mostrar palabra = joer
Quitar una palabra = joer
Quitar otra palabra = poner
Pila= [que, nose]

Process finished with exit code 0
```

4.- Implementa una clase Ficha (ficha de dominó) con su constructor. Después implementa una clase tester con su constructor y su toString() que genere una Lista de 6 fichas al azar y las imprima ordenadas ascendentemente y descendentemente de acuerdo a un “peso ponderado” (que consistirá en el número más alto de la ficha multiplicado por 6 más el otro número). Para eso implementa también el método pesoPonderado(). Realizar el ejercicio con un TreeSet considerando que las fichas del dominó no se pueden repetir.

5.- Implementa una clase Persona (o Person) con atributos nombre y apellido con su constructor, sus getters y su toString. Después implementa una clase tester que añada unas cuantas personas en un HashMap<String, Person> (donde la key es el NIF de la persona).
Esta clase tester deberá tener métodos que impriman: las keys, los values y las entries del mapa.
Y comprobar que:

- Si intentas obtener un valor con una llave que no existe devuelve null.
- Si intentas añadir una entrada <llave, valor> con una llave ya existente, esta entrada sobrescribe la pre-existente.
- Si generamos un TreeMap a partir del HashMap sus entradas estarán ordenadas por key.

```java
import java.util.TreeMap;

public class Main {
    public static void main(String[] args) {
        Tester mapeoPersonas = new Tester();

        mapeoPersonas.add( DNI: "fffff", new Persona( nombre: "Pedro", apellido: "Jimenez"));
        mapeoPersonas.add( DNI: "fgdgd", new Persona( nombre: "Juan", apellido: "Alfonso"));
        mapeoPersonas.add( DNI: "dfefd", new Persona( nombre: "Ana", apellido: "Leon"));
        mapeoPersonas.add( DNI: "ttyyy", new Persona( nombre: "Paco", apellido: "Valdemoro"));
        System.out.println("Se imprime lista");
        mapeoPersonas.imprimirEntri();
        System.out.println("Se añade uno mas");
        mapeoPersonas.add( DNI: "dfssa", new Persona( nombre: "Gabriela", apellido: "Marcel"));
        mapeoPersonas.imprimirEntri();

        TreeMap<String, Persona> miTree = new TreeMap<>();
        miTree.putAll(mapeoPersonas.getGrupoPersonas());

        System.out.println("Se imprime Treemap");
        for (String Key: miTree.keySet()){
            System.out.println(Key + " = " + miTree.get(Key));
        }
    }
```

Main.java | Tester.java | Persona.java

```java
public class Tester {
    private HashMap<String,Persona> grupoPersonas;

    public Tester(){
        grupoPersonas = new HashMap<>();
    }

    public HashMap<String, Persona> getGrupoPersonas() {
        return grupoPersonas;
    }

    public void setGrupoPersonas(HashMap<String, Persona> grupoPersonas) {
        this.grupoPersonas = grupoPersonas;
    }

    public void add(String DNI,Persona persona){
        grupoPersonas.put(DNI,persona);
    }
    public void imprimirClave(){
        for(String Key: grupoPersonas.keySet()){
            System.out.println(Key);
        }
    }

    public void imprimirValor(){
        for(Persona value: grupoPersonas.values()){
            System.out.println(value);
        }
    }
```

Main

```
C:\Users\alfon\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\Je
Se imprime lista
dfefd = Persona{nombre='Ana', apellido='Leon'}
ttyyy = Persona{nombre='Paco', apellido='Valdemoro'}
fffff = Persona{nombre='Pedro', apellido='Jimenez'}
fgdgd = Persona{nombre='Juan', apellido='Alfonso'}
Se añade uno mas
dfefd = Persona{nombre='Ana', apellido='Leon'}
ttyyy = Persona{nombre='Paco', apellido='Valdemoro'}
fffff = Persona{nombre='Pedro', apellido='Jimenez'}
fgdgd = Persona{nombre='Juan', apellido='Alfonso'}
dfssa = Persona{nombre='Gabriela', apellido='Marcel'}
Se imprime Treemap
dfefd = Persona{nombre='Ana', apellido='Leon'}
dfssa = Persona{nombre='Gabriela', apellido='Marcel'}
fffff = Persona{nombre='Pedro', apellido='Jimenez'}
fgdgd = Persona{nombre='Juan', apellido='Alfonso'}
ttyyy = Persona{nombre='Paco', apellido='Valdemoro'}

Process finished with exit code 0
```

6.- Escribir un programa Java que crea un HashSet de Objetos de tipo Coche. El programa pide por teclado los datos de los coches y los guarda en el HashSet. A continuación utilizará un iterator para mostrar por pantalla lo siguiente:

- Todos los coches introducidos.
- Todos los coches de una marca determinada.
- Todos los coches con menos de un número determinado de Kilómetros.
- El coche con mayor número de Kilómetros.
- Todos los coches ordenados por número de kilómetros de menor a mayor.

7.- Crea un programa en Java para calcular el String de mayor longitud de todos los contenidos en un ArrayList de String. El programa utilizará los siguientes métodos:

- Método leerArray(): este método recibe como parámetro el arrayList de Strings vacío. El método pide por teclado cadenas de caracteres y las añade al ArrayList. La lectura de cadenas termina cuando se introduce la palabra "FIN".
- Método cadenaMasLarga():este método recibe como parámetro el ArrayList de Strings con todas las cadenas leídas anteriormente y devuelve el String de mayor longitud.

```java
import java.util.ArrayList;
import java.util.Scanner;
no usages
public class Main {

    no usages
    public static void main(String[] args) {
        ArrayList<String> cadenasCaracteres = new ArrayList();
        leerArray(cadenasCaracteres);
        System.out.println("Cadena de mayor longitud : " + cadenaMasLarga(cadenasCaracteres));
    }
    1 usage
    public static void leerArray(ArrayList<String> cadenas) {
        Scanner sc = new Scanner(System.in);
        String cadena;
        boolean masCadenas;
        do {
            masCadenas = true;
            System.out.print("Introduce una cadena (Fin para terminar): ");
            cadena = sc.nextLine();
            if (cadena.equalsIgnoreCase( anotherString: "FIN")) {
                masCadenas = false;
            } else {
                cadenas.add(cadena);
            }
        } while (masCadenas);
    }

    1 usage
    public static String cadenaMasLarga(ArrayList<String> cadenas) {
        String cadenaMayor = cadenas.get(0);
        for (int i = 1; i < cadenas.size(); i++) {
            if (cadenas.get(i).length() > cadenaMayor.length()) {
                cadenaMayor = cadenas.get(i);
            }
        }
        return cadenaMayor;
    }
}
```

```
C:\Users\alfon\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Prog
Introduce una cadena (Fin para terminar): hola
Introduce una cadena (Fin para terminar): adios
Introduce una cadena (Fin para terminar): que
Introduce una cadena (Fin para terminar): lapalabramaslarga
Introduce una cadena (Fin para terminar): nose
Introduce una cadena (Fin para terminar): que
Introduce una cadena (Fin para terminar): mas
Introduce una cadena (Fin para terminar): poner
Introduce una cadena (Fin para terminar): fin
Cadena de mayor longitud : lapalabramaslarga

Process finished with exit code 0
```