CIFP César Manrique.

Programación 1º de Desarrollo de Aplicaciones Web

Profesor: José David Díaz Díaz

Actividades de la Unidad 11: Colecciones

Juan Carlos Francisco Mesa



Esta obra está licenciada bajo la Licencia Creative Commons Atribución 4.0 Internacional. Para ver una copia de esta licencia, visite http://creativecommons.org/licenses/by/4.0/ o envíe una carta a Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Índice

| Actividades1 |
|---|
| Desarrollo |
| Actividades de comprobación2 |
| 12.1. ¿Qué es Collection?2 |
| 12.2. Los tipos genéricos sirven para:2 |
| 12.3. ¿Para qué sirve una lista?2 |
| 12.4. Un conjunto es una colección de elementos:3 |
| 12.5. ArrayList y LinkedList se diferencian: |
| 12.6. Los métodos de la interfaz Set:3 |
| 12.7. Si la variable a referencia un objeto ArrayList, la expresión new TreeSet (a):3 |
| 12.8. ¿Qué es Collections?4 |
| 12.9. Un mapa en Java es:4 |
| 12.10. Si queremos cambiar el valor de una entrada en un mapa, usaremos el método:4 |
| Actividades de Aplicación5 |
| 12.16. Implementa una aplicación que gestione los socios de un club usando la clase Socio |
| implementada en la Actividad resuelta 12.11. En particular, se deberán ofrecer las opciones de |
| alta, baja y modificación de los datos de un socio. Además, se listarán los socios por nombre o |
| por antigüedad en el club5 |
| 12.17. Implementa la clase Cola genérica utilizando un objeto ArrayList para guardar los |
| elementos9 |
| 12.18. Implementa la clase Pila genérica utilizando un objeto ArrayList para guardar los |
| elementos |

| 12.19. Escribe un programa donde se introduzca por consola una frase que conste exclusiva |
|--|
| mente de palabras separadas por espacios. Las palabras de la frase se almacenarán en una |
| lista. Finalmente, se mostrarán por pantalla las palabras que estén repetidas y, a continuación, |
| las que no lo estén14 |
| 12.22. Introduce por teclado, hasta que se introduzca « fin », una serie de nombres, que se |
| inser tarán en una colección, de forma que se conserve el orden de inserción y que no pue dan |
| repetirse. Al final, la colección se mostrará por pantalla15 |

Actividades

Actividades de la Unidad 11: Colecciones.

En este documento se detallan las actividades a realizar. Se entregará al profesor en la plataforma digital dos ficheros. Un primer fichero pdf con todas las actividades a realizar, el nombre del fichero será "unidad2 + nombre del alumno.pdf". Añadir en el fichero pdf por cada actividad de programación dos capturas de pantalla, una del código y otra de su ejecución. También en el fichero pdf copiar todas las preguntas y las respuestas correctas de las actividades de comprobación. Además, entregar un segundo fichero comprimido con todos los códigos fuentes de cada actividad de programación realizada.

Todas las actividades resueltas se deberán de analizar y no se entregarán.

A continuación, detallamos las actividades a realizar:

- Actividades propuestas. No realizar ninguna.
- Actividades de comprobación. Realizarlas todas. Copiar todas las preguntas y sus respuestas correctas.
- Actividades de aplicación. Realizar las siguientes 12.16, 12.17, 12.18, 12.19, 12.22 y 12.23.
- Actividades de ampliación. No realizar ninguna.

Desarrollo

Actividades de comprobación.

12.1. ¿Qué es Collection?

- a) Una interfaz.
- b) Una clase.
- c) Un sistema operativo.
- d) Un método.

Volver al índice

12.2. Los tipos genéricos sirven para:

- a) Usar objetos de la clase Object.
- b) Usar variables primitivas.
- c) Usar tipos parametrizados.
- d) No tener que usar ningún tipo.

Volver al índice

12.3. ¿Para qué sirve una lista?

- a) Guardar datos primitivos.
- b) Guardar datos que no se pueden repetir.
- c) No tener que ordenar un conjunto de datos.
- d) Guardar, de forma dinámica, datos que se pueden repetir y ordenar.

12.4. Un conjunto es una colección de elementos:

- a) Que no admiten orden.
- b) Que admiten repeticiones.
- c) Que no se pueden alterar.
- d) Cuyo criterio fundamental es el de pertenecer al conjunto.

Volver al índice

12.5. ArrayList y LinkedList se diferencian:

- a) En el número de elementos.
- b) En el rendimiento.
- c) En el orden de los elementos.
- d) En nada.

Volver al índice

12.6. Los métodos de la interfaz Set:

- a) Son los mismos que los de List.
- b) Son los mismos que los de Collection.
- c) Son implementados en la clase ArrayList.
- d) Esta interfaz no tiene métodos.

Volver al índice

12.7. Si la variable a referencia un objeto ArrayList, la expresión new TreeSet (a):

- a) Devuelve un conjunto ordenado con los elementos de a.
- b) Es incorrecta.
- c) Devuelve una lista ordenada.
- d) Devuelve una tabla.

12.8. ¿Qué es Collections?

- a) Una clase cuyos objetos están repetidos.
- b) Una interfaz de la que heredan todas las colecciones.
- c) Una clase con métodos estáticos que sirven para gestionar colecciones.
- d) Nada, le sobra la ese.

Volver al índice

12.9. Un mapa en Java es:

- a) Un gráfico con las relaciones de herencia entre interfaces.
- b) Una colección.
- c) Una representación de los datos por pantalla.
- d) Una estructura dinámica cuyos elementos son parejas clave valor.

Volver al índice

12.10. Si queremos cambiar el valor de una entrada en un mapa, usaremos el método:

- a) put().
- b) set ().
- c) add ().
- d) insert ().

Actividades de Aplicación.

12.16. Implementa una aplicación que gestione los socios de un club usando la clase Socio implementada en la Actividad resuelta 12.11. En particular, se deberán ofrecer las opciones de alta, baja y modificación de los datos de un socio. Además, se listarán los socios por nombre o por antigüedad en el club.

```
package Main;

import java.io.Serializable;
import java.time.LocalDate;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeFormatter;
import java.time.format.DateTimeFormatter;

public class Socio implements Comparable<Socio>, Serializable {

String dni;
String nombre;
LocalDate fechaAlta;

public Socio(String dni, String nombre, String alta) {

this.dni = dni;
this.nombre = nombre;
DateTimeFormatter f = DateTimeFormatter.ofPattern(pattern: "dd/MM/yyyy");
this.fechaAlta = LocalDate.parse(net.alta, formatter:f);
}

public Socio(String dni) {
 this.dni = dni;
}

public Socio(String dni) {
 this.dni = dni;
}

public socio(String dni) {
 this.dni = dni;
}

@Override
public int compareTo(socio o) {
 return dni.compareTo(socio o) .dni);
}

@Override
public boolean equals(Object o) {
 return dni.equals(sombject: ((Socio) o).dni);
}

@Override
public string toString() {
 return dni.equals(sombject: ((Socio) o).dni);
}

@Override
public string toString() {
 return socio(* + "dni=" + dni + ", nombre=" + nombre + ", antiguedade" + antiguedad() + "}\n";
}
```

```
case 4;

System.out.println("Lista de socios ordenados por "
+ "nombre:");
club.listarSociosPorNombre();
break;
case 5;
cse 6;
system.out.println("Lista de socios ordenados por "
+ "antiguedad en el club.");
club.listarSociosPorAntiguedad();
break;
default:
System.out.println(: "Saliendo...");
break;
default:
$
yestem.out.println(: "Opcion invalida.");
$
yestem.out.printle: "AnDNI: ");
$
system.out.printle: "NaDNI: ");
$
system.out.printle: "NaDNI del socio a dar de baja: ");
return datosDni(scanner);
}

public static Socio updateSocio(Scanner scanner){
System.out.print(: "NaDNI del socio a modificar: ");
return datosDni(scanner);
}
```

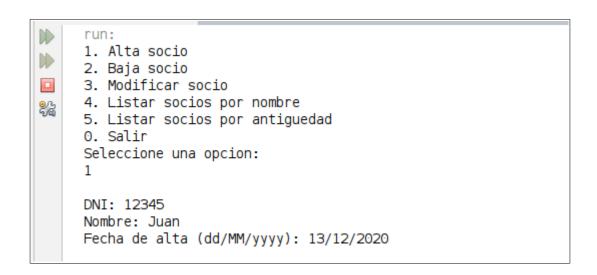
```
package Main;
         import java.util.ArrayList;
import java.util.Collections
import java.util.Comparator;
import java.util.List;
        /**

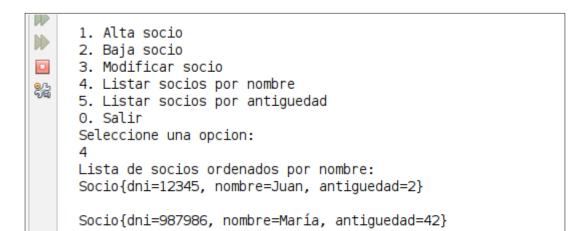
* @author juancfm
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
                   public class Club {
                           public List<Socio> socios;
       public Club() {
    this.socios = new ArrayList⇔();
}
                          public void altaSecie(Socio socio) {
    socios.add(@: socio);
}
        7
        7
                            public void bajaSecie(Socio socio) {
    socios.remove(@: socio);
}
                            public void modificarSocio(Socio socio) {
  int index = socios.indexOf(⊕ socio);
  if (index >= 0) {
    socios.set(index, almost: socio);
  }
        F
                                      lic void listarSociosPorNombre() {
Collections.sort(\int\ ocios, new ComparatorsSocio>() {
    @Override
    public int compare(Socio ol, Socio o2) {
        return ol.nombre.compareTo(wortherSprings o2.nombre);
    }
}
         40
41
42
43
44
45
46
47
48
49
51
53
53
55
56
60
61
                                    });
                                    for (Socio socio : socios) {
   System.out.println(=: socio);
                                      lic void listarSociosPorAntiguedad() {

Collections.sort(ibm: socios, new Comparator≤Socio≥() {

@Override
public int compare(Socio ol, Socio o2) {

return ol.antiguedad() - o2.antiguedad();
}
         for (Socio socio : socios) {
   System.out.println(x: socio);
```



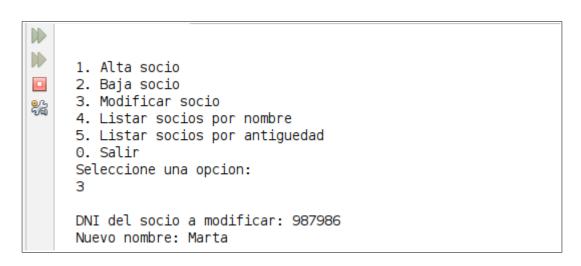


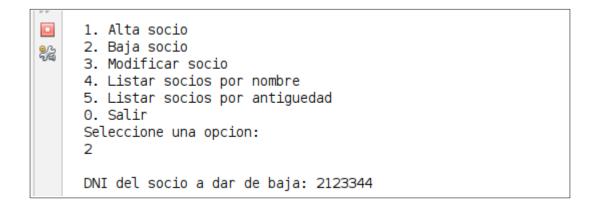
Socio{dni=2123344, nombre=Pedro, antiguedad=1}

1. Alta socio
2. Baja socio
3. Modificar socio
4. Listar socios por nombre
5. Listar socios por antiguedad
0. Salir
Seleccione una opcion:
5
Lista de socios ordenados por antiguedad en el club:
Socio{dni=2123344, nombre=Pedro, antiguedad=1}

Socio{dni=12345, nombre=Juan, antiguedad=2}

Socio{dni=987986, nombre=María, antiguedad=42}





```
1. Alta socio
2. Baja socio
3. Modificar socio
4. Listar socios por nombre
5. Listar socios por antiguedad
0. Salir
Seleccione una opcion:
4
Lista de socios ordenados por nombre:
Socio{dni=12345, nombre=Juan, antiguedad=2}
Socio{dni=987986, nombre=Marta, antiguedad=42}
```

```
1. Alta socio
2. Baja socio
3. Modificar socio
4. Listar socios por nombre
5. Listar socios por antiguedad
0. Salir
Seleccione una opcion:
0
Saliendo...
BUILD SUCCESSFUL (total time: 6 minutes 18 seconds)
```

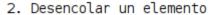
12.17. Implementa la clase Cola genérica utilizando un objeto ArrayList para guardar los elementos.

```
package Main;
2
3 _ import java.util.Scanner;
   * @author juancfm
       public class Hain {
               * 12.17. Implementa la clase Cola genérica utilizando un objeto ArrayList
* para guardar los elementos.
          public static void main(String[] args) {
   Cola cola = new Cola();
   Scanner scanner = new Scanner(source:System.in);
   曱
                   while (true) {
   System.out.println(s: "Menus");
   System.out.println(s: "1. Encolar un elemento");
   System.out.println(s: "2. Desencolar un elemento");
   System.out.println(s: "3. Consultar el primer elemento");
   System.out.println(s: "4. Consultar el tamaño de la cola");
   System.out.println(s: "5. Salir");
                         System.out.print(s: "Ingrese una opción: ");
int opcion = scanner.nextInt();
una línea en blanco para separar las respuestas
                          System.out.println();
                         switch (opcion) {
                                case 3:
    cola consultarPrimero();
                                break;
                                      System.out.println(x: "Saliendo del programa");
System.exit(status:0);
                                     System.out.println(x: "Opción inválida");
                          I una línea en blanco para separar las iteraciones del ciclo while System.out.println();
```



Menu:





- Consultar el primer elemento
- 4. Consultar el tamaño de la cola
- 5. Salir

Ingrese una opción: 1

Ingrese el elemento a encolar: 12 Elemento encolado correctamente



Menu:

- 1. Encolar un elemento
- 2. Desencolar un elemento
- 3. Consultar el primer elemento
- 4. Consultar el tamaño de la cola
- 5. Salir

Ingrese una opción: 4

El tamaño de la cola es: 4



Menu:

- 1. Encolar un elemento
- 2. Desencolar un elemento
- 3. Consultar el primer elemento
- 4. Consultar el tamaño de la cola
- 5. Salir

Ingrese una opción: 2

Elemento desencolado: 12



Menu:

- 1. Encolar un elemento
- 2. Desencolar un elemento
- Consultar el primer elemento
- 4. Consultar el tamaño de la cola
- 5. Salir

Ingrese una opción: 3

El primer elemento es: 15

Menu:

- 1. Encolar un elemento
- 2. Desencolar un elemento
- 3. Consultar el primer elemento
- 4. Consultar el tamaño de la cola
- 5. Salir

Ingrese una opción: 5

Saliendo del programa

BUILD SUCCESSFUL (total time: 2 minutes 0 seconds)

12.18. Implementa la clase Pila genérica utilizando un objeto ArrayList para guardar los elementos.

```
import java.util.Scanner;
5
6
7
8
9
10
11
12
13
14
15
16
                    * @author juancfm
                    public class Main {
                             /*
* 12.18. Implementa la clase Pila genérica utilizando un objeto ArrayList
* para guardar los elementos.
                                     lic static void main(String[] args) {
Pila mmPila = new Pila();
Scanner sc = new Scanner(mare System.in);
18
19
20
21
22
                                       int opcion;
                                               {
System.out.println(= "NEXENSE MENU NEXENSE");
System.out.println(= "1. Apilar elemento");
System.out.println(= "2. Desapilar elemento");
System.out.println(= "3. Consultar tope de la pila");
System.out.println(= "4. Ver tamaño de la pila");
System.out.println(= "5. Verificar si la pila está vacía");
System.out.println(= "0. Salir");
System.out.println(= "Ingrese una opción: ");
opcion = sc.nextInt();
23
25
26
27
28
29
30
32
33
34
35
                                               seitch (opcion) {
  case 1:
    System.out.print(= "Ingrese el elemento a apilar: ");
    int elemento = sc.nextInt();
    miPila.apilar(elemento);
    System.out.println(= "Elemento apilado correctamente");
    hraak:
36
37
38
39
40
41
42
                                                                 miPila.desapilar();
break;
43
44
45
46
47
48
49
50
51
52
53
54
55
                                                                 miPila.consultarTope();
break:
                                                        case 4:
    int tamano = miPila.tamano();
    System.out.println("Tamaño de la pila: " + tamano);
                                                         case 5:
    if (miPila.estaVacia()) {
        System.out.println(= "La pila está vacía");
    } else {
                                                                          System.out.println(= "La pila no está vacía");
56
57
58
59
60
61
62
63
64
65
66
                                                                 System.out.println(: "Saliendo del programa...");
break;
                                                                System.out.println(= "Opción inválida");
break;
68
                                      System.out.println();
} while (opcion != 0);
69
70
71
72
73
                                       sc.close();
```

```
| package Main;
| import java.util.ArrayList;
| wanthor java.util.ArrayList;
| public class Pila {
| private ArrayList<| telementos;
| public Pila() {
| elementos = new ArrayList<|();
| public void apilar(int elemento) {
| elementos.add(elemento);
| public void desapilar() {
| if (elementos.isEmpty()) {
| System.out.println(elementos.remove(elementos.size() - 1));
| }
| public void consultarTope() {
| if (elementos.isEmpty()) {
| System.out.println(elementos.get(elementos.size() - 1));
| }
| System.out.println(elementos.get(elementos.size() - 1));
| }
| public void consultarTope() {
| if (elementos.isEmpty()) {
| System.out.println(elementos.get(elementos.size() - 1));
| }
| public int tamano() {
| return elementos.size();
| }
| public boolean estaVacia() {
| return elementos.isEmpty();
| }
| }
| public boolean estaVacia() {
| return elementos.isEmpty();
| }
| }
| Public boolean estaVacia() {
| return elementos.isEmpty();
| }
| }
| Public boolean estaVacia() {
| return elementos.isEmpty();
| }
| Public boolean estaVacia() {
| return elementos.isEmpty();
| Public boolean estaVacia() {
| Public boolean estaVac
```

```
run:
****** MENÚ ******

1. Apilar elemento
2. Desapilar elemento
3. Consultar tope de la pila
4. Ver tamaño de la pila
5. Verificar si la pila está vacía
0. Salir
Ingrese una opción: 1
Ingrese el elemento a apilar: 12
Elemento apilado correctamente
```

```
***** MENÚ *****

1. Apilar elemento
2. Desapilar elemento
3. Consultar tope de la pila
4. Ver tamaño de la pila
5. Verificar si la pila está vacía
0. Salir
Ingrese una opción: 2
14

***** MENÚ *****
```

```
***** MENÚ ****

1. Apilar elemento
2. Desapilar elemento
3. Consultar tope de la pila
4. Ver tamaño de la pila
5. Verificar si la pila está vacía
0. Salir
Ingrese una opción: 3
13

***** MENÚ *****
```

```
***** MENÚ ****

1. Apilar elemento
2. Desapilar elemento
3. Consultar tope de la pila
4. Ver tamaño de la pila
5. Verificar si la pila está vacía
0. Salir
Ingrese una opción: 4
Tamaño de la pila: 2

***** MENÚ *****
```

```
***** MENÚ *****

1. Apilar elemento
2. Desapilar elemento
3. Consultar tope de la pila
4. Ver tamaño de la pila
5. Verificar si la pila está vacía
0. Salir
Ingrese una opción: 5
La pila no está vacía

***** MENÚ *****
```

```
***** MENÚ *****

1. Apilar elemento
2. Desapilar elemento
3. Consultar tope de la pila
4. Ver tamaño de la pila
5. Verificar si la pila está vacía
0. Salir
Ingrese una opción: 0
Saliendo del programa...

BUILD SUCCESSFUL (total time: 1 minute 0 seconds)
```

12.19. Escribe un programa donde se introduzca por consola una frase que conste exclusiva mente de palabras separadas por espacios. Las palabras de la frase se almacenarán en una lista. Finalmente, se mostrarán por pantalla las palabras que estén repetidas y, a continuación, las que no lo estén.

12.22. Introduce por teclado, hasta que se introduzca « fin », una serie de nombres, que se inser tarán en una colección, de forma que se conserve el orden de inserción y que no pue dan repetirse. Al final, la colección se mostrará por pantalla.

| 12.23. R | epite la | Activida | d de a <u>r</u> | plicación | 12.22 | de | forma | que s | e inserten | los | nombre | es man | te n | iendo |
|----------|----------|----------|-----------------|-----------|-------|----|-------|-------|------------|-----|--------|--------|------|-------|
| el orden | alfabéti | ico . | | | | | | | | | | | | |