

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

Actividades.....	1
Desarrollo.....	2
Actividades de comprobación.....	2
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:.....	3
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 Aplicación.....	5
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.....	5
12.17. Implementa la clase Cola genérica utilizando un objeto ArrayList para guardar los elementos.....	9
12.18. Implementa la clase Pila genérica utilizando un objeto ArrayList para guardar los elementos.....	11

- 12.19. Escribe un programa donde se introduzca por consola una frase que conste exclusivamente 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.....14
- 12.22. Introduce por teclado, hasta que se introduzca « fin », una serie de nombres, que se insertarán en una colección, de forma que se conserve el orden de inserción y que no puedan repetirse. Al final, la colección se mostrará por pantalla.....15

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.

[Volver al índice](#)

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.

[Volver al índice](#)

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.

[Volver al índice](#)

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 () .

[Volver al índice](#)

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.

```

1 package Main;
2
3
4 import java.util.Scanner;
5
6
7 /**
8  * @author juancfm
9  */
10 public class Main {
11     /**
12      * 12.16. Implementa una aplicación que gestione los socios de un club
13      * usando la clase Socio implementada en la Actividad resuelta 12.11.
14      * En particular, se deberán ofrecer las opciones de alta, baja y
15      * modificación de los datos de un socio.
16      * Además, se listarán los socios por nombre o por antigüedad en el club.
17      */
18
19     public static void main(String[] args) {
20         Club club = new Club();
21         try (Scanner scanner = new Scanner(System.in)) {
22             int opcion;
23
24             do {
25                 System.out.println("1. Alta socio");
26                 System.out.println("2. Baja socio");
27                 System.out.println("3. Modificar socio");
28                 System.out.println("4. Listar socios por nombre");
29                 System.out.println("5. Listar socios por antigüedad");
30                 System.out.println("0. Salir");
31                 System.out.print("Seleccione una opcion: \n");
32
33                 opcion = scanner.nextInt();
34                 scanner.nextLine(); // Consumir el salto de línea
35
36                 switch (opcion) {
37                     case 1:
38                         club.altaSocio(socio: altaDatosCompletos(scanner));
39                         break;
40                     case 2:
41                         club.bajaSocio(socio: bajaSocio(scanner));
42                         break;
43                     case 3:
44                         Socio socio = updateSocio(scanner);
45                         int index = club.socios.indexOf(socio);
46                         if (index >= 0) {
47                             socio = club.socios.get(index);
48                             System.out.print("Nuevo nombre: ");
49                             String nombre = scanner.nextLine();
50                             socio.nombre = nombre;
51                             club.modificarSocio(socio);
52                         }
53                     }
54                 }
55             } while (opcion != 0);
56         }
57     }
58 }

```

```

1 package Main;
2
3 import java.io.Serializable;
4 import java.time.LocalDate;
5 import java.time.format.DateTimeFormatter;
6 import java.time.temporal.ChronoUnit;
7
8 public class Socio implements Comparable<Socio>, Serializable {
9
10     String dni;
11     String nombre;
12     LocalDate fechaAlta;
13
14     public Socio(String dni, String nombre, String alta) {
15         this.dni = dni;
16         this.nombre = nombre;
17         DateTimeFormatter f = DateTimeFormatter.ofPattern("dd/MM/yyyy");
18         this.fechaAlta = LocalDate.parse(alta, f);
19     }
20
21     public Socio(String dni) {
22         this.dni = dni;
23     }
24
25     int antigüedad() {
26         return (int) fechaAlta.until(LocalDate.now(),
27                                     ChronoUnit.YEARS);
28     }
29
30     @Override
31     public int compareTo(Socio o) {
32         return dni.compareTo(o.dni);
33     }
34
35     @Override
36     public boolean equals(Object o) {
37         return dni.equals(((Socio) o).dni);
38     }
39
40     @Override
41     public String toString() {
42         return "Socio{" + "dni=" + dni + ", nombre=" + nombre
43             + ", antigüedad=" + antigüedad() + "}\n";
44     }
45 }

```

Actividades de la Unidad 11: Colecciones





```
55         case 4:
56             System.out.println("Lista de socios ordenados por "
57                                 + "nombre:");
58             club.listarSociosPorNombre();
59             break;
60         case 5:
61             System.out.println("Lista de socios ordenados por "
62                                 + "antigüedad en el club:");
63             club.listarSociosPorAntigüedad();
64             break;
65         case 0:
66             System.out.println("Saliendo...");
67             break;
68         default:
69             System.out.println("Opcion invalida.");
70     }
71     System.out.println();
72 } while (opcion != 0);
73 }
74
75 }
76
77 public static Socio altaDatosCompleto(Scanner scanner){
78     System.out.print("\nDNI: ");
79     String dni = scanner.nextLine();
80     System.out.print("Nombre: ");
81     String nombre = scanner.nextLine();
82     System.out.print("Fecha de alta (dd/MM/yyyy): ");
83     String fechaAlta = scanner.nextLine();
84     Socio socio = new Socio(dni, nombre, alta.fechaAlta);
85
86     return socio;
87 }
88
89 public static Socio datosDni(Scanner scanner){
90     String dni = scanner.nextLine();
91
92     return new Socio(dni);
93 }
94
95 public static Socio bajaSocio(Scanner scanner){
96     System.out.print("\nDNI del socio a dar de baja: ");
97
98     return datosDni(scanner);
99 }
100
101 public static Socio updateSocio(Scanner scanner){
102     System.out.print("\nDNI del socio a modificar: ");
103
104     return datosDni(scanner);
105 }
```

```
1 package Main;
2
3 import java.util.ArrayList;
4 import java.util.Collections;
5 import java.util.Comparator;
6 import java.util.List;
7
8
9 /**
10  *
11  * @author juancfm
12  */
13 public class Club {
14
15     public List<Socio> socios;
16
17     public Club() {
18         this.socios = new ArrayList<>();
19     }
20
21     public void altaSocio(Socio socio) {
22         socios.add(socio);
23     }
24
25     public void bajaSocio(Socio socio) {
26         socios.remove(socio);
27     }
28
29     public void modificarSocio(Socio socio) {
30         int index = socios.indexOf(socio);
31         if (index >= 0) {
32             socios.set(index, socio);
33         }
34     }
35
36     public void listarSociosPorNombre() {
37         Collections.sort(socios, new Comparator<Socio>() {
38             @Override
39             public int compare(Socio o1, Socio o2) {
40                 return o1.nombre.compareTo(o2.nombre);
41             }
42         });
43         for (Socio socio : socios) {
44             System.out.println(socio);
45         }
46     }
47
48     public void listarSociosPorAntigüedad() {
49         Collections.sort(socios, new Comparator<Socio>() {
50             @Override
51             public int compare(Socio o1, Socio o2) {
52                 return o1.antigüedad() - o2.antigüedad();
53             }
54         });
55         for (Socio socio : socios) {
56             System.out.println(socio);
57         }
58     }
59 }
60
61 }
```

```
run:
1. Alta socio
2. Baja socio
3. Modificar socio
4. Listar socios por nombre
5. Listar socios por antigüedad
0. Salir
Seleccione una opcion:
1

DNI: 12345
Nombre: Juan
Fecha de alta (dd/MM/yyyy): 13/12/2020
```





Actividades de la Unidad 11: Colecciones



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

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





Socio{dni=2123344, nombre=Pedro, antigüedad=1}
```



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

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

Socio{dni=987986, nombre=María, antigüedad=42}
```



```
1. Alta socio
2. Baja socio
3. Modificar socio
4. Listar socios por nombre
5. Listar socios por antigüedad
0. Salir
Seleccione una opción:
3

DNI del socio a modificar: 987986
Nuevo nombre: Marta
```

Actividades de la Unidad 11: Colecciones

```
1. Alta socio
2. Baja socio
3. Modificar socio
4. Listar socios por nombre
5. Listar socios por antigüedad
0. Salir
Seleccione una opción:
2

DNI del socio a dar de baja: 2123344
```

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

Socio{dni=987986, nombre=Marta, antigüedad=42}
```

```
1. Alta socio
2. Baja socio
3. Modificar socio
4. Listar socios por nombre
5. Listar socios por antigüedad
0. Salir
Seleccione una opción:
0
Saliendo...

BUILD SUCCESSFUL (total time: 6 minutes 18 seconds)
```

[Volver al índice](#)

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

```

1 package Main;
2
3 import java.util.Scanner;
4
5 /**
6  * @author juancf
7  */
8 public class Main {
9     /**
10      * 12.17. Implementa la clase Cola genérica utilizando un objeto ArrayList
11      * para guardar los elementos.
12      */
13
14     public static void main(String[] args) {
15         Cola cola = new Cola();
16         Scanner scanner = new Scanner(System.in);
17
18         while (true) {
19             System.out.println("Menu:");
20             System.out.println("1. Encolar un elemento");
21             System.out.println("2. Desencolar un elemento");
22             System.out.println("3. Consultar el primer elemento");
23             System.out.println("4. Consultar el tamaño de la cola");
24             System.out.println("5. Salir");
25
26             System.out.print("Ingrese una opción: ");
27             int opcion = scanner.nextInt();
28
29             // imprimir una línea en blanco para separar las respuestas
30             System.out.println();
31
32             switch (opcion) {
33                 case 1:
34                     System.out.print("Ingrese el elemento a encolar: ");
35                     int elemento = scanner.nextInt();
36                     cola.encolar(elemento);
37                     System.out.println("Elemento encolado correctamente");
38                     break;
39                 case 2:
40                     cola.desencolar();
41                     break;
42                 case 3:
43                     cola.consultarPrimer();
44                     break;
45                 case 4:
46                     System.out.println("El tamaño de la cola es: "
47                                     + cola.tamano());
48                     break;
49                 case 5:
50                     System.out.println("Saliendo del programa");
51                     System.exit(0);
52                     break;
53                 default:
54                     System.out.println("Opción inválida");
55                     break;
56             }
57
58             // imprimir una línea en blanco para separar las iteraciones del ciclo while
59             System.out.println();
60         }
61     }
62 }

```

```

1 package Main;
2
3 import java.util.ArrayList;
4
5 /**
6  * @author juancf
7  */
8 public class Cola {
9     private ArrayList<Integer> elementos;
10
11     public Cola() {
12         elementos = new ArrayList<>();
13     }
14
15     public void encolar(int elemento) {
16         elementos.add(elemento);
17     }
18
19     public void desencolar() {
20         if (elementos.isEmpty()) {
21             System.out.println("Cola Vacía");
22         }
23         System.out.println("Elemento desencolado: " + elementos.remove(0));
24     }
25
26     public void consultarPrimer() {
27         if (elementos.isEmpty()) {
28             System.out.println("Cola Vacía");
29         }
30         System.out.println("El primer elemento es: " + elementos.get(0));
31     }
32
33     public int tamano() {
34         return elementos.size();
35     }
36
37     public boolean estaVacía() {
38         return elementos.isEmpty();
39     }
40 }

```

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: 1

Ingrese el elemento a encolar: 12
Elemento encolado correctamente

Actividades de la Unidad 11: Colecciones



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
3. 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)

[Volver al índice](#)

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

```

1 package Main;
2
3 import java.util.Scanner;
4
5 /**
6  * @author juancfr
7  */
8 public class Main {
9     /**
10      * 12.18. Implementa la clase Pila genérica utilizando un objeto ArrayList
11      * para guardar los elementos.
12      */
13
14     public static void main(String[] args) {
15         Pila miPila = new Pila();
16         Scanner sc = new Scanner(System.in);
17
18         int opcion;
19         do {
20             System.out.println("===== MENÚ =====");
21             System.out.println("1. Apilar elemento");
22             System.out.println("2. Desapilar elemento");
23             System.out.println("3. Consultar tope de la pila");
24             System.out.println("4. Ver tamaño de la pila");
25             System.out.println("5. Verificar si la pila está vacía");
26             System.out.println("0. Salir");
27             System.out.print("Ingrese una opción: ");
28             opcion = sc.nextInt();
29
30             switch (opcion) {
31                 case 1:
32                     System.out.print("Ingrese el elemento a apilar: ");
33                     int elemento = sc.nextInt();
34                     miPila.apilar(elemento);
35                     System.out.println("Elemento apilado correctamente");
36                     break;
37                 case 2:
38                     miPila.desapilar();
39                     break;
40                 case 3:
41                     miPila.consultarTope();
42                     break;
43                 case 4:
44                     int tamaño = miPila.tamaño();
45                     System.out.println("Tamaño de la pila: " + tamaño);
46                     break;
47                 case 5:
48                     if (miPila.estaVacía()) {
49                         System.out.println("La pila está vacía");
50                     } else {
51                         System.out.println("La pila no está vacía");
52                     }
53                     break;
54                 case 0:
55                     System.out.println("Saliendo del programa...");
56                     break;
57                 default:
58                     System.out.println("Opción inválida");
59                     break;
60             }
61
62             System.out.println();
63         } while (opcion != 0);
64
65         sc.close();
66     }
67 }

```

```

1 package Main;
2
3 import java.util.ArrayList;
4
5 /**
6  * @author juancfr
7  */
8 public class Pila {
9     private ArrayList<Integer> elementos;
10
11     public Pila() {
12         elementos = new ArrayList<>();
13     }
14
15     public void apilar(int elemento) {
16         elementos.add(elemento);
17     }
18
19     public void desapilar() {
20         if (elementos.isEmpty()) {
21             System.out.println("Pila Vacía");
22         }
23         System.out.println("Elemento eliminado: " + elementos.remove(elementos.size() - 1));
24     }
25
26     public void consultarTope() {
27         if (elementos.isEmpty()) {
28             System.out.println("Pila Vacía");
29         }
30         System.out.println("Elemento en el tope: " + elementos.get(elementos.size() - 1));
31     }
32
33     public int tamaño() {
34         return elementos.size();
35     }
36
37     public boolean estaVacía() {
38         return elementos.isEmpty();
39     }
40 }

```

Actividades de la Unidad 11: Colecciones

```
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 *****
```


Actividades de la Unidad 11: Colecciones

```
**** 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)
|
```

[Volver al índice](#)

12.19. Escribe un programa donde se introduzca por consola una frase que conste exclusivamente 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.

[Volver al índice](#)

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

[Volver al índice](#)

Actividades de la Unidad 11: Colecciones

12.23. Repite la Actividad de aplicación 12.22 de forma que se inserten los nombres manteniendo el orden alfabético .

[Volver al índice](#)