FAMILIA PROFESIONAL:	**Informática y Comunicaciones**
CICLOS FORMATIVOS:	**Desarrollo de Aplicaciones Multiplataforma, Desarrollo de Aplicaciones Web**
MÓDULO:	**Programación**

UNIDAD 8: COLECCIONES DE OBJETOS

ACTIVIDADES

AUTORES: **Fernando Rodríguez Alonso**
Sonia Pasamar Franco

ESTRUCTURAS DE DATOS DINÁMICAS

ACTIVIDAD 1x01

Codifica un programa principal **`main`** que:

- Inserte 15 números enteros aleatorios comprendidos entre 100 y 200 en una lista enlazada.
- Visualice en consola esta lista enlazada.
- Calcule y visualice en consola la suma de los elementos de la lista enlazada.
- Calcule y visualice en consola la media aritmética de los elementos de la lista enlazada.

Un ejemplo de ejecución del programa podría ser:

```
LISTA ENLAZADA:
[101, 140, 165, 101, 158, 130, 180, 198, 100, 188, 195, 156, 164, 168, 141]
Se han consultado 15 nodos de la lista.
Suma: 2285
Media: 152.33
```

ACTIVIDAD 1x02

Codifica un programa principal **`main`** que:

- Lea por teclado 10 caracteres y los inserte en una lista enlazada.
- Visualice en consola esta lista enlazada.
- Cree otra lista enlazada con los mismos caracteres de la primera lista pero en orden inverso.
- Visualice en consola esta segunda lista enlazada.

Un ejemplo de ejecución del programa podría ser:

```
¿Carácter 0? A
¿Carácter 1? B
¿Carácter 2? C
¿Carácter 3? D
¿Carácter 4? E
¿Carácter 5? F
¿Carácter 6? G
¿Carácter 7? H
¿Carácter 8? I
¿Carácter 9? J
LISTA ENLAZADA 1:
[A, B, C, D, E, F, G, H, I, J]
Se han consultado 10 nodos de la lista.
LISTA ENLAZADA 2:
[J, I, H, G, F, E, D, C, B, A]
Se han consultado 10 nodos de la lista.
```

ACTIVIDAD 1x03

Codifica un programa principal **`main`** que:

- Lea por teclado una línea de texto.
- Utilice una pila para invertir los caracteres de esta línea de texto y generar otra línea de texto con los mismos caracteres en orden inverso.
- Visualice en consola esta línea de texto invertida.

Un ejemplo de ejecución del programa podría ser:

```
¿Línea de Texto? Hoy hay mucho tráfico en la carretera.
PILA:
[., a, r, e, t, e, r, r, a, c,  , a, l,  , n, e,  ,
 o, c, i, f, á, r, t,  , o, h, c, u, m,  , y, a, h,  , y, o, H]
Se han consultado 38 nodos de la pila.
Línea de Texto Invertida:
.areterrac al ne ocifárt ohcum yah yoH
```

ACTIVIDAD 1x04 (ampliación)

Codifica un programa principal **`main`** que:

- Lea por teclado una línea de texto.
- Utilice una pila para determinar si esta línea de texto es palíndroma o no es palíndroma, ignorando los espacios en blanco y los signos de puntuación.
- Visualice en consola un mensaje indicando si la línea de texto es palíndroma o no es palíndroma.

Un ejemplo de ejecución del programa podría ser:

```
¿Línea de Texto? ¿son robos o sobornos?
PILA FILTRADA:
[s, o, n, r, o, b, o, s, o, s, o, b, o, r, n, o, s]
Se han consultado 17 nodos de la pila.
La línea de texto es palíndroma.
```

Otro ejemplo de ejecución del programa podría ser:

```
¿Línea de Texto? Aquella granja tiene gallinas, cerdos y vacas.
PILA FILTRADA:
[s, a, c, a, v, y, s, o, d, r, e, c, s, a, n, i, l, l, a, g,
 e, n, e, i, t, a, j, n, a, r, g, a, l, l, e, u, q, A]
Se han consultado 38 nodos de la pila.
La línea de texto no es palíndroma.
```

LISTAS DE OBJETOS

ACTIVIDAD 2x01

Codifica un programa principal **`main`** que:

- Lea por teclado una cantidad de números enteros a insertar.
- Genere dicha cantidad de números enteros, los inserte en una colección de la clase **`ArrayList`** (lista vector) y los inserte también en una colección de la clase **`LinkedList`** (lista enlazada).
  Cada número entero a insertar se deberá generar de forma aleatoria entre 10 y 20 (con ambos límites incluidos).
- Recorra los elementos de la lista vector de enteros con un iterador y los visualice en consola.
- Recorra los elementos de la lista vector de enteros por posición y los visualice en consola
- Visualice en consola la lista vector de enteros usando el método `toString`.
- Recorra los elementos de la lista enlazada de enteros con un iterador y los visualice en consola.
- Recorra los elementos de la lista enlazada de enteros por posición y los visualice en consola
- Visualice en consola la lista enlazada de enteros usando el método `toString`.

Un ejemplo de ejecución del programa podría ser:

```
¿Cantidad de Números Enteros? 14
Se van a generar 14 números enteros.
Insertando:
12 * 11 * 17 * 20 * 16 * 20 * 19 * 11 * 12 * 16 * 18 * 10 * 13 * 11 *

Elementos de la Lista Vector de Enteros (con Iterador):
12 11 17 20 16 20 19 11 12 16 18 10 13 11
Elementos de la Lista Vector de Enteros (por Posición):
12 11 17 20 16 20 19 11 12 16 18 10 13 11
Elementos de la Lista Vector de Enteros (Método toString):
[12, 11, 17, 20, 16, 20, 19, 11, 12, 16, 18, 10, 13, 11]

Elementos de la Lista Enlazada de Enteros (con Iterador):
12 11 17 20 16 20 19 11 12 16 18 10 13 11
Elementos de la Lista Enlazada de Enteros (por Posición):
12 11 17 20 16 20 19 11 12 16 18 10 13 11
Elementos de la Lista Enlazada de Enteros (Método toString):
[12, 11, 17, 20, 16, 20, 19, 11, 12, 16, 18, 10, 13, 11]
```

ACTIVIDAD 2x02

Codifica un programa principal **`main`** que:

- Lea por teclado una cantidad de cadenas de caracteres a insertar.
- Genera dicha cantidad de cadenas de caracteres aleatorias a partir de un vector de palabras dado, las inserta en una colección de la clase **`ArrayList`** (lista vector) y las inserta también en una colección de la clase **`LinkedList`** (lista enlazada).
  Cada cadena de caracteres a insertar se deberá seleccionar de forma aleatoria a partir de las siguientes palabras:
  `secuencial, aleatorio, letra, número, frío, calor,`
  `alto, bajo, dibujo, redacción, grande, pequeño, montaña, rio`
- Recorra los elementos de la lista vector de cadenas con un iterador y los visualice en consola.
- Recorra los elementos de la lista vector de cadenas por posición y los visualice en consola
- Visualice en consola la lista vector de cadenas usando el método `toString`.
- Recorra los elementos de la lista enlazada de cadenas con un iterador y los visualice en consola.
- Recorra los elementos de la lista enlazada de cadenas por posición y los visualice en consola
- Visualice en consola la lista enlazada de cadenas usando el método `toString`.

Un ejemplo de ejecución del programa podría ser:

```
¿Cantidad de Cadenas de Caracteres? 14
Se van a generar 14 cadenas de caracteres.
Insertando:
número * calor * alto * bajo * letra * frio * grande * redacción *
frio * alto * secuencial * pequeño * aleatorio * montaña *

Elementos de la Lista Vector de Cadenas (con Iterador):
número calor alto bajo letra frio grande redacción
frio alto secuencial pequeño aleatorio montaña
Elementos de la Lista Vector de Cadenas (por Posición):
número calor alto bajo letra frio grande redacción
frio alto secuencial pequeño aleatorio montaña
Elementos de la Lista Vector de Cadenas (Método toString):
[número, calor, alto, bajo, letra, frio, grande, redacción,
 frio, alto, secuencial, pequeño, aleatorio, montaña]

Elementos de la Lista Enlazada de Cadenas (con Iterador):
número calor alto bajo letra frio grande redacción
frio alto secuencial pequeño aleatorio montaña
Elementos de la Lista Enlazada de Cadenas (por Posición):
número calor alto bajo letra frio grande redacción
frio alto secuencial pequeño aleatorio montaña
Elementos de la Lista Enlazada de Cadenas (Método toString):
[número, calor, alto, bajo, letra, frio, grande, redacción,
 frio, alto, secuencial, pequeño, aleatorio, montaña]
```

ACTIVIDAD 2x03

Codifica una clase **`Libro`** para tratar la información de los diferentes libros de una tienda. De cada uno de ellos se desea guardar: el ISBN, el título, el escritor, el año de publicación, el número de unidades en *stock* y el precio en euros.

El ISBN de cada libro será una cadena de texto identificativa. Cada libro tendrá su propio ISBN, es decir, no podrá haber dos o más libros con el mismo ISBN en la tienda.

Para esta clase **`Libro`**:

- Codifica un constructor que reciba como parámetros el ISBN, el título, el escritor, el año de publicación, el número de unidades y el precio.
- Codifica un método de objeto que devuelva una cadena de texto con el resumen de los atributos del objeto. Para ello, sobrescribe el método **`toString`** que procede de la clase `Object`. El precio se deberá indicar con 2 dígitos decimales.
- Añade los métodos de objeto necesarios, según la lista elegida y las opciones de menú indicadas en el programa principal.

Codifica una clase **`Actividad_2x03`** que incluya un programa principal **`main`**. Este programa gestionará una **lista de libros** (**`ArrayList`** o **`LinkedList`**) mediante el siguiente menú de opciones:

0) Salir del programa.

1) Insertar un libro en la lista.

Leerá por teclado el ISBN, el título, el escritor, el año de publicación, el número de unidades y el precio del libro a insertar.
Si la lista contiene un libro con el ISBN dado, visualizará en consola el mensaje:

```
Ya existe otro libro con ese ISBN en la lista.
```

Si la lista no contiene ningún libro con el ISBN dado, insertará el libro en la lista y visualizará en consola el mensaje:

```
Se ha insertado el libro en la lista.
```

2) Eliminar un libro, por ISBN, de la lista.

Leerá por teclado el ISBN del libro a eliminar.
Si la lista no contiene ningún libro con el ISBN dado, visualizará en consola el mensaje:

```
No existe ningún libro con ese ISBN en la lista.
```

Si la lista contiene un libro con el ISBN dado, eliminará el libro de la lista y visualizará en consola el mensaje:

```
Se ha eliminado el libro de la lista.
```

3) Consultar un libro, por ISBN, de la lista.

Leerá por teclado el ISBN del libro a consultar.
Si la lista no contiene ningún libro con el ISBN dado, visualizará en consola el mensaje:

```
No existe ningún libro con ese ISBN en la lista.
```

Si la lista contiene un libro con el ISBN dado, consultará el libro de la lista y visualizará en consola un resumen con los nombres y valores de todos los atributos de dicho libro.
El siguiente ejemplo muestra el libro cuyo ISBN es `3456789012`:

```
Libro [ISBN = 3456789012, Título = The Lord of the Rings,
       Escritor = J.R.R. Tolkien, AñoPublicación = 1955,
       NúmeroUnidades = 25, Precio = 59,90]
```

4) Consultar todos los libros de la lista.

Si la lista no contiene ningún libro, visualizará en consola el mensaje:

```
La lista está vacía.
```

Si la lista contiene uno o más libros:

- Visualizará en consola un listado de todos los libros contenidos en la lista, indicando para cada libro, su posición dentro de la lista y un resumen con los nombres y valores de todos los atributos de dicho libro.
- Visualizará en consola el número de libros consultados de la lista.

Por ejemplo:

```
(0) Libro [ISBN = 0123456789, Título = The Hobbit,
           Escritor = J.R.R. Tolkien, AñoPublicación = 1937,
           NúmeroUnidades = 40, Precio = 19,75]
(1) Libro [ISBN = 3456789012, Título = The Lord of the Rings,
           Escritor = J.R.R. Tolkien, AñoPublicación = 1955,
           NúmeroUnidades = 25, Precio = 59,90]
(2) Libro [ISBN = 6789012345, Título = Dune,
           Escritor = Frank Herbert, AñoPublicación = 1965,
           NúmeroUnidades = 30, Precio = 24,95]
Se han consultado 3 libros de la lista.
```

5) Consultar todos los libros de la lista, en orden por precio descendente.

Si la lista no contiene ningún libro, visualizará en consola el mensaje:

```
La lista está vacía.
```

Si la lista contiene uno o más libros:

- Visualizará en consola un listado de todos los libros contenidos en la lista, indicando para cada libro, su posición dentro de la lista y un resumen con los nombres y valores de todos los atributos de dicho libro. Este listado deberá estar ordenado por el precio del libro de forma descendente.
- Visualizará en consola el número de libros consultados de la lista.

Por ejemplo:

```
(0) Libro [ISBN = 3456789012, Título = The Lord of the Rings,
           Escritor = J.R.R. Tolkien, AñoPublicación = 1955,
           NúmeroUnidades = 25, Precio = 59,90]
(1) Libro [ISBN = 6789012345, Título = Dune,
           Escritor = Frank Herbert, AñoPublicación = 1965,
           NúmeroUnidades = 30, Precio = 24,95]
(2) Libro [ISBN = 0123456789, Título = The Hobbit,
           Escritor = J.R.R. Tolkien, AñoPublicación = 1937,
           NúmeroUnidades = 40, Precio = 19,75]
Se han consultado 3 libros de la lista.
```

6) Consultar varios libros, por escritor, de la lista.

Leerá por teclado el escritor de los libros a consultar.
Si la lista no contiene ningún libro con el escritor dado, visualizará en consola el mensaje:

```
No existe ningún libro con ese escritor en la lista.
```

Si la lista contiene uno o más libros con el escritor dado:

- Visualizará en consola un listado de los libros del escritor contenidos en la lista, indicando para cada libro, un resumen con los nombres y valores de todos los atributos de dicho libro.
- Visualizará en consola el número de libros consultados de la lista.

El siguiente ejemplo muestra los libros cuyo escritor es `J.R.R. Tolkien`:

```
Libro [ISBN = 0123456789, Título = The Hobbit,
       Escritor = J.R.R. Tolkien, AñoPublicación = 1937,
       NúmeroUnidades = 40, Precio = 19,75]
Libro [ISBN = 3456789012, Título = The Lord of the Rings,
       Escritor = J.R.R. Tolkien, AñoPublicación = 1955,
       NúmeroUnidades = 25, Precio = 59,90]
Se han consultado 2 libros de la lista.
```

Este programa principal deberá validar que la opción de menú elegida sea válida (comprendida entre 0 y 6). Si no lo es, visualizará en consola el mensaje:

```
La opción de menú debe estar comprendida entre 0 y 6.
```

CONJUNTOS DE OBJETOS

ACTIVIDAD 3x01

Codifica un programa principal **`main`** que:

- Lea por teclado una cantidad de números enteros a insertar.
- Genere dicha cantidad de números enteros, los inserte en una colección de la clase **`HashSet`** (conjunto *hash*), los inserte también en una colección de la clase **`LinkedHashSet`** (conjunto *hash* enlazado), y por último los inserte en una colección de la clase **`TreeSet`** (conjunto árbol).
  Cada número entero a insertar se deberá generar de forma aleatoria entre 10 y 20 (con ambos límites incluidos).
- Recorra los elementos del conjunto *hash* de enteros con un iterador y los visualice en consola.
- Visualice en consola el conjunto *hash* de enteros usando el método `toString`.
- Recorra los elementos del conjunto *hash* enlazado de enteros con un iterador y los visualice en consola.
- Visualice en consola el conjunto *hash* enlazado de enteros usando el método `toString`.
- Recorra los elementos del conjunto árbol de enteros con un iterador y los visualice en consola.
- Visualice en consola el conjunto árbol de enteros usando el método `toString`.

Un ejemplo de ejecución del programa podría ser:

```
¿Cantidad de Números Enteros? 14
Se van a generar 14 números enteros.
Insertando:
15 * 18 * 20 * 17 * 10 * 16 * 15 * 16 * 19 * 14 * 18 * 13 * 10 * 11 *

SIN ORDEN
Elementos del Conjunto Hash de Enteros (con Iterador):
16 17 18 19 20 10 11 13 14 15
Elementos del Conjunto Hash de Enteros (Método toString):
[16, 17, 18, 19, 20, 10, 11, 13, 14, 15]

POR ORDEN DE INSERCIÓN
Elementos del Conjunto Hash Enlazado de Enteros (con Iterador):
15 18 20 17 10 16 19 14 13 11
Elementos del Conjunto Hash Enlazado de Enteros (Método toString):
[15, 18, 20, 17, 10, 16, 19, 14, 13, 11]

POR ORDEN NUMÉRICO ASCENDENTE
Elementos del Conjunto Árbol de Enteros (con Iterador):
10 11 13 14 15 16 17 18 19 20
Elementos del Conjunto Árbol de Enteros (Método toString):
[10, 11, 13, 14, 15, 16, 17, 18, 19, 20]
```

ACTIVIDAD 3x02

Codifica un programa principal **`main`** que:

- Lea por teclado una cantidad de cadenas de caracteres a insertar.
- Genere dicha cantidad de cadenas de caracteres aleatorias a partir de un vector de palabras dado, las inserte en una colección de la clase **`HashSet`** (conjunto *hash*), las inserte también en una colección de la clase **`LinkedHashSet`** (conjunto *hash* enlazado), y por último las inserte en una colección de la clase **`TreeSet`** (conjunto árbol).
  Cada cadena de caracteres a insertar se deberá seleccionar de forma aleatoria a partir de las siguientes palabras:
  `secuencial, aleatorio, letra, número, frío, calor,`
  `alto, bajo, dibujo, redacción, grande, pequeño, montaña, rio`
- Recorra los elementos del conjunto *hash* de cadenas con un iterador y los visualice en consola.
- Visualice en consola el conjunto *hash* de cadenas usando el método `toString`.
- Recorra los elementos del conjunto *hash* enlazado de cadenas con un iterador y los visualice en consola.
- Visualice en consola el conjunto *hash* enlazado de cadenas usando el método `toString`.
- Recorra los elementos del conjunto árbol de cadenas con un iterador y los visualice en consola.
- Visualice en consola el conjunto árbol de cadenas usando el método `toString`.

Un ejemplo de ejecución del programa podría ser:

```
¿Cantidad de Cadenas de Caracteres? 14
Se van a generar 14 cadenas de caracteres.
Insertando:
pequeño * letra * redacción * aleatorio * pequeño * bajo * redacción *
número * bajo * montaña * grande * secuencial * rio * redacción *

SIN ORDEN
Elementos del Conjunto Hash de Cadenas (con Iterador):
redacción pequeño aleatorio rio bajo número letra montaña grande secuencial
Elementos del Conjunto Hash de Cadenas (Método toString):
[redacción, pequeño, aleatorio, rio, bajo,
 número, letra, montaña, grande, secuencial]

POR ORDEN DE INSERCIÓN
Elementos del Conjunto Hash Enlazado de Cadenas (con Iterador):
pequeño letra redacción aleatorio bajo número montaña grande secuencial rio
Elementos del Conjunto Hash Enlazado de Cadenas (Método toString):
[pequeño, letra, redacción, aleatorio, bajo,
 número, montaña, grande, secuencial, rio]
```

```
POR ORDEN ALFABÉTICO ASCENDENTE
Elementos del Conjunto Árbol de Cadenas (con Iterador):
aleatorio bajo grande letra montaña número pequeño redacción rio secuencial
Elementos del Conjunto Árbol de Cadenas (Método toString):
[aleatorio, bajo, grande, letra, montaña,
 número, pequeño, redacción, rio, secuencial]
```

ACTIVIDAD 3x03 (ampliación)

Codifica un programa principal **`main`** que:

- Lea por teclado de forma repetitiva números enteros y los inserte en una colección de la clase **`TreeSet`** (conjunto árbol). Este proceso deberá repetirse hasta que se introduzca el número `0`, el cual no deberá insertarse en el conjunto árbol.
- Recorra los elementos del conjunto árbol de enteros con un bucle *for-each* y los visualice en consola.
- Lea por teclado una cantidad de números enteros *N* que deberá ser mayor que 0 y deberá ser menor o igual que la cantidad de números enteros almacenados en el conjunto árbol. Si esta cantidad de números enteros no es válida, visualizará en consola un mensaje de error y leerá por teclado otra cantidad de números enteros. Este proceso deberá repetirse mientras la cantidad de números enteros leída no sea válida.
- Visualice en consola los *N* primeros números enteros del conjunto árbol, en orden numérico ascendente.

Un ejemplo de ejecución del programa podría ser:

```
¿Número Entero (0 para terminar)? 52
¿Número Entero (0 para terminar)? 31
¿Número Entero (0 para terminar)? 67
¿Número Entero (0 para terminar)? 49
¿Número Entero (0 para terminar)? 98
¿Número Entero (0 para terminar)? 11
¿Número Entero (0 para terminar)? 11
¿Número Entero (0 para terminar)? 26
¿Número Entero (0 para terminar)? 75
¿Número Entero (0 para terminar)? 80
¿Número Entero (0 para terminar)? 0

Elementos del Conjunto Árbol de Enteros (Bucle for-each):
11 26 31 49 52 67 75 80 98

¿Cantidad de Números Enteros? 0
La cantidad de números enteros debe estar comprendida entre 1 y 9.
¿Cantidad de Números Enteros? 10
La cantidad de números enteros debe estar comprendida entre 1 y 9.
¿Cantidad de Números Enteros? 5

5 Primeros Elementos del Conjunto Árbol de Enteros:
11 26 31 49 52
```

ACTIVIDAD 3x04 (ampliación)

Codifica un programa principal **`main`** que:

- Lea por teclado de forma repetitiva cadenas de caracteres y las inserte en una colección de la clase **`TreeSet`** (conjunto árbol). Este proceso deberá repetirse hasta que se introduzca la cadena ***, la cual no deberá insertarse en el conjunto árbol.
- Recorra los elementos del conjunto árbol de cadenas con un bucle *for-each* y los visualice en consola.
- Lea por teclado una cantidad de cadenas de caracteres *N* que deberá ser mayor que 0 y deberá ser menor o igual que la cantidad de cadenas de caracteres almacenadas en el conjunto árbol. Si esta cantidad de cadenas de caracteres no es válida, visualizará en consola un mensaje de error y leerá por teclado otra cantidad de cadenas de caracteres. Este proceso deberá repetirse mientras la cantidad de cadenas de caracteres leída no sea válida.
- Visualice en consola las *N* primeras cadenas de caracteres del conjunto árbol, en orden alfabético ascendente.

Un ejemplo de ejecución del programa podría ser:

```
¿Cadena de Caracteres (*** para terminar)? rojo
¿Cadena de Caracteres (*** para terminar)? verde
¿Cadena de Caracteres (*** para terminar)? azul
¿Cadena de Caracteres (*** para terminar)? triángulo
¿Cadena de Caracteres (*** para terminar)? cuadrado
¿Cadena de Caracteres (*** para terminar)? pentágono
¿Cadena de Caracteres (*** para terminar)? círculo
¿Cadena de Caracteres (*** para terminar)? cero
¿Cadena de Caracteres (*** para terminar)? uno
¿Cadena de Caracteres (*** para terminar)? dos
¿Cadena de Caracteres (*** para terminar)? masculino
¿Cadena de Caracteres (*** para terminar)? femenino
¿Cadena de Caracteres (*** para terminar)? neutro
¿Cadena de Caracteres (*** para terminar)? ***

Elementos del Conjunto Árbol de Cadenas (Bucle for-each):
azul cero cuadrado círculo dos femenino masculino neutro pentágono rojo
triángulo uno verde

¿Cantidad de Cadenas de Caracteres? 0
La cantidad de cadenas de caracteres debe estar comprendida entre 1 y 13.
¿Cantidad de Cadenas de Caracteres? 14
La cantidad de cadenas de caracteres debe estar comprendida entre 1 y 13.
¿Cantidad de Cadenas de Caracteres? 7

7 Primeros Elementos del Conjunto Árbol de Cadenas:
azul cero cuadrado círculo dos femenino masculino
```

ACTIVIDAD 3x05

Codifica una clase **`Fecha`** para tratar la información de diferentes fechas, guardando el día, el mes y el año.

Para esta clase **`Fecha`**:

- Codifica un constructor que reciba como parámetros el día, el mes y el año.
- Codifica un método de objeto que devuelva una cadena de texto con el resumen de los atributos del objeto. Para ello, sobrescribe el método **`toString`** que procede de la clase `Object`. La fecha se deberá indicar con el formato `DD/MM/AAAA`.

Codifica una clase **`Cliente`** para tratar la información de los diferentes clientes de una entidad bancaria. De cada una de ellos se desea guardar: el DNI, el nombre, la fecha de nacimiento, el domicilio, el correo electrónico y el saldo total de sus cuentas en euros.

El DNI de cada cliente será una cadena de texto identificativa. Cada cliente tendrá su propio DNI, es decir, no podrá haber dos o más clientes con el mismo DNI en la entidad bancaria.

Para esta clase **`Cliente`**:

- Codifica un constructor que reciba como parámetros el DNI, el nombre, la fecha de nacimiento, el domicilio, el correo electrónico y el saldo total.
- Codifica un método de objeto que devuelva una cadena de texto con el resumen de los atributos del objeto. Para ello, sobrescribe el método **`toString`** que procede de la clase `Object`. El saldo total se deberá indicar con 2 dígitos decimales.
- Añade los métodos de objeto necesarios, según el conjunto elegido y las opciones de menú indicadas en el programa principal.

Codifica una clase **`Actividad_3x05`** que incluya un programa principal **`main`**. Este programa gestionará un **conjunto de clientes** (**`HashSet`**, **`LinkedHashSet`** o **`TreeSet`**) mediante el siguiente menú de opciones:

0) Salir del programa.

1) Insertar un cliente en el conjunto.

Leerá por teclado el DNI, el nombre, la fecha de nacimiento, el domicilio, el correo electrónico y el saldo total del cliente a insertar.
Si el conjunto contiene un cliente con el DNI dado, visualizará en consola el mensaje:

```
Ya existe otro cliente con ese DNI en el conjunto.
```

Si el conjunto no contiene ningún cliente con el DNI dado, insertará el cliente en el conjunto y visualizará en consola el mensaje:

```
Se ha insertado el cliente en el conjunto.
```

2) Eliminar un cliente, por DNI, del conjunto.

Leerá por teclado el DNI del cliente a eliminar.
Si el conjunto no contiene ningún cliente con el DNI dado, visualizará en consola el mensaje:

```
No existe ningún cliente con ese DNI en el conjunto.
```

Si el conjunto contiene un cliente con el DNI dado, eliminará el cliente del conjunto y visualizará en consola el mensaje:

```
Se ha eliminado el cliente del conjunto.
```

3) Consultar un cliente, por DNI, del conjunto.

Leerá por teclado el DNI del cliente a consultar.
Si el conjunto no contiene ningún cliente con el DNI dado, visualizará en consola el mensaje:

```
No existe ningún cliente con ese DNI en el conjunto.
```

Si el conjunto contiene un cliente con el DNI dado, consultará el cliente del conjunto y visualizará en consola un resumen con los nombres y valores de todos los atributos de dicho cliente.
El siguiente ejemplo muestra el cliente cuyo DNI es 33336666C:

```
Cliente [DNI = 33336666C, Nombre = Jorge Ruiz,
         FechaNacimiento = 05/10/1987, Domicilio = Teruel,
         Correo = jr@hotmail.com, SaldoTotal = 29165,28]
```

4) Consultar todos los clientes del conjunto, en orden por DNI ascendente.

Si el conjunto no contiene ningún cliente, visualizará en consola el mensaje:

```
El conjunto está vacío.
```

Si el conjunto contiene uno o más clientes:

- Visualizará en consola un listado de todos los clientes contenidos en el conjunto, indicando para cada cliente, un resumen con los nombres y valores de todos los atributos de dicho cliente. Este listado deberá estar ordenado por el DNI del cliente de forma ascendente.
- Visualizará en consola el número de clientes consultados del conjunto.

Por ejemplo:

```
Cliente [DNI = 11114444B, Nombre = Elena García,
         FechaNacimiento = 12/06/2002, Domicilio = Huesca,
         Correo = eg@gmail.com, SaldoTotal = 43587,50]
Cliente [DNI = 22225555A, Nombre = Pablo Moreno,
         FechaNacimiento = 23/02/1974, Domicilio = Zaragoza,
         Correo = pm@yahoo.com, SaldoTotal = 32108,69]
Cliente [DNI = 33336666C, Nombre = Jorge Ruiz,
         FechaNacimiento = 05/10/1987, Domicilio = Teruel,
         Correo = jr@hotmail.com, SaldoTotal = 29165,28]
Se han consultado 3 clientes del conjunto.
```

5) Consultar todos los clientes del conjunto, en orden por nombre ascendente.

Si el conjunto no contiene ningún cliente, visualizará en consola el mensaje:

```
El conjunto está vacío.
```

Si el conjunto contiene uno o más clientes:

- Visualizará en consola un listado de todos los clientes contenidos en el conjunto, indicando para cada cliente, un resumen con los nombres y valores de todos los atributos de dicho cliente. Este listado deberá estar ordenado por el nombre del cliente de forma ascendente.
- Visualizará en consola el número de clientes consultados del conjunto.
- Puede haber varios clientes con el mismo nombre en el conjunto.

Por ejemplo:

```
Cliente [DNI = 11114444B, Nombre = Elena García,
         FechaNacimiento = 12/06/2002, Domicilio = Huesca,
         Correo = eg@gmail.com, SaldoTotal = 43587,50]
Cliente [DNI = 33336666C, Nombre = Jorge Ruiz,
         FechaNacimiento = 05/10/1987, Domicilio = Teruel,
         Correo = jr@hotmail.com, SaldoTotal = 29165,28]
Cliente [DNI = 22225555A, Nombre = Pablo Moreno,
         FechaNacimiento = 23/02/1974, Domicilio = Zaragoza,
         Correo = pm@yahoo.com, SaldoTotal = 32108,69]
Se han consultado 3 clientes del conjunto.
```

6) Consultar todos los clientes del conjunto, en orden por fecha de nacimiento ascendente.

Si el conjunto no contiene ningún cliente, visualizará en consola el mensaje:

```
El conjunto está vacío.
```

Si el conjunto contiene uno o más clientes:

- Visualizará en consola un listado de todos los clientes contenidos en el conjunto, indicando para cada cliente, un resumen con los nombres y valores de todos los atributos de dicho cliente. Este listado deberá estar ordenado por la fecha de nacimiento del cliente de forma ascendente.
- Visualizará en consola el número de clientes consultados del conjunto.
- Puede haber varios clientes con la misma fecha de nacimiento en el conjunto.

Por ejemplo:

```
Cliente [DNI = 22225555A, Nombre = Pablo Moreno,
         FechaNacimiento = 23/02/1974, Domicilio = Zaragoza,
         Correo = pm@yahoo.com, SaldoTotal = 32108,69]
Cliente [DNI = 33336666C, Nombre = Jorge Ruiz,
         FechaNacimiento = 05/10/1987, Domicilio = Teruel,
         Correo = jr@hotmail.com, SaldoTotal = 29165,28]
Cliente [DNI = 11114444B, Nombre = Elena García,
         FechaNacimiento = 12/06/2002, Domicilio = Huesca,
         Correo = eg@gmail.com, SaldoTotal = 43587,50]
Se han consultado 3 clientes del conjunto.
```

Este programa principal deberá validar que la opción de menú elegida sea válida (comprendida entre 0 y 6). Si no lo es, visualizará en consola el mensaje:

```
La opción de menú debe estar comprendida entre 0 y 6.
```

ACTIVIDAD 3x06 (ampliación)

Codifica una clase **`Pelicula`** para tratar la información de las diferentes películas de una plataforma de *streaming*. De cada una de ellas se desea guardar: el código, el título, el director, el año de estreno, la duración en minutos y la puntuación otorgada por críticos de cine.

El código de cada película será una cadena de texto identificativa. Cada película tendrá su propio código, es decir, no podrá haber dos o más películas con el mismo código en la plataforma de *streaming*.

Para esta clase **`Pelicula`**:

- Codifica un constructor que reciba como parámetros el código, el título, el director, el año de estreno, la duración y la puntuación.
- Codifica un método de objeto que devuelva una cadena de texto con el resumen de los atributos del objeto. Para ello, sobrescribe el método **`toString`** que procede de la clase `Object`. La puntuación se deberá indicar con 1 dígito decimal.
- Añade los métodos de objeto necesarios, según el conjunto elegido y las opciones de menú indicadas en el programa principal.

Codifica una clase **`Actividad_3x06`** que incluya un programa principal **`main`**. Este programa gestionará un **conjunto de películas** (**`HashSet`**, **`LinkedHashSet`** o **`TreeSet`**) mediante el siguiente menú de opciones:

0) Salir del programa.

1) Insertar una película en el conjunto.

Leerá por teclado el código, el título, el director, el año de estreno, la duración y la puntuación de la película a insertar.
Si el conjunto contiene una película con el código dado, visualizará en consola el mensaje:

```
Ya existe otra película con ese código en el conjunto.
```

Si el conjunto no contiene ninguna película con el código dado, insertará la película en el conjunto y visualizará en consola el mensaje:

```
Se ha insertado la película en el conjunto.
```

2) Consultar todas las películas del conjunto, en orden de inserción.

Si el conjunto no contiene ninguna película, visualizará en consola el mensaje:

```
El conjunto está vacío.
```

Si el conjunto contiene una o más películas:

- Visualizará en consola un listado de todas las películas contenidas en el conjunto, indicando para cada película, un resumen con los nombres y valores de todos los atributos de dicha película. Este listado deberá estar en orden de inserción.
- Visualizará en consola el número de películas consultadas del conjunto.

Por ejemplo:

```
Película [Código = primera, Título = Raiders of the Lost Ark,
          Director = Steven Spielberg, AñoEstreno = 1981,
          Duración = 115, Puntuación = 8,4]
Película [Código = segunda, Título = Jurassic Park,
          Director = Steven Spielberg, AñoEstreno = 1993,
          Duración = 127, Puntuación = 8,1]
Película [Código = tercera, Título = Avatar,
          Director = James Cameron, AñoEstreno = 2009,
          Duración = 162, Puntuación = 7,8]
```

```
Película [Código = cuarta, Título = Titanic,
          Director = James Cameron, AñoEstreno = 1997,
          Duración = 195, Puntuación = 7,8]
Película [Código = quinta, Título = Goodfellas,
          Director = Martin Scorsese, AñoEstreno = 1990,
          Duración = 146, Puntuación = 8,7]
Película [Código = sexta, Título = Pulp Fiction,
          Director = Quentin Tarantino, AñoEstreno = 1994,
          Duración = 154, Puntuación = 8,9]
Película [Código = séptima, Título = Gladiator,
          Director = Ridley Scott, AñoEstreno = 2000,
          Duración = 155, Puntuación = 8,5]
Se han consultado 7 películas del conjunto.
```

3) Consultar todas las películas del conjunto, en orden por año de estreno ascendente.

Si el conjunto no contiene ninguna película, visualizará en consola el mensaje:

```
El conjunto está vacío.
```

Si el conjunto contiene una o más películas:

- Visualizará en consola un listado de todas las películas contenidas en el conjunto, indicando para cada película, un resumen con los nombres y valores de todos los atributos de dicha película. Este listado deberá estar ordenado por el año de estreno de la película de forma ascendente.
- Visualizará en consola el número de películas consultadas del conjunto.

Por ejemplo:

```
Película [Código = primera, Título = Raiders of the Lost Ark,
          Director = Steven Spielberg, AñoEstreno = 1981,
          Duración = 115, Puntuación = 8,4]
Película [Código = quinta, Título = Goodfellas,
          Director = Martin Scorsese, AñoEstreno = 1990,
          Duración = 146, Puntuación = 8,7]
Película [Código = segunda, Título = Jurassic Park,
          Director = Steven Spielberg, AñoEstreno = 1993,
          Duración = 127, Puntuación = 8,1]
Película [Código = sexta, Título = Pulp Fiction,
          Director = Quentin Tarantino, AñoEstreno = 1994,
          Duración = 154, Puntuación = 8,9]
Película [Código = cuarta, Título = Titanic,
          Director = James Cameron, AñoEstreno = 1997,
          Duración = 195, Puntuación = 7,8]
Película [Código = séptima, Título = Gladiator,
          Director = Ridley Scott, AñoEstreno = 2000,
          Duración = 155, Puntuación = 8,5]
Película [Código = tercera, Título = Avatar,
          Director = James Cameron, AñoEstreno = 2009,
          Duración = 162, Puntuación = 7,8]
Se han consultado 7 películas del conjunto.
```

4) Consultar varias películas, por director, del conjunto, en orden por puntuación descendente.

Leerá por teclado el director de las películas a consultar.
Si el conjunto no contiene ninguna película con el director dado, visualizará en consola el mensaje:

```
No existe ninguna película con ese director en el conjunto.
```

Si el conjunto contiene una o más películas con el director dado:

- Visualizará en consola un listado de las películas del director contenidas en el conjunto, indicando para cada película, un resumen con los nombres y valores de todos los atributos de dicha película. Este listado deberá estar ordenado por la puntuación de la película de forma descendente.
- Visualizará en consola el número de películas consultadas del conjunto.

El siguiente ejemplo muestra las películas cuyo director es `Steven Spielberg`, ordenadas por puntuación descendente:

```
Película [Código = primera, Título = Raiders of the Lost Ark,
          Director = Steven Spielberg, AñoEstreno = 1981,
          Duración = 115, Puntuación = 8,4]
Película [Código = segunda, Título = Jurassic Park,
          Director = Steven Spielberg, AñoEstreno = 1993,
          Duración = 127, Puntuación = 8,1]
Se han consultado 2 películas del conjunto.
```

5) Eliminar varias películas, entre dos duraciones, del conjunto.

Leerá por teclado la duración mínima y la duración máxima de las películas a eliminar.
Si el conjunto no contiene ninguna película entre las duraciones mínima y máxima dadas, visualizará en consola el mensaje:

```
No existe ninguna película entre esas duraciones mínima y máxima
en el conjunto.
```

Si el conjunto contiene una o más películas entre las duraciones mínima y máxima dadas:

- Eliminará las películas cuya duración está comprendida entre dos duraciones mínima y máxima del conjunto.
- Visualizará en consola el número de películas eliminadas del conjunto.

El siguiente ejemplo muestra el mensaje tras eliminar las películas que tienen una duración entre `151` y `200` minutos:

```
Se han eliminado 4 películas del conjunto.
```

Este programa principal deberá validar que la opción de menú elegida sea válida (comprendida entre 0 y 5). Si no lo es, visualizará en consola el mensaje:

```
La opción de menú debe estar comprendida entre 0 y 5.
```

MAPAS DE OBJETOS

ACTIVIDAD 4x01

Codifica un programa principal **`main`** que:

- Lea por teclado de forma repetitiva nombres y notas de una serie de alumnos y los inserte como pares en una colección de la clase **`TreeMap`** (mapa árbol), definiendo el nombre como clave y la nota como valor asociado y guardando estos datos ordenados por nombre de forma ascendente.
  - Si el nombre leído es una clave del mapa árbol, visualizará en consola un mensaje indicando que ese nombre ya se encuentra en el mapa árbol.
  - En caso contrario, leerá una nota (que deberá estar comprendida entre 1 y 10) e insertará el par nombre/nota en el mapa árbol.

  Este proceso deberá repetirse hasta que se introduzca la cadena ***, la cual no deberá insertarse en el mapa árbol.
- Recorra los elementos del mapa árbol de alumnos con un iterador sobre el conjunto de claves y los visualice en consola.
- Recorra los elementos del mapa árbol de alumnos con un bucle *for-each* sobre el conjunto de claves y los visualice en consola.
- Recorra los elementos del mapa árbol de alumnos con un iterador sobre el conjunto de pares o entradas clave/valor y los visualice en consola.
- Recorra los elementos del mapa árbol de alumnos con un bucle *for-each* sobre el conjunto de pares o entradas clave/valor y los visualice en consola.

Un ejemplo de ejecución del programa podría ser:

```
¿Nombre (*** para terminar)? Juan
¿Nota? 5
¿Nombre (*** para terminar)? Pablo
¿Nota? 7
¿Nombre (*** para terminar)? Jesús
¿Nota? 4
¿Nombre (*** para terminar)? José
¿Nota? 2
¿Nombre (*** para terminar)? Francisco
¿Nota? -1
La nota debe estar comprendida entre 1 y 10.
¿Nota? 9
¿Nombre (*** para terminar)? Elena
¿Nota? 6
¿Nombre (*** para terminar)? María
¿Nota? 1
¿Nombre (*** para terminar)? Isabel
¿Nota? 8
¿Nombre (*** para terminar)? Marta
¿Nota? 3
```

```
¿Nombre (*** para terminar)? Lucía
¿Nota? 12
La nota debe estar comprendida entre 1 y 10.
¿Nota? 10
¿Nombre (*** para terminar)? Pablo
Ya existe un alumno con ese nombre en el mapa.
¿Nombre (*** para terminar)? Marta
Ya existe un alumno con ese nombre en el mapa.
¿Nombre (*** para terminar)? ***

Elementos del Mapa Árbol de Alumnos (Iterador de Claves):
(Nombre = Elena, Nota = 6) (Nombre = Francisco, Nota = 9)
(Nombre = Isabel, Nota = 8) (Nombre = Jesús, Nota = 4)
(Nombre = José, Nota = 2) (Nombre = Juan, Nota = 5)
(Nombre = Lucía, Nota = 10) (Nombre = Marta, Nota = 3)
(Nombre = María, Nota = 1) (Nombre = Pablo, Nota = 7)

Elementos del Mapa Árbol de Alumnos (Bucle for-each de Claves):
(Nombre = Elena, Nota = 6) (Nombre = Francisco, Nota = 9)
(Nombre = Isabel, Nota = 8) (Nombre = Jesús, Nota = 4)
(Nombre = José, Nota = 2) (Nombre = Juan, Nota = 5)
(Nombre = Lucía, Nota = 10) (Nombre = Marta, Nota = 3)
(Nombre = María, Nota = 1) (Nombre = Pablo, Nota = 7)

Elementos del Mapa Árbol de Alumnos (Iterador de Entradas):
(Nombre = Elena, Nota = 6) (Nombre = Francisco, Nota = 9)
(Nombre = Isabel, Nota = 8) (Nombre = Jesús, Nota = 4)
(Nombre = José, Nota = 2) (Nombre = Juan, Nota = 5)
(Nombre = Lucía, Nota = 10) (Nombre = Marta, Nota = 3)
(Nombre = María, Nota = 1) (Nombre = Pablo, Nota = 7)

Elementos del Mapa Árbol de Alumnos (Bucle for-each de Entradas):
(Nombre = Elena, Nota = 6) (Nombre = Francisco, Nota = 9)
(Nombre = Isabel, Nota = 8) (Nombre = Jesús, Nota = 4)
(Nombre = José, Nota = 2) (Nombre = Juan, Nota = 5)
(Nombre = Lucía, Nota = 10) (Nombre = Marta, Nota = 3)
(Nombre = María, Nota = 1) (Nombre = Pablo, Nota = 7)
```

ACTIVIDAD 4x02

Codifica un programa principal **`main`** que:

- Lea por teclado de forma repetitiva notas y nombres de una serie de alumnos y los inserte como pares en una colección de la clase **`TreeMap`** (mapa árbol), definiendo la nota como clave y un conjunto de nombres (una colección de la clase **`TreeSet`**) como valor asociado y guardando estos datos ordenados por nota de forma ascendente y nombre de forma ascendente.
  - Si el nombre leído está incluido en algún valor asociado del mapa árbol, visualizará en consola un mensaje indicando que ese nombre ya se encuentra en el mapa árbol.
  - Si el nombre leído no está incluido en ningún valor asociado del mapa y la nota leída es una clave del mapa árbol, obtendrá el conjunto de nombres asociado a la nota del mapa árbol y añadirá el nombre a este conjunto de nombres.
  - Si el nombre leído no está incluido en ningún valor asociado del mapa y la nota leída no es una clave del mapa árbol, creará un conjunto de nombres que contenga el nombre e insertará el par formado por la nota y este conjunto de nombres en el mapa árbol.

  Este proceso deberá repetirse hasta que se introduzca el número `0`, el cual no deberá insertarse en el mapa árbol.
- Recorra los elementos del mapa árbol de notas a través de su conjunto de claves y los visualice en consola.
- Recorra los elementos del mapa árbol de notas a través de su conjunto de pares o entradas clave/valor y los visualice en consola.

Un ejemplo de ejecución del programa podría ser:

```
¿Nota (0 para terminar)? 1
¿Nombre? Juan
¿Nota (0 para terminar)? 2
¿Nombre? Marta
¿Nota (0 para terminar)? 3
¿Nombre? Pablo
¿Nota (0 para terminar)? 3
¿Nombre? Valeria
¿Nota (0 para terminar)? 4
¿Nombre? Isabel
¿Nota (0 para terminar)? 4
¿Nombre? Diego
¿Nota (0 para terminar)? 4
¿Nombre? Ricardo
¿Nota (0 para terminar)? 5
¿Nombre? Pedro
¿Nota (0 para terminar)? 5
¿Nombre? Sofía
¿Nota (0 para terminar)? 5
¿Nombre? Paula
¿Nota (0 para terminar)? 5
¿Nombre? Carlos
¿Nota (0 para terminar)? 6
¿Nombre? María
¿Nota (0 para terminar)? 6
¿Nombre? Alberto
```

```
¿Nota (0 para terminar)? 6
¿Nombre? José
¿Nota (0 para terminar)? 6
¿Nombre? Pilar
¿Nota (0 para terminar)? 7
¿Nombre? Elena
¿Nota (0 para terminar)? 7
¿Nombre? Teresa
¿Nota (0 para terminar)? 7
¿Nombre? Carmen
¿Nota (0 para terminar)? 8
¿Nombre? Jorge
¿Nota (0 para terminar)? 8
¿Nombre? Luis
¿Nota (0 para terminar)? 9
¿Nombre? Lucía
¿Nota (0 para terminar)? 10
¿Nombre? Francisco
¿Nota (0 para terminar)? -2
La nota debe estar comprendida entre 0 y 10.
¿Nota (0 para terminar)? 3
¿Nombre? Valeria
Ya existe un alumno con ese nombre en el mapa.
¿Nota (0 para terminar)? 0

Elementos del Mapa Árbol de Notas (Conjunto de Claves):
Nota = 1 --> Nombres = [Juan]
Nota = 2 --> Nombres = [Marta]
Nota = 3 --> Nombres = [Pablo, Valeria]
Nota = 4 --> Nombres = [Diego, Isabel, Ricardo]
Nota = 5 --> Nombres = [Carlos, Paula, Pedro, Sofía]
Nota = 6 --> Nombres = [Alberto, José, María, Pilar]
Nota = 7 --> Nombres = [Carmen, Elena, Teresa]
Nota = 8 --> Nombres = [Jorge, Luis]
Nota = 9 --> Nombres = [Lucía]
Nota = 10 --> Nombres = [Francisco]

Elementos del Mapa Árbol de Notas (Conjunto de Entradas):
Nota = 1 --> Nombres = [Juan]
Nota = 2 --> Nombres = [Marta]
Nota = 3 --> Nombres = [Pablo, Valeria]
Nota = 4 --> Nombres = [Diego, Isabel, Ricardo]
Nota = 5 --> Nombres = [Carlos, Paula, Pedro, Sofía]
Nota = 6 --> Nombres = [Alberto, José, María, Pilar]
Nota = 7 --> Nombres = [Carmen, Elena, Teresa]
Nota = 8 --> Nombres = [Jorge, Luis]
Nota = 9 --> Nombres = [Lucía]
Nota = 10 --> Nombres = [Francisco]
```