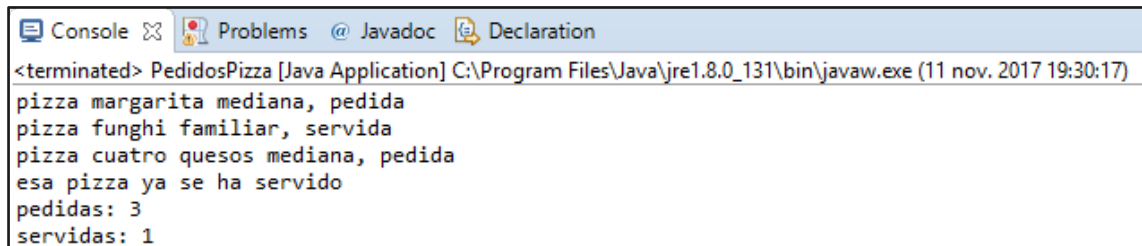


A realizar en paquete objetos

1. Implementa la clase Caballo (piensa antes los posibles atributos y métodos). Pruébala creando instancias y aplicándole algunos métodos.
2. Crea la clase Pizza con los atributos y métodos necesarios. Sobre cada pizza se necesita saber el tamaño - mediana o familiar - el tipo - margarita, cuatro quesos o funghi - y su estado - pedida o servida. La clase debe almacenar información sobre el número total de pizzas que se han pedido y que se han servido. Siempre que se crea una pizza nueva, su estado es "pedida". El siguiente código del programa principal, que prueba la clase anterior, debe dar la salida que se muestra:

```
public class PedidosPizza {  
    public static void main(String[] args) {  
        Pizza p1 = new Pizza("margarita", "mediana");  
        Pizza p2 = new Pizza("funghi", "familiar");  
        p2.sirve();  
        Pizza p3 = new Pizza("cuatro quesos", "mediana");  
        System.out.println(p1);  
        System.out.println(p2);  
        System.out.println(p3);  
        p2.sirve();  
        System.out.println("pedidas: " + Pizza.getTotalPedidas());  
        System.out.println("servidas: " + Pizza.getTotalServidas());  
    }  
}
```



The screenshot shows a Java IDE window with tabs for Console, Problems, Javadoc, and Declaration. The Console tab is active, displaying the output of the application. The output shows three pizzas being created and their states, followed by the total number of orders and servings.

```
<terminated> PedidosPizza [Java Application] C:\Program Files\Java\jre1.8.0_131\bin\javaw.exe (11 nov. 2017 19:30:17)  
pizza margarita mediana, pedida  
pizza funghi familiar, servida  
pizza cuatro quesos mediana, pedida  
esa pizza ya se ha servido  
pedidas: 3  
servidas: 1
```

3. Queremos gestionar la venta de entradas (no numeradas) de Expocoches DAM que tiene 3 zonas, la sala principal con 1000 entradas disponibles, la zona de compra-venta con 200 entradas disponibles y la zona vip con 25 entradas disponibles. Hay que controlar que existen entradas antes de venderlas. La clase Zona con sus atributos y métodos se muestra a continuación:

```
public class Zona {
    private int entradasPorVender;
    public Zona(int n){
        entradasPorVender = n;
    }

    public int getEntradasPorVender() {
        return entradasPorVender;
    }

    //Vende un número de entradas. Comprueba si quedan entradas libres antes de realizar la venta.
    public void vender(int n) {
        if (this.entradasPorVender == 0) {
            System.out.println("Lo siento, las entradas para esa zona están agotadas.");
        } else if (this.entradasPorVender < n) {
            System.out.println("Sólo quedan " + this.entradasPorVender + " entradas para esa zona.");
        }

        if (this.entradasPorVender >= n) {
            entradasPorVender -= n;
            System.out.println("Aquí- tiene sus " + n + " entradas, gracias.");
        }
    }
}
```

El menú del programa debe ser el que se muestra a continuación. Cuando elegimos la opción 2, se nos debe preguntar para qué zona queremos las entradas y cuántas queremos. Lógicamente, el programa debe controlar que no se puedan vender más entradas de la cuenta.

- | |
|--|
| <ol style="list-style-type: none">1. Mostrar número de entradas libres2. Vender entradas3. Salir |
|--|

A realizar en paquete colecciones_I

1. Desarrolla una clase Cancion con los siguientes atributos:

- ✓ título: una variable String que guarda el título de la canción.
- ✓ autor: una variable String que guarda el autor de la canción.

y los siguientes métodos:

- ✓ Cancion(String, String): constructor que recibe como parámetros el título y el autor de la canción (por este orden).
- ✓ Cancion(): constructor predeterminado que inicializa el título y el autor a cadenas vacías.
- ✓ dameTitulo(): devuelve el título de la canción.
- ✓ dameAutor(): devuelve el autor de la canción.
- ✓ ponTitulo(String): establece el título de la canción.
- ✓ ponAutor(String): establece el autor de la canción.

2. Desarrolla una clase CD con los siguientes atributos:

- ✓ canciones: un array de objetos de la clase Cancion.
- ✓ contador: la siguiente posición libre del array canciones.

y los siguientes métodos:

- ✓ CD(): constructor predeterminado (creará el array canciones).
- ✓ numeroCanciones(): devuelve el valor del contador de canciones.
- ✓ dameCancion(int): devuelve la Cancion que se encuentra en la posición indicada.
- ✓ grabaCancion(int, Cancion): cambia la Cancion de la posición indicada por la nueva Cancion proporcionada. Devuelve cierto/falso si se ha podido realizar la operación o no.
- ✓ agrega(Cancion): agrega al final del array la Cancion proporcionada. Devuelve cierto/falso si se ha podido realizar la operación o no.
- ✓ elimina(int): elimina la Cancion que se encuentra en la posición indicada. Devuelve cierto/falso si se ha podido realizar la operación o no.

3. Crea una clase Libro que modele la información que se mantiene en una biblioteca sobre cada libro: título, autor (usa la clase Persona), ISBN, páginas, edición, editorial, lugar (ciudad y país) y fecha de edición (usa la clase Fecha). La clase debe proporcionar los siguientes servicios: getters y setters, método para leer la información y método para mostrar la información. Este último método mostrará la información del libro con este formato:

Título: Introduction to Java Programming
3a. edición
Autor: Liang, Y. Daniel
ISBN: 0-13-031997-X
Prentice-Hall, New Jersey (USA), viernes 16 de noviembre de 2001
784 páginas

4. Desarrollar una lista de Libros **ordenada por título**. La funcionalidad de la lista será la habitual: conocer el número de libros que hay en la lista, insertar un nuevo libro (en la posición que le corresponda), eliminar el libro de una determinada posición y obtener el libro de una determinada posición. También incluirá un método para buscar un libro a partir de una parte de su título (sin distinguir entre mayúsculas y minúsculas); el método devolverá la posición en la que se encuentra el libro (–1 si no se encuentra).

A realizar en paquete colecciones_II

1. Escribe 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 apilarPalabra para poner una palabra en la cima de la pila, el método desapilarPalabra para quitar el elemento de la cima de la pila devolviéndolo y el método obtenerPalabraCima para obtener la palabra situada en la cima de la pila sin quitarla de ella. También se implementará el método pilaPalabrasVacía para determinar si la pila está o no vacía. Los métodos deben implementarse utilizando la clase LinkedList.

Escribe un programa que utilizando la clase PilaPalabras, introduzca varias cadenas de caracteres en la pila y las desapile mostrándolas por pantalla.

2. Realiza la clase del ejercicio 1 anterior pero utilizando la clase ArrayList.
3. Escribe una clase Producto con atributos código (int), nombre (String), tipo (String), precio (double), stock (int). Implementa la clase ArrayProducto, con un atributo de nombre lista y tipo ArrayList<Producto> y con los métodos tamaño de la lista, imprimirLista, agregar un producto, buscar y recuperar el producto por la posición, buscar y recuperar el producto por su código, eliminar un producto por su código, cambiarPrecio (del producto mediante su código y el nuevo precio). Escribe después un programa para probar la clase anterior.
4. Realiza el ejercicio 3 anterior pero utilizando la clase LinkedList.

A realizar en paquete colecciones_III

1. Realiza una copia del ejemplo ArrayListLibros (paquete _03colecciones2). De la clase ArrayListLibros.java, borra el método public boolean insertarOrden(Libro p), pues utilizaremos el otro método public boolean insertar(Libro p) para añadir libros que se encuentra también dentro de la clase anterior. Añade dos clases más al proyecto, que implementen la interfaz Comparator: TituloComparator y PaginasComparator, que permitan ordenar la colección por título y número de páginas respectivamente. Prueba con datos el funcionamiento del proyecto (puedes modificar una copia de la clase PruebaArrayListLibros).
2. Implementa una nueva solución del ejercicio anterior pero utilizando Streams (voluntario).
3. Implementa una nueva solución del ejemplo ArrayListLibros (paquete _03colecciones2) pero utilizando la colección HashSet en lugar del ArrayList. La funcionalidad de la colección será la habitual: conocer el número de libros que hay en la colección, insertar un nuevo libro, eliminar el libro que se le pase como argumento, averiguar si la colección contiene un libro. También incluirá un método para buscar un libro a partir de una parte de su título (sin distinguir entre mayúsculas y minúsculas); el método imprimirá todos los datos de los libros encontrados. Prueba con datos el funcionamiento del proyecto.
4. Ídem del ejercicio anterior pero utilizando la colección TreeSet. Prueba con datos el funcionamiento del proyecto.
5. Crea un mini-diccionario español-inglés que contenga, al menos, 20 palabras (con su correspondiente traducción). Utiliza un objeto de la clase HashMap para almacenar las parejas de palabras. El programa pedirá una palabra en español y dará la correspondiente traducción en inglés.
Realiza otro programa que escoja al azar 5 palabras en español del mini-diccionario del ejercicio anterior. El programa irá pidiendo que el usuario teclee la traducción al inglés de cada una de las palabras y comprobará si son correctas. Al final, el programa deberá mostrar cuántas respuestas son válidas y cuántas erróneas.

A realizar en paquete excepciones

- a) Realiza una clase que pida al usuario un entero utilizando para la entrada la clase Scanner. Averigua qué excepción concreta se genera cuando lo que se recibe no es un entero y atrápala.
- b) Escribe una clase con un método de nombre enviarMensaje, que reciba por parámetro una cadena de caracteres correspondiente a una dirección de correo electrónico. El método comprobará que la dirección recibida es correcta elevando la excepción DirCorreoIncorrectaExcepcion en caso contrario. La comprobación consistirá en verificar que la dirección contiene el carácter “@”, algún carácter después de él antes del carácter “. ” y algún carácter después de éste.