Persistencia de Datos

Para que los datos de la aplicación se mantengan entre diferentes ejecuciones, se planteó un sistema de serialización de objetos:

La lógica se puede ver en nuestro proyecto en la clase *PersistenceManager,* encontrada en (es.upm.etsisi.poo.persistence)

A)Lógica:

➔ Guardado

Se usa el objeto *ObjectOutputStream*, para serializar un *StoreData*, y con este escribir la información en disco en forma de fichero.

➔ Carga

Se usa el *ObjectInputStream* para realizar el proceso inverso a la serialización

B)Almacenamiento:

Se guardan los datos en disco en un archivo de formato binario, llamando *store_data.dat*, este nombre se le define con la constante que da el camino al archivo.
Para esto se hace uso de un *StoreData* que guarda en conjunto el estado de la tienda tras una ejecución.

```java
package es.upm.etsisi.poo.persistence;
import es.upm.etsisi.poo.utils.StaticMessages;
import java.io.*;

public class PersistenceManager {  Jorge
    private static final String FILE_PATH = "store_data.dat"; // El archivo local

    public static void save(StoreData data) {  Jorge
        try (ObjectOutputStream oos = new ObjectOutputStream(new FileOutputStream(FILE_PATH))) {
            oos.writeObject(data);
        } catch (IOException e) {
            System.out.println(String.format(StaticMessages.PERSISTENCE_SAVE_ERROR, e.getMessage()));
        }
    }

    public static StoreData load() {  Jorge
        File file = new File(FILE_PATH);
        if (!file.exists()) {
            return null; // Primera ejecución, no hay datos
        }
        try (ObjectInputStream ois = new ObjectInputStream(new FileInputStream(file))) {
            return (StoreData) ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            return null;
        }
    }
}
```

C)Manejo de errores

Antes de intentar leer con un condicional se ve si el archivo existe(si hay algo guardado), si no existe simplemente no se lee nada, y si existe se usa un *try-catch* que captura errores de lectura o de clase de datos inicializados incorrectamente.
En el caso de guardar se capturan IOExceptions, y si falla se le informa al usuario.
Debido a la estructura usada no hace falta un bloque *Finally* porque ya se cierra de manera automática el fichero.
A lo largo del proyecto hemos usado una clase llamada *StaticMessages*, la cual se encarga de almacenar distintos tipos de mensajes usados a lo largo de la implementación, entre estos mensajes existe un apartado de Errores el cual contiene todos lo mensajes dados por el programa en caso de errores, esto se puede ver claramente en la clase Main (*StoreApp*) donde cada vez que se encuentre un error con un throws lo manejamos haciendo uso de los mensajes hablados.

D)Librerías Usadas

Para el desarrollo, no se han utilizado librerías externas o programas de terceros, todo se ha hecho de manera nativa, es decir, con librerías estándar;
se ha utilizado java.io, específicamente la ya mencionadas *ObjectOutputStream, ObjectInputStream*
Y *java.util* para la gestión de datos, de aquí se usaron *arrayList y HashMaps*
Para evitar problemas de que se intente acceder a alguna estructura desde dos lados se usó java Serializable.
esto se puede ver en clases como *AbstractProduct, Ticket y User*

E)Ejemplos

```
public class UserDatabase {  22 usages   Jorge +1

    private static UserDatabase instance; // Singleton  3 usages
    private static ArrayList<User> users = new ArrayList<>();; // Unica Database  9 usages

    private UserDatabase() {}  1 usage   Jorge
```

```
public class ProductCatalog {  18 usages   Jorge +3
    private static final int MAX_PRODUCTS = 200;  1 usage
    private static HashMap<String, AbstractProduct> products = new HashMap<>();  15 usages

    public static AbstractProduct getProduct(String id) { return products.get(id); }

    public static Map<String, AbstractProduct> getList() { return products; }


    public static ArrayList<AbstractProduct> getProducts() { return new ArrayList<>(products.values()); }
```

```java
import es.upm.etsisi.poo.model.products.AbstractProduct;
import es.upm.etsisi.poo.model.sales.Ticket;
import es.upm.etsisi.poo.model.users.User;
import java.io.Serializable;
import java.util.List;

public class StoreData implements Serializable {  6 usages  Jorge
    private static final long serialVersionUID = 1L;  no usages

    private List<User> users;  2 usages
    private List<AbstractProduct> products;  2 usages
    private List<Ticket> tickets;  1 usage

    public StoreData(List<User> users, List<AbstractProduct> products, List<Ticket> tickets) {  16 usages  Jorge
        this.users = users;
        this.products = products;
        this.tickets = tickets;
    }

    public List<User> getUsers() { return users; }  Jorge
    public List<AbstractProduct> getProducts() { return products; }  Jorge
}
```

*Aquí se puede ver lo del uso de serializable, en este caso es para guardar usuarios productos y tickets de una, envés de manejar distintos archivos, de esta manera tenemos toda la persistencia centralizada en un solo punto, simplificando el código, y garantizando que todo se mantenga unido, en vez de tener tickets guardados por un lado y los usuarios en otro, esto a su vez facilita un posible mantenimiento del código.

F) Algunos ejemplos del funcionamiento:

```
Welcome to the ticket module App.
Ticket module. Type 'help' to see commands.
tUPM> help
Commands:
  client add "<nombre>" (<DNI>|<NIF>) <email> <cashId>
  client remove <DNI>
  client list
  cash add [<id>] "<nombre>"<email>
  cash remove <id>
  cash list
  cash tickets <id>
  ticket new [<id>] <cashId> <userId> -[c/p/s] (default -p option)
  ticket add <ticketId><cashId> <prodId> <amount> [--p<txt> --p<txt>]
  ticket remove <ticketId><cashId> <prodId>
  ticket print <ticketId> <cashId>
  ticket list
  prod add ([<id>] "<name>" <category> <price> [<maxPers>] || ("<name>" <category> )
  prod update <id> NAME|CATEGORY|PRICE <value>
  prod addFood [<id>] "<name>" <price> <expiration:yyyy-MM-dd> <max_people>
  prod addMeeting [<id>] "<name>" <price> <expiration:yyyy-MM-dd> <max_people>
  prod list
  prod remove <id>
  help
  echo "<text>"
```

```
Categories: MERCH, STATIONERY, CLOTHES, BOOK, ELECTRONICS
Discounts if there are ≥2 units in the category: MERCH 0%, STATIONERY 5%, CLOTHES 7%, BOOK 10%, ELECTRONICS 3%.

tUPM> echo "Agrego Libro"
"Agrego Libro"

tUPM> prod add 1 "Libro POO" BOOK 25
{class:Product, id:1, name:'Libro POO', category:BOOK, price:25.0}
prod add: ok

tUPM> echo "Agrego Camiseta"
"Agrego Camiseta"

tUPM> prod add 2 "Camiseta talla:M UPM" CLOTHES 15
{class:Product, id:2, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
prod add: ok

tUPM> prod add "Camiseta talla:M UPM" CLOTHES 15
{class:Product, id:PR6413594, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
tUPM> echo "Listo Productos"
"Listo Productos"

tUPM> prod list
Catalog:
  {class:Product, id:1, name:'Libro POO', category:BOOK, price:25.0}
  {class:Product, id:2, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
  {class:Product, id:PR6413594, name:'Camiseta talla:M UPM', category:CLOTHES, price:15.0}
prod list: ok

tUPM> echo "Actualizo Nombre y Precio del Libro"
"Actualizo Nombre y Precio del Libro"

tUPM> prod update 1 NAME "Libro POO V2"
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:25.0}
prod update: ok

tUPM> prod update 1 PRICE 30
{class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0}
```

```
tUPM> echo "Creo un Trabajador"
"Creo un Trabajador"

tUPM> cash add UW1234567 "pepecurro3" pepe0@upm.es
Cash{identifier='UW1234567', name='pepecurro3', email='pepe0@upm.es'}
cash add: ok

tUPM> cash add "pepecurro2" pepe0@upm.es
Cash{identifier='UW5235834', name='pepecurro2', email='pepe0@upm.es'}
cash add: ok

tUPM> echo "Creo tres cliente listo borro uno y listo"
"Creo tres cliente listo borro uno y listo"

tUPM> client add "Pepe3" 55630667S pepe1@upm.es UW1234567
USER{identifier='55630667S', name='Pepe3', email='pepe1@upm.es', cash=UW1234567}
client add: ok

tUPM> client add "Pepe2" 98948334B pepe2@upm.es UW1234567
USER{identifier='98948334B', name='Pepe2', email='pepe2@upm.es', cash=UW1234567}
client add: ok

tUPM> client add "Pepe1" Y8682724P pepe3@upm.es UW1234567
USER{identifier='Y8682724P', name='Pepe1', email='pepe3@upm.es', cash=UW1234567}
client add: ok
```

```
tUPM> client list
Client:
  USER{identifier='Y8682724P', name='Pepe1', email='pepe3@upm.es', cash=UW1234567}
  USER{identifier='98948334B', name='Pepe2', email='pepe2@upm.es', cash=UW1234567}
  USER{identifier='55630667S', name='Pepe3', email='pepe1@upm.es', cash=UW1234567}
client list: ok

tUPM> client remove Y8682724P
client remove: ok

tUPM> client list
Client:
  USER{identifier='98948334B', name='Pepe2', email='pepe2@upm.es', cash=UW1234567}
  USER{identifier='55630667S', name='Pepe3', email='pepe1@upm.es', cash=UW1234567}
client list: ok

tUPM> echo "creo un cajero nuevo listo, lo booro y listo de nuevo"
"creo un cajero nuevo listo, lo booro y listo de nuevo"

tUPM> cash add UW1234569 "pepecurro1" pepe0@upm.es
Cash{identifier='UW1234569', name='pepecurro1', email='pepe0@upm.es'}
cash add: ok

tUPM> cash list
Cash:
  Cash{identifier='UW1234569', name='pepecurro1', email='pepe0@upm.es'}
  Cash{identifier='UW5235834', name='pepecurro2', email='pepe0@upm.es'}
  Cash{identifier='UW1234567', name='pepecurro3', email='pepe0@upm.es'}
cash list: ok
```

```
tUPM> cash remove UW1234569
cash remove: ok

tUPM> cash list
Cash:
  Cash{identifier='UW5235834', name='pepecurro2', email='pepe0@upm.es'}
  Cash{identifier='UW1234567', name='pepecurro3', email='pepe0@upm.es'}
cash list: ok

tUPM> echo "miro los tickets de UW1234567 creo un ticket y vuelvo a printar"
"miro los tickets de UW1234567 creo un ticket y vuelvo a printar"

tUPM> cash tickets UW1234567
Tickets:
cash tickets: ok

tUPM> ticket new UW1234567 55630667S
Ticket : 26-01-18-15:10-44883
  Total price: 0.0
  Total discount: 0.0
  Final Price: 0.0
ticket new: ok

tUPM> cash tickets UW1234567
Tickets:
  26-01-18-15:10-44883->EMPTY
cash tickets: ok
```

```
tUPM> echo "creo un ticket con ID"
"creo un ticket con ID"

tUPM> ticket new 212121 UW1234567 55630667S
Ticket : 212121
  Total price: 0.0
  Total discount: 0.0
  Final Price: 0.0
ticket new: ok

tUPM> echo "Agrego un producto al ticket e imprimo el ticket"
"Agrego un producto al ticket e imprimo el ticket"
```

```
tUPM> ticket add 212121 UW1234567 1 20
Ticket : 212121
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
```

```
tUPM> ticket list
Ticket List:
  212121 - OPEN [55630667S]
  26-01-18-15:10-44883 - EMPTY [55630667S]
ticket list: ok

tUPM> ticket print 212121 UW1234567
Ticket : 212121-26-01-18-15: 10
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 Total price: 600.0
 Total discount: 60.0
 Final Price: 540.0
ticket print: ok
```

```
tUPM> ticket list
Ticket List:
  212121-26-01-18-15: 10 - CLOSE [55630667S]
  26-01-18-15:10-44883 - EMPTY [55630667S]
ticket list: ok

tUPM> echo "Agrego dos productos al ticket e imprimo el ticket"
"Agrego dos productos al ticket e imprimo el ticket"

tUPM> ticket new 212123 UW1234567 98948334B
Ticket : 212123
  Total price: 0.0
  Total discount: 0.0
  Final Price: 0.0
ticket new: ok
```

```
tUPM> ticket add 212123 UW1234567 1 20
Ticket : 212123
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
 {class:Product, id:1, name:'Libro POO V2', category:BOOK, price:30.0} **discount -3.0
```