PRACTICA NO.1 CARRITO DE COMPRA

Ares Ulises Juárez Martínez
3CV17

15 DE MAYO DE 2022

SANDRA IVETTE BAUTISTA ROSALES
Escuela Superior de Computo

Contenido

Objetivo 2

Introducción 2

Cuando un objeto Serializable se escribe con writeObject, luego se modifica y se escribe por segunda vez, ¿por qué falta la modificación cuando se deserializa la transmisión? 2

Apache PDFBox® - A Java PDF Library 2

Class Desktop 2

Building Java Projects with Maven 2

Desarrollo 3

Cuestionario 17

Conclusiones 18

Referencias 18

Objetivo

El estudiante implementará una aplicación de carrito de compra para la selección y adquisición de artículos, generación de recibo de compra y el envío de múltiples objetos serializados a través de la red haciendo uso de sockets de flujo bloqueantes.

Introducción

Cuando un objeto Serializable se escribe con writeObject, luego se modifica y se escribe por segunda vez, ¿por qué falta la modificación cuando se deserializa la transmisión?

La clase realiza un seguimiento de cada objeto que serializa y envía solo el identificador si el objeto se escribe en la secuencia una vez posterior. Esta es la forma en que trata con gráficos de objetos. El correspondiente realiza un seguimiento de todos los objetos que ha creado y sus identificadores, de modo que cuando el identificador se vuelve a ver, puede devolver el mismo objeto. Tanto los flujos de salida como los de entrada mantienen este estado hasta que se liberan.

Alternativamente, la clase implementa un método de reinicio que descarta la memoria de haber enviado un objeto, por lo que enviar un objeto nuevamente hará una copia. [1]

Apache PDFBox® - A Java PDF Library

La biblioteca Apache PDFBox® es una herramienta Java de código abierto para trabajar con documentos PDF. Este proyecto permite la creación de nuevos documentos PDF, la manipulación de documentos existentes y la capacidad de extraer contenido de los documentos. Apache PDFBox también incluye varias utilidades de línea de comandos. Apache PDFBox se publica bajo la Licencia Apache v2.0. [2]

Class Desktop

La clase Desktop permite que una aplicación Java inicie aplicaciones asociadas registradas en el escritorio nativo para manejar un URI o un archivo.

Las operaciones admitidas incluyen:

- iniciar el navegador predeterminado del usuario para mostrar un URI específico;
- iniciar el cliente de correo predeterminado del usuario con un mailto URI opcional;
- iniciar una aplicación registrada para abrir, editar o imprimir un archivo específico.

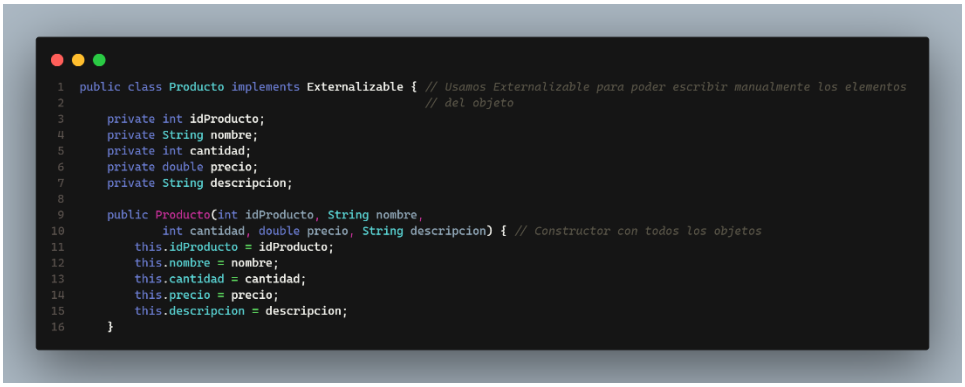
Esta clase proporciona métodos correspondientes a estas operaciones. Los métodos buscan la aplicación asociada registrada en la plataforma actual y la inician para manejar un URI o archivo. Si no hay una aplicación asociada o la aplicación asociada no se inicia, se genera una excepción. [3]

Building Java Projects with Maven

Apache Maven es una herramienta de comprensión y gestión de proyectos de software. Basado en el concepto de un modelo de objetos de proyecto (POM), Maven puede administrar la construcción, los informes y la documentación de un proyecto desde una pieza central de información. [4]

Desarrollo

Se comenzó programando la clase de productos, definiendo sus atributos e implementando la interfaz de Externizable



```java
public class Producto implements Externalizable { // Usamos Externalizable para poder escribir manualmente los elementos
                                                   // del objeto
    private int idProducto;
    private String nombre;
    private int cantidad;
    private double precio;
    private String descripcion;

    public Producto(int idProducto, String nombre,
            int cantidad, double precio, String descripcion) { // Constructor con todos los objetos
        this.idProducto = idProducto;
        this.nombre = nombre;
        this.cantidad = cantidad;
        this.precio = precio;
        this.descripcion = descripcion;
    }
```

Código 1. Segmento de la definición de Producto

Ahora se procede a definir un método para obtener las imágenes de su producto

```java
public ArrayList<File> obtenerImagenes() { // Metodo para obtener las imagenes de cada producto, esta fijo a la
                                           // cantidad de productos
        ArrayList<File> imagenes = new ArrayList<File>();
        Path path = Paths.get("./Imagenes/");
        File a1, a2, a3, a4;
        if (Files.exists(path)) {
            a1 = new File("./Imagenes/" + idProducto + "_1.png");
            a2 = new File("./Imagenes/" + idProducto + "_2.png");
            a3 = new File("./Imagenes/" + idProducto + "_3.png");
            a4 = new File("./Imagenes/" + idProducto + "_4.png");
        } else {
            a1 = new File("./" + idProducto + "_1.png");
            a2 = new File("./" + idProducto + "_2.png");
            a3 = new File("./" + idProducto + "_3.png");
            a4 = new File("./" + idProducto + "_4.png");
        }
        imagenes.add(a1);
        imagenes.add(a2);
        imagenes.add(a3);
        imagenes.add(a4);
        return imagenes;
    }
```

Código 2 Segmento del producto para obtener las imágenes

Después se implementaron los métodos para poder realizar la serialización del producto.

```java
@Override
    public void writeExternal(ObjectOutput out) throws IOException { // Metodo para escribir el objeto
        out.writeInt(idProducto);
        out.writeUTF(nombre);
        out.writeUTF(descripcion);
        out.writeInt(cantidad);
        out.writeDouble(precio);

    }

    @Override
    public void readExternal(ObjectInput in)
            throws IOException, ClassNotFoundException { // Metodo para leer el objeto
        this.idProducto = in.readInt();
        this.nombre = in.readUTF();
        this.descripcion = in.readUTF();
        this.cantidad = in.readInt();
        this.precio = in.readDouble();

    }
```

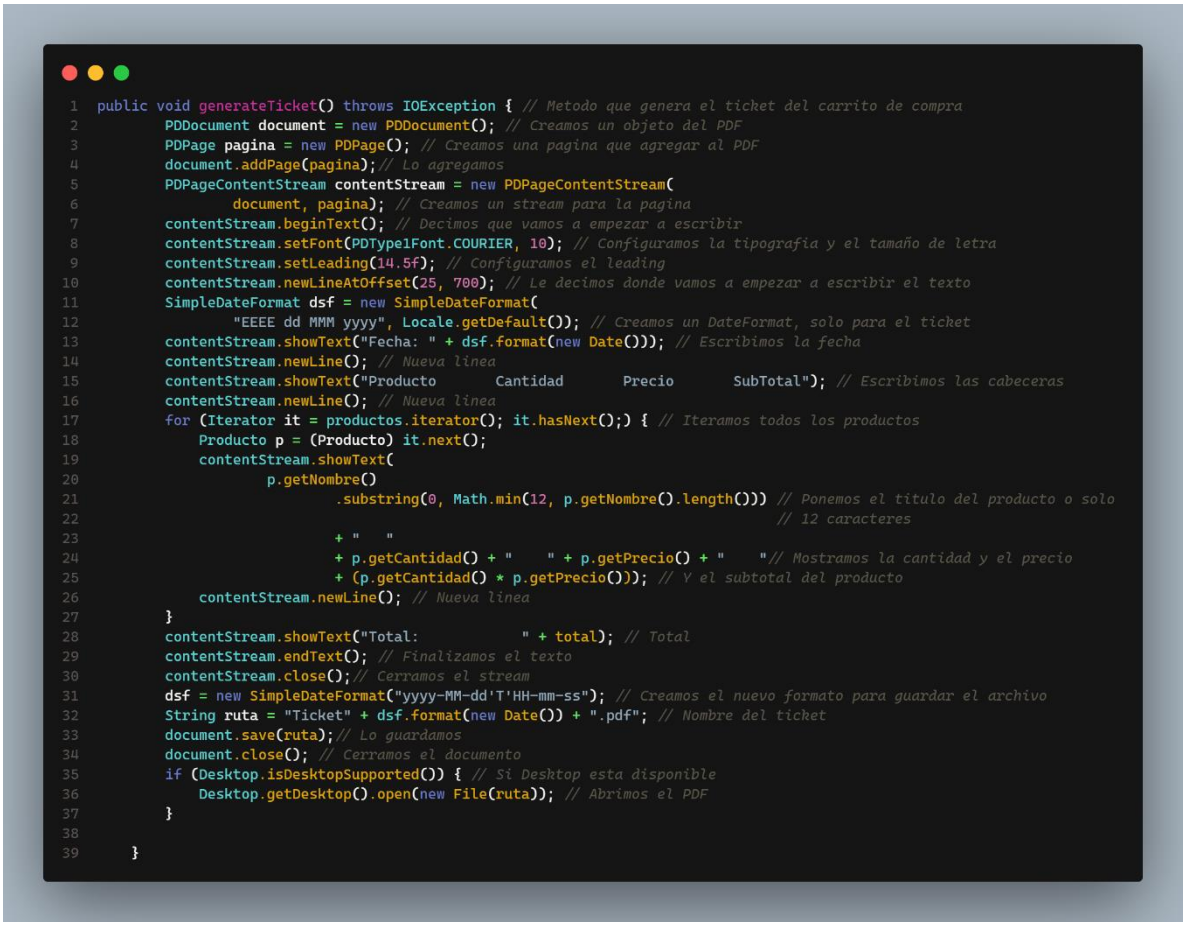
Código 3 Segmento de código de serialización de Productos

A continuación, se realizó la clase de CarritoDeCompra, con sus atributos y de igual manera implementando la interfaz de Externizable, el cual va a tener un ArrayList con los productos

```java
public class CarritoDeCompra implements Externalizable { // Usamos Externalizable para poder escribir el ArrayList

    private ArrayList<Producto> productos;
    private double total;

    public CarritoDeCompra() { // Constructor default
        productos = new ArrayList<Producto>();
        total = 0.0;
    }
```

Código 4 Segmento de código del CarritoDeCompra

Y después se tiene el código de generación de ticket, utilizando la biblioteca de PDFBox, el cual solo genera una página simple de ticket.



```java
public void generateTicket() throws IOException { // Metodo que genera el ticket del carrito de compra
        PDDocument document = new PDDocument(); // Creamos un objeto del PDF
        PDPage pagina = new PDPage(); // Creamos una pagina que agregar al PDF
        document.addPage(pagina);// Lo agregamos
        PDPageContentStream contentStream = new PDPageContentStream(
                document, pagina); // Creamos un stream para la pagina
        contentStream.beginText(); // Decimos que vamos a empezar a escribir
        contentStream.setFont(PDType1Font.COURIER, 10); // Configuramos la tipografia y el tamaño de letra
        contentStream.setLeading(14.5f); // Configuramos el leading
        contentStream.newLineAtOffset(25, 700); // Le decimos donde vamos a empezar a escribir el texto
        SimpleDateFormat dsf = new SimpleDateFormat(
                "EEEE dd MMM yyyy", Locale.getDefault()); // Creamos un DateFormat, solo para el ticket
        contentStream.showText("Fecha: " + dsf.format(new Date())); // Escribimos la fecha
        contentStream.newLine(); // Nueva linea
        contentStream.showText("Producto       Cantidad       Precio       SubTotal"); // Escribimos las cabeceras
        contentStream.newLine(); // Nueva linea
        for (Iterator it = productos.iterator(); it.hasNext();) { // Iteramos todos los productos
            Producto p = (Producto) it.next();
            contentStream.showText(
                    p.getNombre()
                            .substring(0, Math.min(12, p.getNombre().length())) // Ponemos el titulo del producto o solo
                                                                                // 12 caracteres
                            + "   "
                            + p.getCantidad() + "     " + p.getPrecio() + "     "// Mostramos la cantidad y el precio
                            + (p.getCantidad() * p.getPrecio())); // Y el subtotal del producto
            contentStream.newLine(); // Nueva linea
        }
        contentStream.showText("Total:            " + total); // Total
        contentStream.endText(); // Finalizamos el texto
        contentStream.close();// Cerramos el stream
        dsf = new SimpleDateFormat("yyyy-MM-dd'T'HH-mm-ss"); // Creamos el nuevo formato para guardar el archivo
        String ruta = "Ticket" + dsf.format(new Date()) + ".pdf"; // Nombre del ticket
        document.save(ruta);// Lo guardamos
        document.close(); // Cerramos el documento
        if (Desktop.isDesktopSupported()) { // Si Desktop esta disponible
            Desktop.getDesktop().open(new File(ruta)); // Abrimos el PDF
        }

    }
```

Código 5 Segmento de código que genera tickets

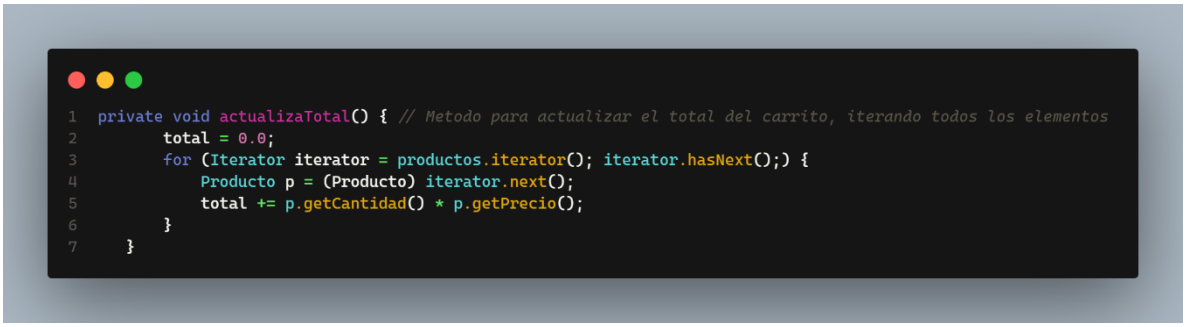
Y después los métodos que modifican los productos del carrito

```java
public void addProduct(Producto nuevoProducto) { // Agregamos un producto al carrito de compra
        boolean seActualizo = false;
        for (int i = 0; i < productos.size(); i++) {
            Producto p = (Producto) productos.get(i);
            if (nuevoProducto.getIdProducto() == p.getIdProducto()) { // Checamos si ya esta registrado
                p.setCantidad(p.getCantidad() + nuevoProducto.getCantidad()); // Actualizamos la cantidad
                seActualizo = true;
                break;
            }
        }
        if (!seActualizo) { // Si no lo encontro
            productos.add(nuevoProducto); // Lo agregamos al ArrayList
        }
        actualizaTotal(); // Actualizamos el total
    }
```

Código 6 Segmento de código que agrega un producto al carrito

```java
public void removeProduct(int idProducto, boolean reIndex) { // Quitamos el producto del IDProducto y le pasamos si
                                                             // debemos reindexar
        for (int i = 0; i < productos.size(); i++) { // Iteramos todos los productos
            Producto get = (Producto) productos.get(i);
            if (get.getIdProducto() == idProducto) { // Si el producto es el mismo
                productos.remove(get); // Lo eliminamos
                break;
            }
        }
        if (reIndex) // SI hay que reindexar
            for (int i = 0; i < productos.size(); i++) {
                Producto get = (Producto) productos.get(i);
                get.setIdProducto(i + 1); // Cambiamos el ID
            }
        actualizaTotal(); // Actualizamos el total
    }
```

Código 7 Segmento de código que elimina un producto del Carrito



```java
private void actualizaTotal() { // Metodo para actualizar el total del carrito, iterando todos los elementos
        total = 0.0;
        for (Iterator iterator = productos.iterator(); iterator.hasNext();) {
            Producto p = (Producto) iterator.next();
            total += p.getCantidad() * p.getPrecio();
        }
    }
```

Código 8 Segmento de código que actualiza el total del carrito

```java
public Producto getProducto(int idProducto) { // Obtenemos el producto dado el ID
        Producto p = null;
        for (int i = 0; i < productos.size(); i++) {
            Producto get = (Producto) productos.get(i);
            if (get.getIdProducto() == idProducto) { // SI el producto es igual al ID
                p = get;
                break;
            }
        }
        return p;// Lo regresamos
    }
```

Código 9 Segmento de código que obtiene un producto del Carrito

Y por último los métodos para serializar el objeto, donde se hace una distinción debido al ArrayList

```java
@Override
    public void writeExternal(ObjectOutput out) throws IOException { // Metodo para escribir el objeto
        out.writeDouble(total);
        out.writeInt(productos.size());
        for (Producto producto : productos) {
            out.writeObject(producto);
        }

    }

    @Override
    public void readExternal(ObjectInput in) throws IOException, ClassNotFoundException { // Metodo para leer el objeto
        this.total = in.readDouble();
        int cantidad = in.readInt();
        productos = new ArrayList<Producto>();
        for (int i = 0; i < cantidad; i++) {
            productos.add((Producto) in.readObject());
        }
    }
```

Código 10 Segmento de código que serializa el objeto

Para la gestión de las peticiones y comunicaciones entre el cliente y servidor, se realizo un gestor, donde existen 4 métodos

- Para gestionar el cliente desde el cliente
- Para gestionar el cliente desde el servidor
- Para gestionar el administrador desde el cliente
- Para gestionar el administrador desde el servidor

```java
public void gestionaClienteCliente(ObjectInputStream ois, ObjectOutputStream oos, BufferedReader br)
            throws IOException, ClassNotFoundException, CloneNotSupportedException { // Metodo para gestionar el cliente
                                                                                      // desde el cliente
        if (ois.readBoolean()) {
            CarritoDeCompra existencias = (CarritoDeCompra) ois.readObject(); // Obtenemos el carrito actual
            CarritoDeCompra carr = new CarritoDeCompra();
            boolean sePuedeSalir = false;
            Producto p, copiaProducto;
            int IdProducto;
            while (!sePuedeSalir) {
                System.out.println("" +
                        "Ingresa la operacion que quieras realizar:\n" +
                        "1)Revisar carrito de compras\n" +
                        "2)Listar productos en tienda\n" +
                        "3)Mostrar detalles del producto\n" +
                        "4)Comprar producto\n" +
                        "5)Modificar carrito\n" +
                        "6)Finalizar compra\n" +
                        "7)Salir\n" +
                        "");

                int opcion = Integer.parseInt(br.readLine());
                switch (opcion) {
                    case 1: // Revisar carrito
                        System.out.println(carr);
                        break;
                    case 2:// Listar producots en tienda
                        oos.writeInt(4);
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        existencias = (CarritoDeCompra) ois.readObject(); // Revisamos las existencias de productos
                        for (Iterator<Producto> iterator = existencias.getProductos().iterator(); iterator.hasNext();) {
                            p = iterator.next();
                            System.out.println("ID: " + p.getIdProducto() + "\tNombre: " + p.getNombre()
                                    + "\tExistencias: " + p.getCantidad());
                        }
                        break;
                    case 3:// Mostrar detalles del producto
                        oos.writeInt(1);
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        System.out.print("Ingresa el ID del producto: ");

                        IdProducto = Integer.parseInt(br.readLine());
                        oos.writeInt(IdProducto); // Enviamos el ID del producto
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        p = (Producto) ois.readObject(); // Obtenemos el producto
                        System.out.println(p);

                        int cantidad = ois.readInt();// Cantidad de imagenes del producto
                        byte[] b = new byte[1024]; // Buffer para escribir la imagen
                        for (int i = 0; i < cantidad; i++) {
                            int n;
                            long recibidos = 0, tam = ois.readLong();// Leemos el tamaño del archivo
                            File archivo = new File("Producto_" + (i + 1) + ".png"); // Creamos un archivo
                            try (DataOutputStream dos = new DataOutputStream(
                                    new FileOutputStream(archivo))) {
                                while (recibidos < tam) { // Escribimos la imagen mientras tengamos que recibir
                                    n = ois.read(b);
                                    dos.write(b, 0, n);
                                    recibidos += n;
                                }
                            }
                            muestraImagen(archivo); // Mostramos la imagen
                        }
                        break;
```

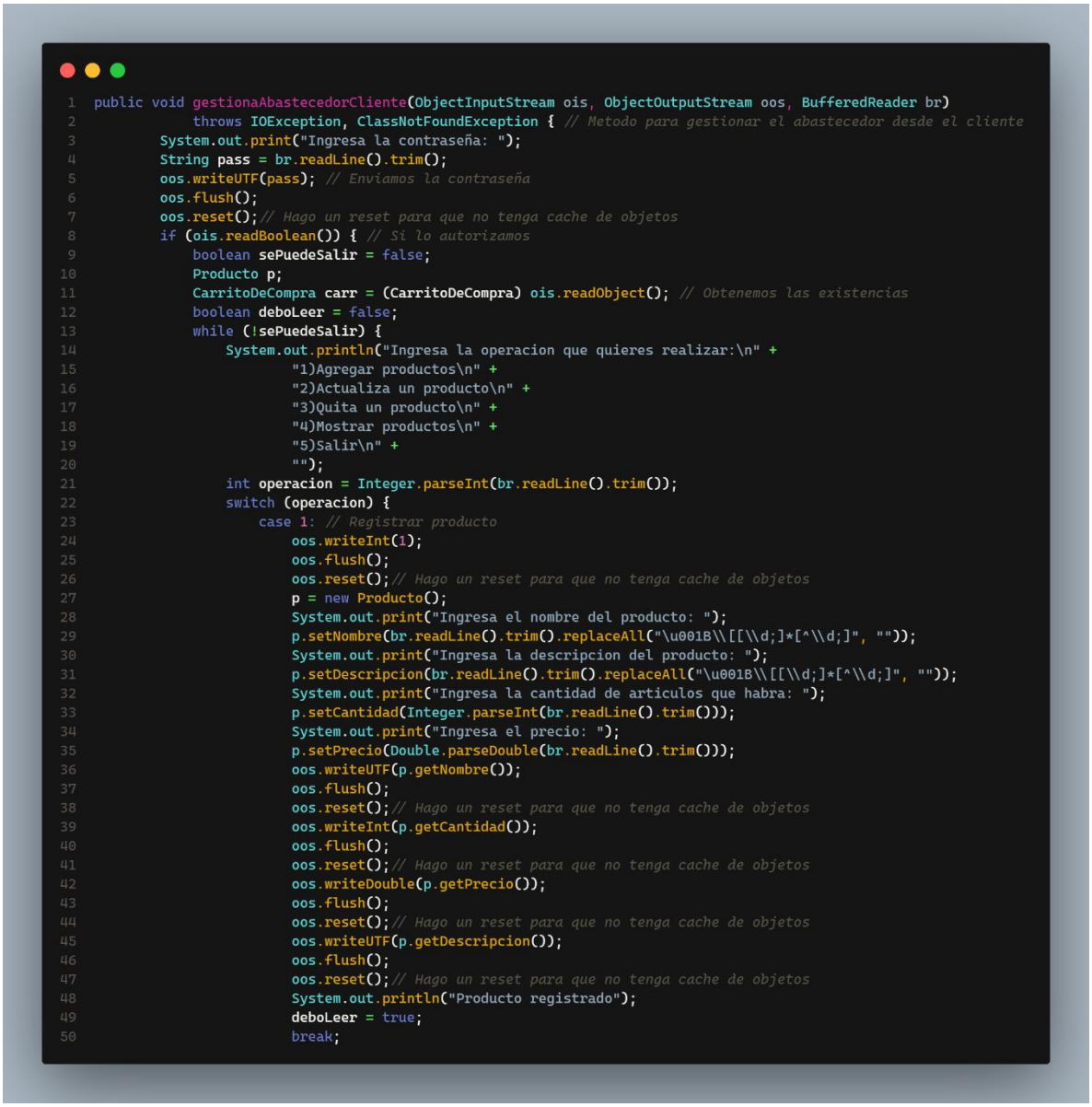
Código 11. Segmento de código que gestiona el cliente desde el cliente, parte 1

```
case 4:// Comprar producto
                        System.out.print("Ingresa el ID del producto: ");

                        IdProducto = Integer.parseInt(br.readLine());
                        p = carr.getProducto(IdProducto); // Obtenemos el producto para revisar si ya esta en el carrito
                        if (p == null) { // Si no existe entonces hay que tomarlo de las existencias
                            p = existencias.getProducto(IdProducto); // Lo buscamos en las existencias
                            if (!(p == null)) { // Si lo encontro entonces lo clonamos
                                p = (Producto) p.clone();// Lo clonamos
                                p.setCantidad(0); // Ponemos la cantidad de 0 para que lo cambiemos despues
                            }
                        }
                        if (!(p == null)) { // Si el producto es diferente de null
                            copiaProducto = (Producto) p.clone(); // Lo clonamos
                            System.out.print("Ingresa la cantidad del producto: ");
                            copiaProducto.setCantidad(Integer.parseInt(br.readLine())); // Ponemos la cantidad desde el
                                                                                        // teclado
                            oos.writeInt(2); // Le decimos que queremos comprar un objeto
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            copiaProducto.setCantidad(p.getCantidad() + copiaProducto.getCantidad()); // Lo agregamos a
                                                                                                      // la copia
                            oos.writeObject(copiaProducto); // Lo enviamos
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            if (ois.readBoolean()) { // Significa que si hay existencias suficientes
                                copiaProducto.setCantidad(copiaProducto.getCantidad() - p.getCantidad());
                                carr.addProduct(copiaProducto); // Lo agregamos al carrito
                                System.out.println("Articulo agregado al carrito");

                            } else { // No hay existencias
                                System.out.println("No hay suficientes existencias");
                            }
                        } else {
                            System.out.println("El producto no existe");
                        }
                        break;
                    case 5:// Modificar carrito
                        boolean sePuedeSalir2 = false;
                        while (!sePuedeSalir2) { // Entramos en el ciclo para modificar el carrito
                            System.out.println("" +
                                    "Ingresa la operacion que quieras realizar:\n" +
                                    "1)Mostrar carrito\n" +
                                    "2)Modifcar cantidad de un producto\n" +
                                    "3)Eliminar producto\n" +
                                    "4)Salir\n" +
                                    "");

                            int opcion2 = Integer.parseInt(br.readLine());
                            switch (opcion2) {
                                case 1:// Mostrar carrito
                                    System.out.println(carr);

                                    break;
                                case 2:// Modificar cantidad de un producto
                                    System.out.print("Ingresa el ID del producto: ");

                                    IdProducto = Integer.parseInt(br.readLine());
                                    p = carr.getProducto(IdProducto); // Buscamos el producto que queremos modificar

                                    if (!(p == null)) { // Si el objeto es diferente de nulo
                                        p = (Producto) p.clone();
                                        System.out.print("Ingresa la cantidad del producto: ");

```

Código 12 Segmento de código que gestiona el cliente desde el cliente, parte 2

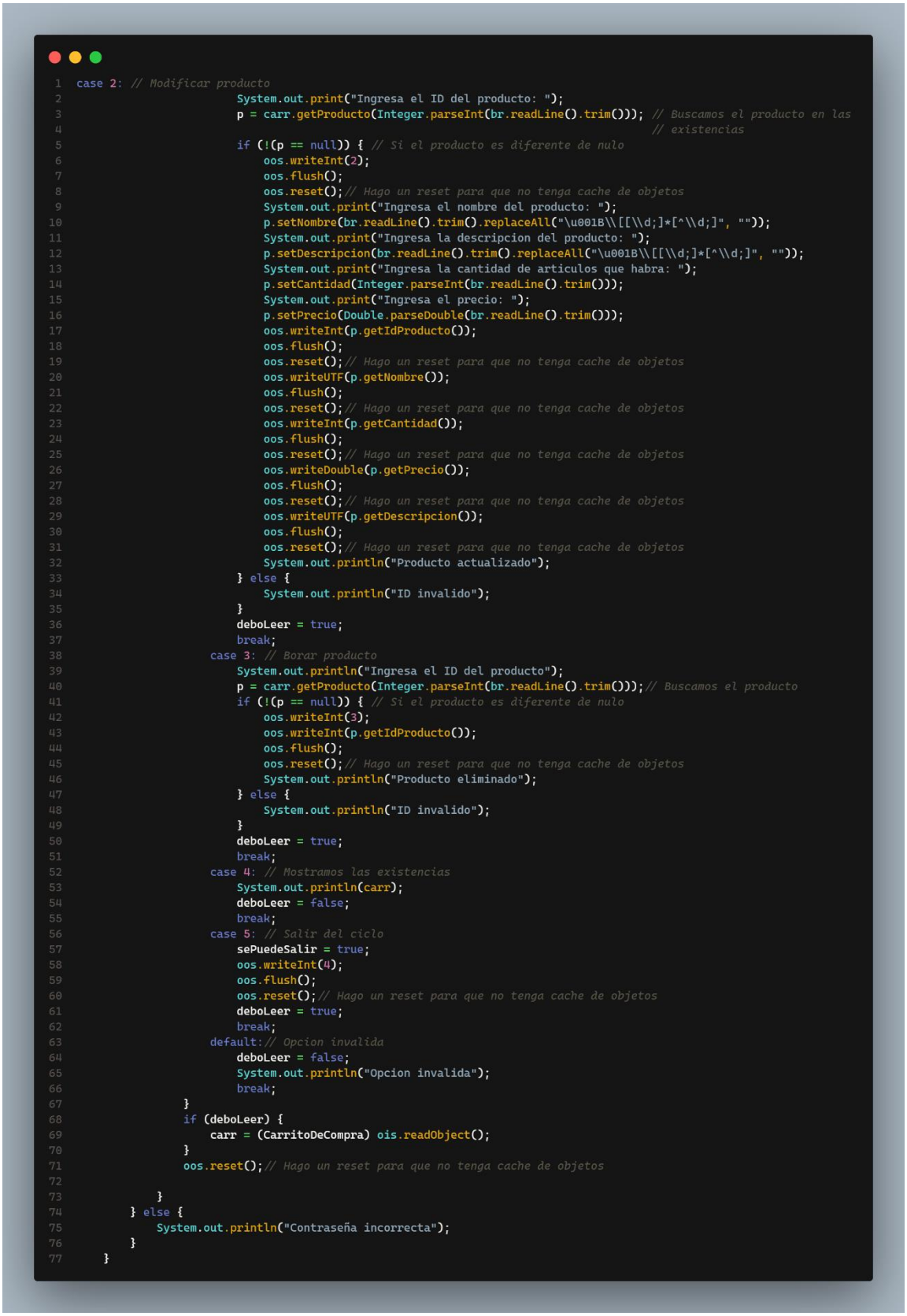
```
                                    p.setCantidad(Integer.parseInt(br.readLine()));
                                    oos.writeInt(2);
                                    oos.flush();
                                    oos.reset();// Hago un reset para que no tenga cache de objetos
                                    oos.writeObject(p);
                                    oos.flush();
                                    oos.reset();// Hago un reset para que no tenga cache de objetos
                                    if (ois.readBoolean()) { // Si hay existencias
                                        carr.removeProduct(p.getIdProducto(), false); // Lo borramos
                                        carr.addProduct(p);// Y lo agregamos
                                        System.out.println("Carrito modificado correctamente");
                                    } else {
                                        System.out.println("No hay suficientes existencias");
                                    }
                                } else {
                                    System.out.println("El producto no esta en el carrito");
                                }
                                break;
                            case 3:// Elimar producto
                                System.out.print("Ingresa el ID del producto: ");

                                IdProducto = Integer.parseInt(br.readLine());
                                p = carr.getProducto(IdProducto); // Lo buscamos en el carrito
                                if (!(p == null)) {
                                    carr.removeProduct(IdProducto, false); // Lo quitamos
                                    System.out.println("Producto eliminado del carrito");
                                } else {
                                    System.out.println("El producto no esta en el carrito");
                                }
                                break;
                            case 4: // Salir
                                sePuedeSalir2 = true;
                                break;
                            default:
                                System.out.println("Opcion invalida");
                                break;
                        }
                    }
                    break;
                case 6: // Finalizar compra
                    oos.writeInt(3);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                    oos.writeObject(carr);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                    if (ois.readBoolean()) { // Si paso la compra
                        carr.generateTicket(); // Generamos el ticket
                        carr = new CarritoDeCompra(); // Creamos un nuevo carrito de compra
                        System.out.println("Compra existosa");
                    } else {
                        System.out.println("Hubo un error con la compra");
                    }
                    break;
                case 7:
                    // Finalizar compra
                    sePuedeSalir = true;
                    oos.writeInt(5);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                    break;
                default:
                    System.out.println("Opcion invalida");
                    break;
            }
            oos.reset();// Hago un reset para que no tenga cache de objetos

        }
    } else {
        System.out.println("Pues no puedes comprar gg");
    }
}
```

Código 13 Segmento de código que gestiona el cliente desde el cliente, parte 3



```java
public void gestionaAbastecedorCliente(ObjectInputStream ois, ObjectOutputStream oos, BufferedReader br)
            throws IOException, ClassNotFoundException { // Metodo para gestionar el abastecedor desde el cliente
        System.out.print("Ingresa la contraseña: ");
        String pass = br.readLine().trim();
        oos.writeUTF(pass); // Enviamos la contraseña
        oos.flush();
        oos.reset();// Hago un reset para que no tenga cache de objetos
        if (ois.readBoolean()) { // Si lo autorizamos
            boolean sePuedeSalir = false;
            Producto p;
            CarritoDeCompra carr = (CarritoDeCompra) ois.readObject(); // Obtenemos las existencias
            boolean deboLeer = false;
            while (!sePuedeSalir) {
                System.out.println("Ingresa la operacion que quieres realizar:\n" +
                        "1)Agregar productos\n" +
                        "2)Actualiza un producto\n" +
                        "3)Quita un producto\n" +
                        "4)Mostrar productos\n" +
                        "5)Salir\n" +
                        "");
                int operacion = Integer.parseInt(br.readLine().trim());
                switch (operacion) {
                    case 1: // Registrar producto
                        oos.writeInt(1);
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        p = new Producto();
                        System.out.print("Ingresa el nombre del producto: ");
                        p.setNombre(br.readLine().trim().replaceAll("\u001B\\[[\\d;]*[^\\d;]", ""));
                        System.out.print("Ingresa la descripcion del producto: ");
                        p.setDescripcion(br.readLine().trim().replaceAll("\u001B\\[[\\d;]*[^\\d;]", ""));
                        System.out.print("Ingresa la cantidad de articulos que habra: ");
                        p.setCantidad(Integer.parseInt(br.readLine().trim()));
                        System.out.print("Ingresa el precio: ");
                        p.setPrecio(Double.parseDouble(br.readLine().trim()));
                        oos.writeUTF(p.getNombre());
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        oos.writeInt(p.getCantidad());
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        oos.writeDouble(p.getPrecio());
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        oos.writeUTF(p.getDescripcion());
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        System.out.println("Producto registrado");
                        deboLeer = true;
                        break;
```

Código 14 Segmento de código que gestiona el abastecedor desde el cliente, parte 1



```
case 2: // Modificar producto
                        System.out.print("Ingresa el ID del producto: ");
                        p = carr.getProducto(Integer.parseInt(br.readLine().trim())); // Buscamos el producto en las
                                                                                      // existencias
                        if (!(p == null)) { // Si el producto es diferente de nulo
                            oos.writeInt(2);
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            System.out.print("Ingresa el nombre del producto: ");
                            p.setNombre(br.readLine().trim().replaceAll("\u001B\\[[\\d;]*[^\\d;]", ""));
                            System.out.print("Ingresa la descripcion del producto: ");
                            p.setDescripcion(br.readLine().trim().replaceAll("\u001B\\[[\\d;]*[^\\d;]", ""));
                            System.out.print("Ingresa la cantidad de articulos que habra: ");
                            p.setCantidad(Integer.parseInt(br.readLine().trim()));
                            System.out.print("Ingresa el precio: ");
                            p.setPrecio(Double.parseDouble(br.readLine().trim()));
                            oos.writeInt(p.getIdProducto());
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            oos.writeUTF(p.getNombre());
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            oos.writeInt(p.getCantidad());
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            oos.writeDouble(p.getPrecio());
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            oos.writeUTF(p.getDescripcion());
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            System.out.println("Producto actualizado");
                        } else {
                            System.out.println("ID invalido");
                        }
                        deboLeer = true;
                        break;
                    case 3: // Borar producto
                        System.out.println("Ingresa el ID del producto");
                        p = carr.getProducto(Integer.parseInt(br.readLine().trim()));// Buscamos el producto
                        if (!(p == null)) { // Si el producto es diferente de nulo
                            oos.writeInt(3);
                            oos.writeInt(p.getIdProducto());
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                            System.out.println("Producto eliminado");
                        } else {
                            System.out.println("ID invalido");
                        }
                        deboLeer = true;
                        break;
                    case 4: // Mostramos las existencias
                        System.out.println(carr);
                        deboLeer = false;
                        break;
                    case 5: // Salir del ciclo
                        sePuedeSalir = true;
                        oos.writeInt(4);
                        oos.flush();
                        oos.reset();// Hago un reset para que no tenga cache de objetos
                        deboLeer = true;
                        break;
                    default:// Opcion invalida
                        deboLeer = false;
                        System.out.println("Opcion invalida");
                        break;
                }
                if (deboLeer) {
                    carr = (CarritoDeCompra) ois.readObject();
                }
                oos.reset();// Hago un reset para que no tenga cache de objetos

            }
        } else {
            System.out.println("Contraseña incorrecta");
        }
    }
```

Código 15 Segmento de código que gestiona el abastecedor desde el cliente, parte 2

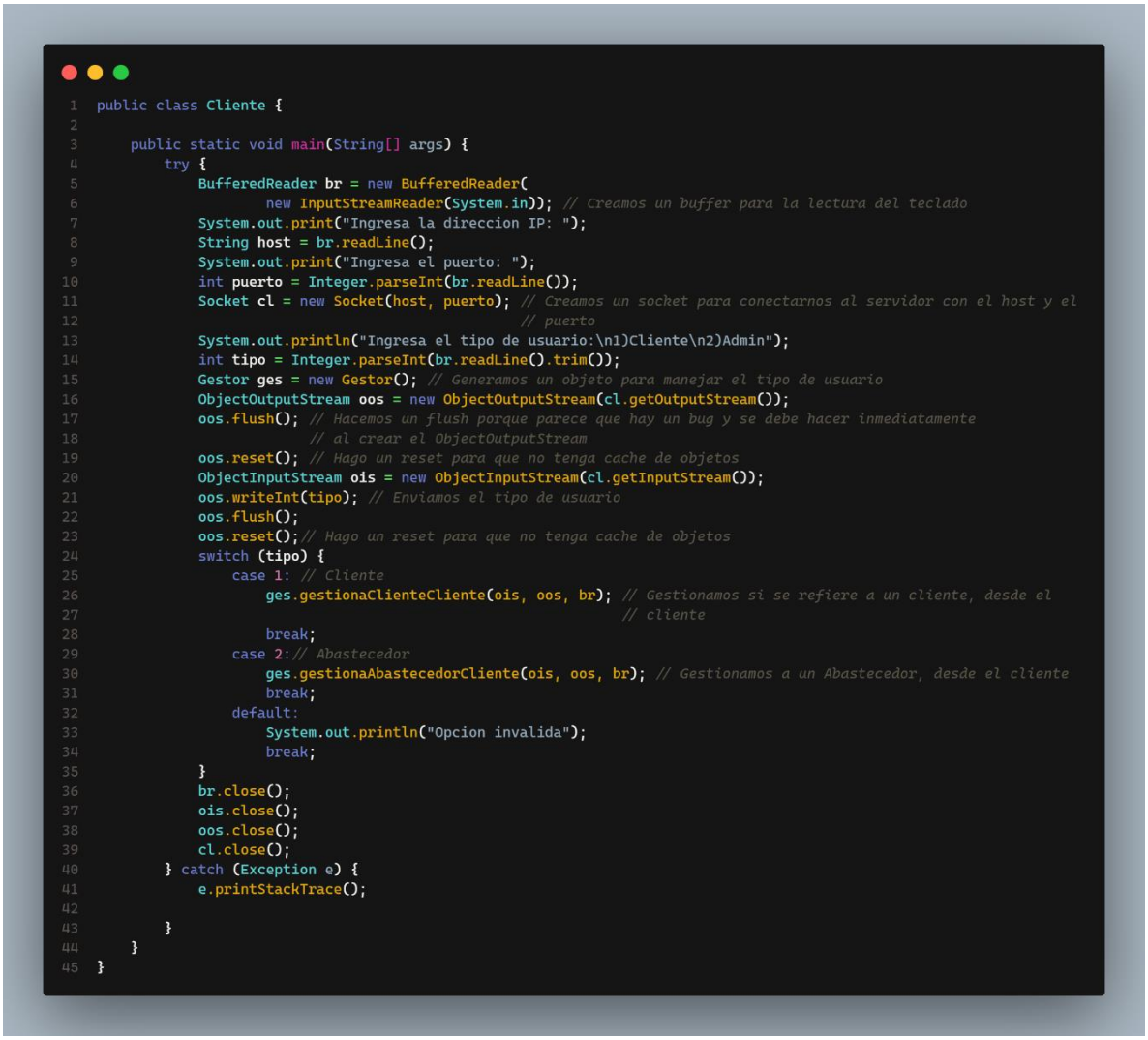
```java
public void gestionaClienteServidor(ObjectInputStream ois, ObjectOutputStream oos)
        throws IOException, ClassNotFoundException, CloneNotSupportedException {
    CarritoDeCompra carr = getCarritoCompra();
    Producto p;
    int idProducto;
    oos.writeObject(carr);
    oos.flush();
    oos.reset();// Hago un reset para que no tenga cache de objetos
    boolean sePuedeSalir = false;
    while (!sePuedeSalir) {
        int operacion = ois.readInt();
        switch (operacion) {
            case 1:// Mostrar detalles del producto
                idProducto = ois.readInt();
                p = carr.getProducto(idProducto);
                oos.writeObject(p);
                oos.flush();
                oos.reset();// Hago un reset para que no tenga cache de objetos
                ArrayList<File> archivos = p.obtenerImagenes();
                oos.writeInt(archivos.size());
                oos.flush();
                oos.reset();// Hago un reset para que no tenga cache de objetos
                byte[] b = new byte[1024];
                for (File file : archivos) {
                    int n;
                    long enviados = 0, tam = file.length();
                    oos.writeLong(tam);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                    try (DataInputStream dis = new DataInputStream(new FileInputStream(file))) {
                        while (enviados < tam) {
                            n = dis.read(b);
                            oos.write(b, 0, n);
                            oos.flush();
                            oos.reset(); // Hago un reset para que no tenga cache de objetos
                            enviados += (long) n;
                        }
                    }
                }
                break;
            case 2: // Comprobar existencias
                p = (Producto) ois.readObject();
                if (carr.getProducto(p.getIdProducto()).getCantidad() ≥ p.getCantidad()) {
                    oos.writeBoolean(true);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                } else {
                    oos.writeBoolean(false);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                }
                break;
            case 3:// Comprar nuevo carrito
                CarritoDeCompra nuevo = (CarritoDeCompra) ois.readObject();
                CarritoDeCompra propuesta = (CarritoDeCompra) carr.clone();
                boolean siCompro = true;
                for (Iterator<Producto> iterator = nuevo.getProductos().iterator(); iterator.hasNext();) {
                    p = iterator.next();
                    if (carr.getProducto(p.getIdProducto()).getCantidad() < p.getCantidad()) {
                        oos.writeBoolean(false);
                        oos.flush();
                        oos.reset(); // Hago un reset para que no tenga cache de objetos
                        siCompro = false;
                        break;
                    } else {
                        Producto actualiza = (Producto) carr.getProducto(p.getIdProducto()).clone();
                        actualiza.setCantidad(-p.getCantidad());
                        propuesta.addProduct(actualiza);
                    }
                }
                if (siCompro) {
                    carr.setProductos(propuesta.getProductos());
                    carr.setTotal(0.0);

                    oos.writeBoolean(true);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                } else {
                    oos.writeBoolean(false);
                    oos.flush();
                    oos.reset();// Hago un reset para que no tenga cache de objetos
                }
                break;
            case 4:// Devuelve productos
                oos.writeObject(carr);
                oos.flush();
                oos.reset();// Hago un reset para que no tenga cache de objetos
                break;
            case 5:
                sePuedeSalir = true;
                break;
        }
        oos.reset();// Hago un reset para que no tenga cache de objetos
    }

    saveCarritoCompra(carr);
}
```

Código 16 Segmento de código que gestiona el cliente desde el servidor

```java
public void gestionaAbastecedorServidor(ObjectInputStream ois, ObjectOutputStream oos)
        throws IOException { // Manejamos el abastecedor desde el servidor
    CarritoDeCompra carr = getCarritoCompra();
    Producto p;
    oos.writeObject(carr);
    oos.flush();
    oos.reset(); // Hago un reset para que no tenga cache de objetos
    boolean sePuedeSalir = false;
    while (!sePuedeSalir) {// Ciclamos mientras no se deba salir
        int operacion = ois.readInt();
        switch (operacion) {
            case 1: // Agrega producto
                p = new Producto(); // Recibimos las propiedades del nuevo producto
                p.setIdProducto(carr.getProductos().size() + 1);
                p.setNombre(ois.readUTF());
                p.setCantidad(ois.readInt());
                p.setPrecio(ois.readDouble());
                p.setDescripcion(ois.readUTF());
                carr.addProduct(p); // Lo agregamos
                carr.setTotal(0.0); // Cambiamos el total a 0
                break;
            case 2:// Actualiza producto
                p = new Producto();
                p.setIdProducto(ois.readInt());
                p.setNombre(ois.readUTF());
                p.setCantidad(ois.readInt());
                p.setPrecio(ois.readDouble());
                p.setDescripcion(ois.readUTF());
                carr.removeProduct(p.getIdProducto(), true); // Lo borramos y reindexamos
                carr.addProduct(p); // Agregamos producto
                carr.setTotal(0.0); // Cambiamos el total a 0
                break;
            case 3:// Quita producto
                carr.removeProduct(ois.readInt(), true); // Lo quitamos de la lista
                carr.setTotal(0.0);
                break;
            case 4:// Salir
                sePuedeSalir = true;
                break;
        }
        saveCarritoCompra(carr);

        oos.writeObject(carr);
        oos.flush();// Enviamos el carrito actualizado
        oos.reset();// Hago un reset para que no tenga cache de objetos
    }

}
```

Código 17 Segmento de código que gestiona el abastecedor desde el servidor

Y por último creamos un objeto para el servidor y para el cliente, que se encargue de iniciar los procesos para cada uno.



```java
public class Cliente {

    public static void main(String[] args) {
        try {
            BufferedReader br = new BufferedReader(
                    new InputStreamReader(System.in)); // Creamos un buffer para la lectura del teclado
            System.out.print("Ingresa la direccion IP: ");
            String host = br.readLine();
            System.out.print("Ingresa el puerto: ");
            int puerto = Integer.parseInt(br.readLine());
            Socket cl = new Socket(host, puerto); // Creamos un socket para conectarnos al servidor con el host y el
                                                  // puerto
            System.out.println("Ingresa el tipo de usuario:\n1)Cliente\n2)Admin");
            int tipo = Integer.parseInt(br.readLine().trim());
            Gestor ges = new Gestor(); // Generamos un objeto para manejar el tipo de usuario
            ObjectOutputStream oos = new ObjectOutputStream(cl.getOutputStream());
            oos.flush(); // Hacemos un flush porque parece que hay un bug y se debe hacer inmediatamente
                         // al crear el ObjectOutputStream
            oos.reset(); // Hago un reset para que no tenga cache de objetos
            ObjectInputStream ois = new ObjectInputStream(cl.getInputStream());
            oos.writeInt(tipo); // Enviamos el tipo de usuario
            oos.flush();
            oos.reset();// Hago un reset para que no tenga cache de objetos
            switch (tipo) {
                case 1: // Cliente
                    ges.gestionaClienteCliente(ois, oos, br); // Gestionamos si se refiere a un cliente, desde el
                                                              // cliente
                    break;
                case 2:// Abastecedor
                    ges.gestionaAbastecedorCliente(ois, oos, br); // Gestionamos a un Abastecedor, desde el cliente
                    break;
                default:
                    System.out.println("Opcion invalida");
                    break;
            }
            br.close();
            ois.close();
            oos.close();
            cl.close();
        } catch (Exception e) {
            e.printStackTrace();

        }
    }
}
```

Código 18 Segmento de código del cliente

```java
public class Servidor {

    public static void main(String[] args) {
        while (true) {
            try {
                System.out.println("Esperando conexion en el puerto 3070");
                Gestor ges = new Gestor();// Creamos el objeto gestor de usuarios
                ServerSocket s = new ServerSocket(3070); // Generamos un Servidor en el puerto 3070
                Socket cl = s.accept(); // Esperamos conexion
                System.out.println("Conexion aceptada");
                System.out.println("Conexión establecida desde"
                        + cl.getInetAddress() + ":" + cl.getPort()); // Mostramos la informacion de la conexion
                ObjectOutputStream oos = new ObjectOutputStream(cl.getOutputStream());
                oos.flush(); // Hacemos flush por una cosa de Java
                oos.reset();// Hago un reset para que no tenga cache de objetos
                ObjectInputStream ois = new ObjectInputStream(cl.getInputStream());
                int tipo = ois.readInt(); // Recibimos el tipo de usuario del cliente
                switch (tipo) {
                    case 1:// Cliente
                        oos.writeBoolean(true); // Si autorizado
                        ges.gestionaClienteServidor(ois, oos); // Gestiona Cliente desde el servidor
                        oos.flush();
                        oos.reset();
                        break;
                    case 2:// Abastecedor
                        String pass = ois.readUTF();// Recibimos la contraseña
                        if (!(pass == null) && pass.equals("12345")) { // Solo la comparamos con 12345
                            oos.writeBoolean(true); // Le devolvemos que si paso
                            oos.flush();
                            oos.reset(); // Hago un reset para que no tenga cache de objetos
                            ges.gestionaAbastecedorServidor(ois, oos); // Gestiona Abastacedor desde el Servidor
                        } else {
                            oos.writeBoolean(false); // No autorizado
                            oos.flush();
                            oos.reset();// Hago un reset para que no tenga cache de objetos
                        }
                        break;
                    default:
                        oos.writeBoolean(false); // No autorizado
                        oos.flush();
                        oos.reset();
                        break;
                }
                ois.close();
                oos.close();
                cl.close();
                s.close();
                TimeUnit.SECONDS.sleep(5);// Sleep de 5 segundos
            } catch (Exception e) {
                e.printStackTrace();
                System.out.println("Pues se murio");
                try {
                    TimeUnit.SECONDS.sleep(10); // Sleep de 10 segundos, porque puede ser que el puerto siga ocupado
                } catch (Exception e2) {
                    e2.printStackTrace();
                }
            }
        }
    }
}
```

Código 19 Segmento de código del Servidor

Y finalmente solo las pruebas

Código 20 Ventana de pruebas, parte 1

Código 21 Ventana de pruebas, parte 2

Cuestionario

1. ¿Qué es serialización?
    a. La serialización es el proceso de codificación de un objeto para sus diferentes usos
2. ¿En qué difiere del marshalling?
    a. Se refiere al proceso de codificación para el transporte de datos
3. ¿Por qué es importante la serialización?
    a. Porque tiene diferentes usos
        i. Un método de persistencia de objetos

    ii. Un método para la distribución de objetos
    iii. Un método para detectar cambios en variables en el tiempo.
4. Cuatro ejemplos de tecnologías en las que se haga uso de la serialización.
    a. Java
    b. Python
    c. PHP
    d. Perl
5. ¿Qué limita el tamaño máximo del archivo a enviar?
    a. El limite del paquete del protocolo

Conclusiones

Referencias

[1] Oracle, «Object Serialization: Frequently Asked Questions,» Oracle, [En línea]. Available: https://www.oracle.com/java/technologies/javase/serializationfaq-jsp.html. [Último acceso: 4 Junio 2022].

[2] Apache PDFBox, «Apache PDFBox | A Java PDF Library,» Apache PDFBox, 5 Mayo 2022. [En línea]. Available: https://pdfbox.apache.org/. [Último acceso: 4 Junio 2022].

[3] Oracle, «Desktop (Java Platform SE 8),» Oracle, [En línea]. Available: https://docs.oracle.com/javase/8/docs/api/java/awt/Desktop.html#open(java.io.File). [Último acceso: 4 Junio 2022].

[4] «Getting Started | Building Java Projects with Maven,» Spring, [En línea]. Available: https://spring.io/guides/gs/maven/#scratch. [Último acceso: 4 Junio 2022].