```
package Examen.luiseldelbar;

import java.io.BufferedWriter;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileWriter;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Scanner;

public class Bar {

	public static ArrayList<Reserva> reservas = new ArrayList<Reserva>();
	public static final String FICHERO = "reservas.bin";

	public static int leerInt() {
		Scanner teclado = new Scanner(System.in);
		while (true) {
			try {
				return teclado.nextInt();
			} catch (Exception e) {
				System.out.println("Número incorrecto");
				teclado.nextLine();
			}
		}
	}

	public static int menu() {
		int op = 0;
		while (op < 1 || op > 5) {
			System.out.println("1. Añadir reserva");
			System.out.println("2. Mostrar todas las reservas");
			System.out
					.println("3. Mostrar listado de las reservas con catering");
			System.out.println("4. Anular reserva");
			System.out.println("5. Salir");
			op = leerInt();
		}
		return op;
	}

	public static Reserva buscar(int id) {
		Iterator<Reserva> it = reservas.iterator();
		while (it.hasNext()) {
			Reserva r = it.next();
```

```
            if (r.getId() == id)
                return r;
        }
        return null;
    }

    public static void anadirReserva() {
        Scanner teclado = new Scanner(System.in);
        int op = 0;
        while (op != 1 && op != 2) {
            System.out.println(
                    "¿Qué tipo de reserva quiere realizar?\n1. Sólo local\n2.
Con catering y barra libre");
            op = leerInt();
        }
        System.out.print("Año: ");
        int año = leerInt();
        int semana = 0;
        while (!Reserva.esSemanaValida(semana)) {
            System.out.print("Semana: ");
            semana = leerInt();
        }

        if (buscar(Reserva.generarId(año, semana)) == null) {

            System.out.print("Cliente: ");
            String cliente = teclado.nextLine();
            if (op == 2) {
                int personas = 0;
                while (!ReservaCateringBarraLibre.esPersonasValido(personas)) {
                    System.out.print("Número de personas: ");
                    personas = leerInt();
                }
                System.out.print("Horas contratadas: ");
                int horas = leerInt();
                reservas.add(new ReservaCateringBarraLibre(año, semana,
cliente,
                        personas, horas));
            } else {
                reservas.add(new ReservaLocal(año, semana, cliente));
            }
        } else
            System.out.println("Esa fecha ya estña reservada.");
    }

    public static void guardarFicheroTexto() {
        Scanner teclado = new Scanner(System.in);
        System.out.print("Nombre del fichero: ");
        String f = teclado.nextLine();
```

```
        BufferedWriter bw = null;
        try {
            bw = new BufferedWriter(new FileWriter(f));
            Iterator<Reserva> it = reservas.iterator();
            while (it.hasNext()) {
                bw.write(it.next().toString() + "\n");
            }
        } catch (IOException e) {
            System.out.println("Error escribiendo el fichero de texto");
        } finally {
            try {
                if (bw != null)
                    bw.close();
            } catch (IOException e) {
                System.out.println("Error cerrando el fichero");
            }
        }
    }

    public static void guardarDatos() {
        ObjectOutputStream oos = null;
        try {
            oos = new ObjectOutputStream(new FileOutputStream(FICHERO));
            oos.writeObject(reservas);
        } catch (IOException e) {
            System.out.println("Error guardando datos");

        } finally {
            try {
                if (oos != null)
                    oos.close();
            } catch (IOException e) {
                System.out.println("Error cerrando el fichero");
            }
        }
    }

    public static void recuperarDatos() {
        ObjectInputStream ois = null;
        try {
            ois = new ObjectInputStream(new FileInputStream(FICHERO));
            reservas = (ArrayList<Reserva>) ois.readObject();
        } catch (FileNotFoundException e) {

        } catch (ClassNotFoundException e) {
            System.out.println("Error en los datos");
        } catch (IOException e) {
            System.out.println("Error leyendo los datos");
        } catch (Exception e) {
```

```
            System.out.println("Error en los datos");
        } finally {
            try {
                if (ois != null)
                    ois.close();
            } catch (IOException e) {
                System.out.println("Error cerrando el fichero");
            }
        }
    }

    public static void mostrarReservas() {
        Iterator<Reserva> it = reservas.iterator();
        while (it.hasNext()) {
            System.out.println(it.next());

        }
        int op = 0;
        while (op != 1 && op != 2) {
            System.out.println(
                    "¿Desea guardar el listado en un fichero?\n1. Si\n2. No");
            op = leerInt();
        }
        if (op == 1) {
            guardarFicheroTexto();
        }

    }

    public static void mostrarReservasCatering() {
        Iterator<Reserva> it = reservas.iterator();
        while (it.hasNext()) {
            Reserva r = it.next();
            if (r instanceof ReservaCateringBarraLibre)
                System.out.println(r);
        }
    }

    public static void anularReserva() {
        System.out.print("Año: ");
        int año = leerInt();
        int semana = 0;
        while (!Reserva.esSemanaValida(semana)) {
            System.out.print("Semana: ");
            semana = leerInt();
        }

        Reserva r = buscar(Reserva.generarId(año, semana));
        if (r == null)
```

```
                System.out
                        .println("No he encontrado ninguna reserva para esa fecha");
            else {
                reservas.remove(r);
                System.out.println("La reserva ha sido anulada correctamente");
            }
        }

        public static void main(String[] args) {

            recuperarDatos();
            int op = menu();
            while (op != 5) {
                switch (op) {
                    case 1 :
                        anadirReserva();
                        break;
                    case 2 :
                        mostrarReservas();
                        break;
                    case 3 :
                        mostrarReservasCatering();
                        break;
                    case 4 :
                        anularReserva();
                        break;
                }
                op = menu();
            }
            guardarDatos();
        }

    }
```

```java
package Examen.luiseldelbar;

import java.io.Serializable;

public abstract class Reserva implements Serializable {
	protected int id;
	protected String cliente;

	public Reserva(int año, int semana, String cliente) {
		id = generarId(año, semana);
		this.cliente = cliente;
	}

	public static int generarId(int año, int semana) {
		return año * 100 + semana;
	}

	public static boolean esSemanaValida(int semana) {
		return semana >= 1 && semana <= 53;
	}

	public int getId() {
		return id;
	}

	@Override
	public String toString() {
		return id + "\t" + cliente;
	}

	public abstract double calcularPrecio();

}
```

```
package Examen.luiseldelbar;

public class ReservaCateringBarraLibre extends Reserva {

	private int nPersonas;
	private int horas;
	private static final double PRECIO_PERSONA = 25;
	private static final int HORAS_MIN = 2;
	private static final double HORA_ADC = 5;
	private static final double PERSONAS_MIN = 15;
	private static final double PERSONAS_MAX = 30;

	public ReservaCateringBarraLibre(int año, int semana, String cliente,
			int nPersonas, int horas) {
		super(año, semana, cliente);
		this.nPersonas = nPersonas;
		if (horas < 2)
			this.horas = 2;
		else
			this.horas = horas;
	}

	public static boolean esPersonasValido(int nPersonas) {
		return nPersonas >= PERSONAS_MIN && nPersonas <=
PERSONAS_MAX;
	}

	@Override
	public double calcularPrecio() {
		return nPersonas * PRECIO_PERSONA
				+ (horas - HORAS_MIN) * HORA_ADC * nPersonas;
	}

	@Override
	public String toString() {
		return super.toString() + "\t" + nPersonas + " personas\t" + horas
				+ " horas\t" + calcularPrecio() + " €";

	}

}
```

```
package Examen.luiseldelbar;

public class ReservaLocal extends Reserva {

	private static final double PRECIO = 250;

	public ReservaLocal(int año, int semana, String cliente) {
		super(año, semana, cliente);
	}

	@Override
	public double calcularPrecio() {
		return PRECIO;
	}

	@Override
	public String toString() {
		return super.toString() + "\t" + calcularPrecio() + " €";
	}
}
```