Práctica de Programación Orientada a Objetos

Curso 2018 / 2019

Juan Carlos Amor Gutiérrez

jamor3@alumno.uned.es

Telf. 653.81.99.37

Análisis de la aplicación:

He dicidido crear dos herencias, una de la clase Clientes y otra de la clase Atracciones.

Se han creado un ArrayList para guardar los clientes que entran al parque mayores de 3 años, con una superclase Cliente, de la que heredan Adulto, Nino y Senior. He usado el polimorfismo para el ArrayList, así tengo todos los clientes en un mismo ArrayList para las estadísticas.

He creado otro ArrayList de Empleado para guardar todos los trabajadores del parque, he calculado el sueldo de cada puesto para grabarlo como un atributo de la clase.

Otro ArrayList para las atracciones, de la superclase Atracciones, de la que heredan las clases que son los tipos de atracciones que hay en el parque. Cada una con su valor en los campos de la superclase.

La clase que lleva todo el peso de la práctica es la clase Menu, crea los objetos empleados y atracciones. Deja a la clase Entrada el generar los objetos clientes después de pedirle todos los datos de cada entrada.

MENU PARQUE DE ATRACCIONES

==========

[1]. Menú Entradas"

[2]. Menú Empleados"

[3]. Menú Datos Estadísticos"

[4]. Menú Atracciones"

[5]. Salir"

A través de este menú he dividido las cuatro casuísticas que se presentan en el parque.

El bloque entradas con la clase Clientes y sus herederas.

El bloque empleados con un menú de agregar, borrar y listar los empleados.

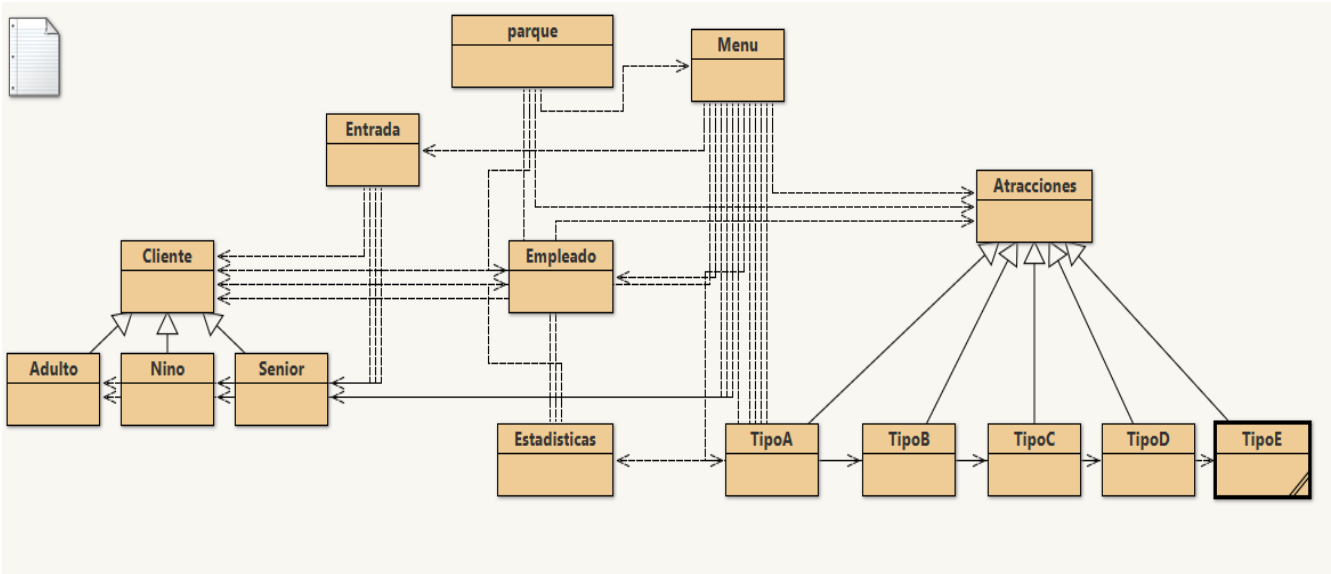
El bloque Atracciones con los distintos tipos de atracciones y con las características concretas de cada tipo, así como el número de atracciones funcionando, para calcular el número de trabajadores necesarios, como su puesto y salario.

El bloque Estadístico se mueve a través de los ArrayList para mostrar la información, así como para calcular empleados, sueldos, etc.., dependiendo de las atracciones que estén funcionando en ese momento.

Los descuentos se lo he aplicado en las entradas según la práctica.

La entrada familiar si elegía VIP se la adjudicaba automáticamente a todas las entradas relacionadas con la familia y lo mismo si no lo seleccionaba, no podía ninguno de la familia elegirlo.

Diagrama de clases:



Descripción de la clases:

La clase Menu

Ha sido la encargada que crear los objetos tanto de empleados como de las atracciones. Métodos:

public void menuPrincipal()

El menú de texto para principal del programa.

public void menuEntradasPrincipal()

En este menú se pregunta si quiere la entrada para el día o venta anticipada.

public void menuEmitirEntradas()

Este menú pregunta si la entrada es individual o familiar y creamos el objeto Entrada para generarla.

public void limpiarPantalla()

Se añaden líneas en blanco para borrar la pantalla

public void menuEmpleados()

El menú de agregar, borrar y listar de los empleados.

public void agregarEmpleado()

Se piden los datos del nuevo empleado y se le añade el sueldo dependiendo del puesto de trabajo. Se graban en un ArrayList de empleados.

public void borrarEmpleado()

Para borrar un empleado del ArrayList te pide que introduzcas el Dni y si lo encuentra te pide confirmación para su borrado.

public void listarEmpleados()

Te imprime por pantalla todos los empleados con sus atributos.

public void menuAtracciones()

Un menú para generar las atracciones, borrarlas o listarlas.

public void agregarAtracciones()

Se general las atracciones según su tipo y el número de ellas que están funcionando. Y se añaden al ArrayList de atracciones.

public void listarAtracciones()

Se imprimen por pantalla las atracciones y su número. Así como si admiten pase VIP o para que tipo de clientes puede montar en ellas. También nos da la cantidad de empleados de atención al cliente, como de relaciones públicas que se necesitan, con las atracciones que están funcionando. Nos da el gasto de ese personal mensual y anual.

public void borrarAtracciones()

Borra las atracciones del ArrayList.

public boolean preguntasVarias(String pregunta)

Este método lo he creado para las distintas preguntas de los datos y comprobar que introduzca bien el S o N, non diferencia entre mayúsculas y minúsculas. Nos devuelve true si es Si o False si en No.

public void cargarDatos()

Creamos objetos para comprobar el programa. 3 de Nino, 3 de Adulto, 3 de Senior, 3 Empleados y Atracciones.

La clase Entrada

Ha sido la encargada de pedir los datos de cada cliente que accede al parque y de la creación de estos objetos , con su incorporación al ArrayList Clientes usando el polimorfismo, ya que cada cliente se graba dependiendo de su edad: Niño(Nino), Adulto o Senior.

public void menuGenerarEntrada()

En este método preguntamos todas las casuísticas para generar la entrada, tanto si son individuales como si son familiares.

Si la entrada es familiar si compra el pase VIP todas serán VIP, si no lo compra ninguna podrá comprarlo. Pide cuantas entradas son familiares, como mínimo serán 4, y lleva un contador de ellas, para poder salir debe rellanarlas todas.

Si la edad es inferior a 3 años, no le genera entrada y tampoco la cuenta en la entrada familiar.

Se realizan descuentos dependiendo de las edades. Carnet Joven, estudiante y desempleado sólo se pregunta si es adulto.

Se muestra en pantalla la entrada y se pide confirmación para proceder a grabarla.

Al final crea los objetos con los datos introducidos y los graba en un ArrayList de Cliente, entre 3 y 12 años el objeto es Nino, entre 13 y 64 es Adulto y para más de 64 años el objeto es de tipo Senior.

public void fechaVentaAnticipada()

Se pide la fecha de entrada si se comprueba su formato, el aceptado es (dd/mm/aaaa).

Se mira si la fecha corresponde a una de las tres temporadas:

1.- temporada media.

2.- temporada alta.

3.- temporada media.

Mediante un switch se informa que sólo se podrá usar dentro de la temporada que la haya comprado.

public void imprimirEntrada()

Imprime la entrada por la pantalla y llama al método calcularPrecioEntrada para imprimir el precio de la entrada.

public boolean finDeSemana()

Comprueba si la fecha de la entrada es un fin de semana para no aplicar el descuento de entre semana. Comprobamos que el día no sea ni viernes, ni sábado ni domingo. Retornando true si es fin de semana.

public double calcularPrecioEntrada()

Calcula el precio de la entrada. Primero guarda en precioMinimo el 10% del valor de la entrada, sin descuentos y dependiendo de la edad.

Luego aplica los descuentos:

Carnet Joven 10%

Carnet Estudiante 10%

Desempleado 10%

Diversidad Funcional 20%

Temporada baja 25%

Entrada Familiar 15%

Entrada de Tarde 50%

Entre Semana 15%

Y si es temporada Alta aumenta el precio en un 25%

Si se elige suplemento VIP dependiendo de la edad se suma:

Niño 25 euros.

Adulto 50 euros

Senior 32.5 euros.

public boolean preguntasVarias(String pregunta)

Este método lo he creado para las distintas preguntas de los datos y comprobar que introduzca bien el S o N, non diferencia entre mayúsculas y minúsculas. Nos devuelve true si es Si o False si en No.

public boolean validarFecha(String fecha)

Comprueba que la fecha este bien escrita con el formato (dd/mm/aaaa).

public void limpiarPantalla()

Imprime líneas en blanco para borrar la pantalla.

public int comprobarTemporada() throws ParseException

En este método comprobamos en que temporada esta la fecha de la entrada. Usamos Calendar con sus métodos DAY_OF_MONTH y MONTH.

La clase Atracciones

Es una superclase polimórfica para crear el ArrayList de Atracciones y grabar los objetos creados cada uno de su tipo de atracción

Métodos geters y seters para los campos de la clase Atracciones.

public boolean esSenior()

public void setSenior(boolean senior)

public boolean esAdulto()

public void setAdulto(boolean adulto)

public boolean esNino()

public void setNino(boolean nino)

public boolean esVip()

public void setVip(boolean vip)

public int getAlturaMax()

public void setAlturaMax(int alturaMax)

public int getAlturaMin()

public void setAlturaMin(int alturaMin)

public int getAyudantesAtraccion()

public void setAyudantesAtraccion(int ayudantesAtraccion)

public int getResponsableAtraccion()

public void setResponsableAtraccion(int responsableAtraccion)

public int getNumeroDeAtracciones()

public void setNumeroDeAtracciones(int numeroDeAtracciones)

public String toString()

Subclases de Atracciones:

TipoA

TipoB

TipoC

TipoD

TipoE

Todas ellas heredan de Atracciones y dependen del tipo de atracción que sean.

La Clase Empleado

Ha creado los empleados con su puesto y salario que les corresponde. Han sido grabados en un ArrayList de Empleado.

Métodos geters y seters para los campos de la clase Empleado

public String getDni()

public void setDni(String dni)

public String getNombre()

public void setNombre(String nombre)

public String getPuesto()

public void setPuesto(String puesto)

public int getTelefono()

public void setTelefono(int telefono)

public String getDireccion()

public void setDireccion(String direccion)

public double getSueldoMes()

public void setSueldoMes(double sueldoMes)

La Clase Estadisticas

Ha generado las estadísticas de los clientes que han pasado por el parque por meses, el precio medio de la entrada y el coste anual y mensual de los empleados del parque.

public void menuEstadisticas()

Menú para acceder a los distinto datos.

public void numeroDeVisitantes()

Imprime por pantalla por orden de los meses del año, todos los clientes del parque.

public void precioMedioEntrada()

Suma todos los precios de las entradas del ArrayList de clientes y lo divide por un contador del ArrayList y si el contador es 0, no lo divide para no generar error. El formato de salida es con 2 decimales

public void gastoPersonal()

Calcula el total de ayudantes, responsables, atención al cliente y relaciones públicas. Lo multiplica por su sueldo y lo saca por mes. El total del mes de cada puesto lo multiplico por 12 y saco el anual.

Todo lo imprimo por pantalla.

public String toString()

Sobrescribo el método toString**.**

La Clase Cliente

Esta superclase es polimórfica para grabar los clientes de distintas subclases.

Métodos geters y seters para los campos de la clase Cliente

public String getDni()

public void setDni(String dni)

public String getNombre()

public int getEdad()

public void setEdad(int edad)

public int getAlturaCm()

public void setAlturaCm(int alturaCm)

public String getFechaEntrada()

public void setFechaEntrada(String fechaEntrada)

public boolean esDiversidadFuncional()

public void setDiversidadFuncional(boolean diversidadFuncional)

public boolean esEntradaVip()

public void setEntradaVip(boolean entradaVip)

public boolean esEntradaTarde()

public void setEntradaTarde(boolean entradaTarde)

public double esPrecioEntrada()

public void setPrecioEntrada(double precioEntrada)

public String toString()

Subclases de Cliente:

Adulto

Métodos geters y seters para los campos de la clase Adulto

public boolean esEstudiante()

public void setEstudiante(boolean estudiante)

public boolean esCarnetJoven()

public void setCarnetJoven(boolean carnetJoven)

public boolean esDesempleado()

public void setDesempleado(boolean desempleado)

public String toString()

Nino

Métodos geters y seters para los campos de la clase Adulto

public String getDniAdultoacompania()

public void setDniAdultoacompania(String dniAdultoacompania)

public String toString()

Senior

Todas ellas heredan de Clientes y dependen de la edad del cliente.

La Clase parque

Contine el método main.

Crea los ArrayList. El objeto menú y accede al menú principal.

Código fuente de las clases implementadas.

Clase parque:

```
import java.util.ArrayList;

/**
 * La Clase parque para iniciar el programa.
 *
 * @author (Juan Carlos Amor Gutierrez)
 * @version (1.0 5/5/19)
 */
public class parque {

    public ArrayList<Cliente> clientes;
    public ArrayList<Empleado> empleados;
    public ArrayList<Atracciones> atracciones;

    public static void main(String[] args) {
        ArrayList<Cliente> clientes = new ArrayList<Cliente>();
        ArrayList<Empleado> empleados = new ArrayList<Empleado>();
        ArrayList<Atracciones> atracciones = new ArrayList<Atracciones>();
        Menu menu = new Menu(clientes, empleados, atracciones);
        menu.menuPrincipal();
    }
```

}

Clase Cliente:

```
/**
 * Clientes del Parque Atracciones Superclase
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Cliente {

    private String dni;
    private String nombre;
    private int edad;
    private int alturaCm;
    private String fechaEntrada;
    private boolean diversidadFuncional;
    boolean entradaVip;
    boolean entradaTarde;
    double precioEntrada;

    /**
     * Constructor de objetos para la clase Cliente
     *
     * @param dni
     * @param nombre
```

```
     * @param edad
     * @param alturaCm
     * @param fechaEntrada
     * @param diversidadFuncional
     * @param entradaVip
     * @param entradaTarde
     * @param precioEntrada
     */

    public Cliente(String dni, String nombre, int edad, int alturaCm, String fechaEntrada,
            boolean diversidadFuncional, boolean entradaVip, boolean entradaTarde,
            double precioEntrada) {
        super();
        this.dni = dni;
        this.nombre = nombre;
        this.edad = edad;
        this.alturaCm = alturaCm;
        this.fechaEntrada = fechaEntrada;
        this.diversidadFuncional = diversidadFuncional;
        this.entradaVip = entradaVip;
        this.entradaTarde = entradaTarde;
        this.precioEntrada = precioEntrada;
    }

    public String getDni() {
```

```
    return dni;
}

public void setDni(String dni) {
    this.dni = dni;
}

public String getNombre() {
    return nombre;
}

public void setNombre(String nombre) {
    this.nombre = nombre;
}

public int getEdad() {
    return edad;
}

public void setEdad(int edad) {
    this.edad = edad;
}

public int getAlturaCm() {
    return alturaCm;
}
```

```
public void setAlturaCm(int alturaCm) {
   this.alturaCm = alturaCm;
}

public String getFechaEntrada() {
   return fechaEntrada;
}

public void setFechaEntrada(String fechaEntrada) {
   this.fechaEntrada = fechaEntrada;
}

public boolean esDiversidadFuncional() {
   return diversidadFuncional;
}

public void setDiversidadFuncional(boolean diversidadFuncional) {
   this.diversidadFuncional = diversidadFuncional;
}

public boolean esEntradaVip() {
   return entradaVip;
}

public void setEntradaVip(boolean entradaVip) {
```

```
        this.entradaVip = entradaVip;
    }

    public boolean esEntradaTarde() {
        return entradaTarde;
    }

    public void setEntradaTarde(boolean entradaTarde) {
        this.entradaTarde = entradaTarde;
    }

    public double esPrecioEntrada() {
        return precioEntrada;
    }

    public void setPrecioEntrada(double precioEntrada) {
        this.precioEntrada = precioEntrada;
    }

    @Override
    public String toString() {
        return "Cliente{" + "dni=" + dni + ", nombre=" + nombre + ", edad="
+ edad
                + ", alturaCm=" + alturaCm + ", fechaEntrada=" + fechaEntrada
                + ", diversidadFuncional=" + diversidadFuncional + ",
entradaVip="
```

```
                + entradaVip + ", entradaTarde=" + entradaTarde + ",
precioEntrada="
                + precioEntrada + '}';
    }
}
```

Clase Adulto:

```
/**
 * Adultos del parque de Atracciones
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Adulto extends Cliente {

    private boolean estudiante;
    private boolean carnetJoven;
    private boolean desempleado;

    /**
     * Constructor de objetos para la clase Adulto
     *
     * @param dni
     * @param nombre
     * @param edad
```

```
     * @param alturaCm
     * @param fechaEntrada
     * @param diversidadFuncional
     * @param entradaVip
     * @param entradaTarde
     * @param precioEntrada
     * @param estudiante
     * @param carnetJoven
     * @param desempleado
     *
     */

    public Adulto(String dni, String nombre, int edad, int alturaCm, String fechaEntrada,
            boolean diversidadFuncional, boolean entradaVip, boolean entradaTarde,
            double precioEntrada, boolean estudiante, boolean carnetJoven, boolean desempleado) {
        super(dni, nombre, edad, alturaCm, fechaEntrada, diversidadFuncional, entradaVip, entradaTarde, precioEntrada);
        this.estudiante = estudiante;
        this.carnetJoven = carnetJoven;
        this.desempleado = desempleado;
    }

    public boolean esEstudiante() {
        return estudiante;
```

```
    }

    public void setEstudiante(boolean estudiante) {
        this.estudiante = estudiante;
    }

    public boolean esCarnetJoven() {
        return carnetJoven;
    }

    public void setCarnetJoven(boolean carnetJoven) {
        this.carnetJoven = carnetJoven;
    }

    public boolean esDesempleado() {
        return desempleado;
    }

    public void setDesempleado(boolean desempleado) {
        this.desempleado = desempleado;
    }

    @Override
    public String toString() {
        return "Adulto{" + "estudiante=" + estudiante + ", carnetJoven=" +
carnetJoven + ", desempleado=" + desempleado + '}';
    }
```

}

Clase Nino:

```
/**
 * Niños del parque de Atracciones
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Nino extends Cliente {

        private String dniAdultoacompania;

      /**
    * Constructor de objetos para la clase Nino
    *
    * @param dni
    * @param nombre
    * @param edad
    * @param alturaCm
    * @param fechaEntrada
    * @param diversidadFuncional
    * @param entradaVip
    * @param entradaTarde
```

```
     * @param precioEntrada
     * @param dniAdultoacompania
     */

        public Nino(String dni, String nombre, int edad, int alturaCm,
String fechaEntrada,
            boolean diversidadFuncional, boolean entradaVip, boolean
entradaTarde,
                double precioEntrada, String dniAdultoacompania) {
                super(dni, nombre, edad, alturaCm, fechaEntrada,
diversidadFuncional, entradaVip, entradaTarde, precioEntrada);
                this.dniAdultoacompania = dniAdultoacompania;
        }

        public String getDniAdultoacompania() {
                return dniAdultoacompania;
        }

        public void setDniAdultoacompania(String dniAdultoacompania) {
                this.dniAdultoacompania = dniAdultoacompania;
        }

    @Override
    public String toString() {
       return "Nino{" + "dniAdultoacompania=" + dniAdultoacompania +
'}';
    }
```

```
}
```

Clase Senior:

```
/**
 * Senior del parque Atracciones
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Senior extends Cliente {

    /**
     * Constructor de objetos para la clase Senior
     *
     * @param dni
     * @param nombre
     * @param edad
     * @param alturaCm
     * @param fechaEntrada
     * @param diversidadFuncional
     * @param entradaVip
     * @param entradaTarde
     * @param precioEntrada
```

```
     *
     */

    public Senior(String dni, String nombre, int edad, int alturaCm, String fechaEntrada,
            boolean diversidadFuncional, boolean entradaVip, boolean entradaTarde,
            double precioEntrada) {
        super(dni, nombre, edad, alturaCm, fechaEntrada, diversidadFuncional,
                entradaVip, entradaTarde, precioEntrada);
    }

}
```

Clase Entrada:

```
/**
 * Write a description of class Entrada here.
 *
 * @author (Juan Carlos Amor Gutierrez)
 * @version (1.0 05/05/2019)
 */
import java.util.InputMismatchException;
import java.util.Scanner;
import java.text.ParseException;
import java.text.SimpleDateFormat;
```

```
import java.util.ArrayList;
import java.util.Calendar;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Entrada {

    private String dni;
    private String nombre;
    private String dniacompaniante;
    private int edad;
    private int alturaCm;
    private double precioEntrada;
    private boolean diversidadFuncional; // minusvalía
    private boolean carnetJoven;
    private boolean carnetEstudiante;
    private boolean desempleado;
    private boolean entradaTarde;
    private boolean entradaVip;
    private String fechaEntrada;
    private ArrayList<Cliente> clientes;
    private boolean ventaAnticipada;
    private boolean entradaFamilia;

    /**
```

```
     * Constructor de objetos para la clase Entrada
     *
     * @param ventaAnticipada
     * @param entradaFamilia
     * @param clientes
     *
     */
    public Entrada(boolean ventaAnticipada, boolean entradaFamilia,
ArrayList<Cliente> clientes) {
        this.dni = "";
        this.nombre = "";
        this.dniacompaniante = "";
        this.edad = 0;
        this.alturaCm = 0;
        this.precioEntrada = 0;
        this.diversidadFuncional = false;
        this.carnetJoven = false;
        this.carnetEstudiante = false;
        this.desempleado = false;
        this.entradaTarde = false;
        this.entradaVip = false;
        this.fechaEntrada = "";
        this.clientes = clientes;
        this.ventaAnticipada = ventaAnticipada;
        this.entradaFamilia = entradaFamilia;
    }
```

```
        // Menú principal para crear las entradas
        public void menuGenerarEntrada() {

            Scanner sn = new Scanner(System.in);
            boolean salir = false;
            boolean todasEntradasVip = false;
            boolean entradaAceptada = false;

            int numeroEntradaFamilia = 0;
            int contadorEntradaFamilia = 0;
            //cargarFechasTempora();
            limpiarPantalla();
            System.out.println("\n\n\n\n");
            System.out.println("Entrada Parque de Atraccciones");
            System.out.println("==============================");
            System.out.println();
            if (entradaFamilia) {
                while (numeroEntradaFamilia < 4) {
                    System.out.print("Número de entradas familiares (min 4
personas): ");
                    numeroEntradaFamilia = sn.nextInt();
                }
            }
            // Peticiones de datos para toda la casuística respecto a las entradas
            while (!salir) {
                try {
                    if (entradaFamilia) {
```

```
                    contadorEntradaFamilia++;
                    System.out.println("Entrada Familiar numero: " +
contadorEntradaFamilia);
                    System.out.println("==========================");
                }
                System.out.println();
                System.out.println();
                System.out.print("Edad: ");
                edad = sn.nextInt();
                System.out.println();
                if (edad > 2) {
                    dni = sn.nextLine();
                    System.out.print("DNI del Cliente: ");
                    dni = sn.nextLine();
                    System.out.println();
                    System.out.print("Nombre del Cliente: ");
                    nombre = sn.nextLine();
                    System.out.println();
                    if (edad < 13) {
                        System.out.print("DNI del Acompañante: ");
                        dniacompaniante = sn.nextLine();
                        System.out.println();
                    }
                    System.out.print("Altura en centimetros: ");
                    alturaCm = sn.nextInt();
                    System.out.println();
                    diversidadFuncional = false;
```

```
                diversidadFuncional = preguntasVarias("Diversidad
Funcional");
                entradaTarde = false;
                entradaTarde = preguntasVarias("Entrada por la Tarde");
                entradaVip = false;
                if (!entradaFamilia) {
                   entradaVip = preguntasVarias("Entrada VIP");
                }
                if (todasEntradasVip && entradaFamilia) {
                   entradaVip = true;
                }

                if (contadorEntradaFamilia == 1 && entradaFamilia) {
                   entradaVip = preguntasVarias("Entrada VIP");
                   todasEntradasVip = entradaVip;
                }

                if (entradaVip && entradaFamilia) {
                   System.out.println("Todas las Entradas Familiares son
VIP");
                   System.out.println();
                } else if (entradaFamilia) {
                   System.out.println("todas las entradas familiares deben ser
iguales en el suplemento VIP");
                   System.out.println();
                }
```

```
                    if (edad > 12 && edad < 64) {
                        carnetJoven = false;
                        carnetJoven = preguntasVarias("Carnet Joven");
                        carnetEstudiante = false;
                        carnetEstudiante = preguntasVarias("Carnet Estudiante");
                        desempleado = false;
                        desempleado = preguntasVarias("Desempleado");
                    }

                    if (ventaAnticipada) {
                        fechaVentaAnticipada();
                    } else {
                        Date fecha = new Date();
                        String fechaFormato = "dd/MM/yyyy";
                        SimpleDateFormat formatoFecha = new
SimpleDateFormat(fechaFormato);
                        fechaEntrada = (formatoFecha.format(fecha));

                    }

                } else {
                    System.out.println("Niños menores de 3 años no necesitan
entrada");
                    System.out.println();
                    System.out.println();
                }
```

```
                imprimirEntrada();


                entradaAceptada = preguntasVarias("Es correcta la entrada");
                if (entradaAceptada) {
                   if (edad > 2 && edad < 13) {
                      Nino entradaNino = new Nino(dni, nombre, edad,
alturaCm, fechaEntrada,
                            diversidadFuncional, entradaVip, entradaTarde,
precioEntrada, dniacompaniante);
                      clientes.add(entradaNino);
                   }
                   if (edad > 12 && edad < 64) {
                      Adulto entradaAdulto = new Adulto(dni, nombre, edad,
alturaCm, fechaEntrada,
                            diversidadFuncional, entradaVip, entradaTarde,
precioEntrada,
                            carnetEstudiante, carnetJoven, desempleado);
                      clientes.add(entradaAdulto);
                   }
                   if (edad > 64) {
                      Senior entradaSenior = new Senior(dni, nombre, edad,
alturaCm, fechaEntrada,
                            diversidadFuncional, entradaVip, entradaTarde,
precioEntrada);
                      clientes.add(entradaSenior);
                   }
                }
                if (!entradaFamilia) {
```

```
            salir = true;
        }
        if (edad < 3 && entradaFamilia) {
            contadorEntradaFamilia--;
        }
        if (numeroEntradaFamilia == contadorEntradaFamilia) {
            salir = true;
        }

    } catch (InputMismatchException e) {
        System.out.println("Solo numeros");
        sn.next();
    }
  }

}
// Método para introducir una fecha del año
public void fechaVentaAnticipada() {
  do {
    Scanner leerFecha = new Scanner(System.in);
    System.out.println("Fecha Entrada (dd/mm/aaaa)");
    fechaEntrada = leerFecha.next();
  } while (!validarFecha(fechaEntrada));
  System.out.println();
  System.out.println();
  int temporada = 0;
```

```
            try {
                temporada = comprobarTemporada();
            } catch (ParseException ex) {
            }
            // Información de cuando se puede usar la entrada según la
temporada
            switch (temporada) {
                case 1:
                    System.out.println("Puedes usarla solo en temporada media");
                    break;
                case 2:
                    System.out.println("Puedes usarla solo en temporada alta");
                    break;
                case 3:
                    System.out.println("Puedes usarla solo en temportada baja");
                    break;
            }

            System.out.println("\n\n");
        }

            // Imprimir en pantalla la entrada con el precio final
        public void imprimirEntrada() {
            limpiarPantalla();
            if (edad > 2) {
                System.out.println("Entrada Parque Atracciones");
                System.out.println("==========================");
```

```
            System.out.println("Dni: " + dni + "  Nombre: " + nombre);
            this.precioEntrada = calcularPrecioEntrada();
            System.out.println("El precio de la entrada: " + precioEntrada);
            System.out.println("\n\n");
        }

    }

        // Comprobar si es fin de semana
    public boolean finDeSemana() {
        int numeroDia = 0;
        String fechaFormato = "dd/MM/yyyy";
        SimpleDateFormat formatoFecha = new
SimpleDateFormat(fechaFormato);
        Calendar cal = Calendar.getInstance();
        try {
            cal.setTime(formatoFecha.parse(fechaEntrada));
        } catch (ParseException ex) {
        }
        numeroDia = cal.get(Calendar.DAY_OF_WEEK);
        if (numeroDia == 1 || numeroDia == 6 || numeroDia == 7) {
            return true;
        }
        return false;
    }
```

```
        // Calcular el precio de la entrada mínimo y los descuentos
aplicables
        public double calcularPrecioEntrada() {
            double precioAcumulado = 0;
            double precioMinimo = 0;
            double precioDeLaEntrada = 0;
            int temporada = 0;

            if (edad > 2 && edad < 13) {
                System.out.println("Entrada Niño");
                System.out.println("============");
                precioDeLaEntrada = 30;
                precioAcumulado = 30;
                precioMinimo = (precioDeLaEntrada * 0.10);
            }
            if (edad > 64) {
                System.out.println("Entrada Senior");
                System.out.println("==============");
                precioDeLaEntrada = 39;
                precioAcumulado = 39;
                precioMinimo = (precioDeLaEntrada * 0.10);
            }

            if (edad > 12 && edad < 64) {
                System.out.println("Entrada Adulto");
                System.out.println("==============");
                precioDeLaEntrada = 60;
```

```
                precioAcumulado = 60;
                precioMinimo = (precioDeLaEntrada * 0.10);
            }
            if (carnetJoven) {
                System.out.println("Descuento del 10% por Carnet Joven");
                precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.10);
            }
            if (carnetEstudiante) {
                System.out.println("Descuento del 10% por Carnet Estudiante");
                precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.10);
            }
            if (desempleado) {
                System.out.println("Descuento del 10% por Desempleado");
                precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.10);
            }
            if (diversidadFuncional) {
                System.out.println("Descuento del 20% por Discapacidad
Funcional");
                precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.20);
            }
            try {
                temporada = comprobarTemporada();
            } catch (ParseException ex) {
            }
```

```
            if (temporada == 2) {
               System.out.println("Aumento del 25% por Temporada Alta");
               precioAcumulado = precioAcumulado + (precioDeLaEntrada *
0.25);
            }

            if (temporada == 3) {
               System.out.println("Descuento del 25% por Temporada Baja");
               precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.25);
            }
            if (entradaFamilia) {
               System.out.println("Descuento del 15% por Entrada Familiar");
               precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.15);
            }
            if (entradaTarde) {
               System.out.println("Descuento del 50% por Entrada de Tarde");
               precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.50);
            }

            if (!finDeSemana()) {
               System.out.println("Descuento del 15% por Entrada entre
semana");
               precioAcumulado = precioAcumulado - (precioDeLaEntrada *
0.15);
            }
```

```
if (precioAcumulado < precioMinimo) {
   precioDeLaEntrada = precioMinimo;
} else {
   precioDeLaEntrada = precioAcumulado;
}
   // Suma de la entrada VIP
if (edad > 2 && edad < 13) {
   if (entradaVip) {
      System.out.println("Entrada VIP + 25 euros");
      precioDeLaEntrada = precioDeLaEntrada + 25;
   }
}
if (edad > 64) {
   if (entradaVip) {
      System.out.println("Entrada VIP + 32.5 euros");
      precioDeLaEntrada = precioDeLaEntrada + 32.5;
   }
}

if (edad > 12 && edad < 65) {
   if (entradaVip) {
      System.out.println("Entrada VIP + 50 euros");
      precioDeLaEntrada = precioDeLaEntrada + 50;
   }
}
```

```
    return precioDeLaEntrada;
}

    // Método para realizar preguntas para los descuentos.
    // Devuelve verdadero o falso a la pregunta.
public boolean preguntasVarias(String pregunta) {
    Scanner sn = new Scanner(System.in);
    boolean salir = false;
    String respuesta = "";
    boolean respuestaDevuelta = false;

    try {
        do {
            System.out.print(pregunta + " (S/N) :");
            respuesta = sn.nextLine();
            if (respuesta.equalsIgnoreCase("s")) {
                respuestaDevuelta = true;
                salir = true;
            }
            if (respuesta.equalsIgnoreCase("n")) {
                respuestaDevuelta = false;
                salir = true;
            }
            System.out.println();
        } while (!salir);
    } catch (InputMismatchException e) {
```

```
            }
            return respuestaDevuelta;
        }

            // Método para comprobar que la fecha esté bien introducida
        public boolean validarFecha(String fecha) {
            try {
                SimpleDateFormat formatoFecha = new
SimpleDateFormat("dd/MM/yyyy");
                formatoFecha.setLenient(false);
                formatoFecha.parse(fecha);
            } catch (ParseException e) {
                return false;
            }
            return true;
        }

        public void limpiarPantalla() {
            System.out.println("\n\n\n\n\n\n");
        }
            // Se comprueban las temporadas según la fecha de la entrada
        public int comprobarTemporada() throws ParseException {
            int temporada = 1; // 1=temporada media, 2=temporada alta,
3=temporada baja
            int numeroMes = 0;
            int numeroDia = 0;
```

```
        SimpleDateFormat formatoFecha = new
SimpleDateFormat("dd/MM/yyyy");
        Calendar cal = Calendar.getInstance();
        cal.setTime(formatoFecha.parse(fechaEntrada));
        try {
           cal.setTime(formatoFecha.parse(fechaEntrada));
        } catch (ParseException ex) {
        }
        numeroDia = cal.get(Calendar.DAY_OF_MONTH);
        numeroMes = cal.get(Calendar.MONTH);
        if (numeroMes == 7 || numeroMes == 8 || numeroMes == 11) {
           temporada = 2; // Temporada alta
        }
        if (numeroMes == 0) {
           if (numeroDia > 0 && numeroDia < 9) { // Navidad
              temporada = 2; // temporada alta
           }
        }
        if (numeroMes == 3) {
           if (numeroDia > 14 && numeroDia < 22) { // Semana Santa
              temporada = 2; // temporada alta
           }
        }
        if (numeroMes == 1 || numeroMes == 10) {
           temporada = 3; // temporada baja
        }
        return temporada;
```

```
    }

}
```

Clase Empleado:

```
/**
 * Empleados
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Empleado {

    // Campos de los empleados
    private String dni;
    private String nombre;
    private String puesto;
    private int telefono;
    private String direccion;
    private double sueldoMes;
    /**
     * Constructor para objetos de la clase Empleado
     *
```

```
     * @param dni
     * @param nombre
     * @param direccion
     * @param telefono
     * @param puesto
     * @param sueldoMes
     *
     */

    public Empleado(String dni, String nombre, String direccion, int
telefono,
            String puesto, double sueldoMes) {
        this.dni = dni;
        this.nombre = nombre;
        this.puesto = puesto;
        this.telefono = telefono;
        this.direccion = direccion;
        this.sueldoMes = sueldoMes;
    }

    public String getDni() {
        return dni;
    }

    public void setDni(String dni) {
        this.dni = dni;
    }
```

```
public String getNombre() {
   return nombre;
}

public void setNombre(String nombre) {
   this.nombre = nombre;
}

public String getPuesto() {
   return puesto;
}

public void setPuesto(String puesto) {
   this.puesto = puesto;
}

public int getTelefono() {
   return telefono;
}

public void setTelefono(int telefono) {
   this.telefono = telefono;
}

public String getDireccion() {
```

```
        return direccion;
    }

    public void setDireccion(String direccion) {
        this.direccion = direccion;
    }

    public double getSueldoMes() {
        return sueldoMes;
    }

    public void setSueldoMes(double sueldoMes) {
        this.sueldoMes = sueldoMes;
    }
}
```

Clase Estadistica:

```
import java.text.ParseException;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Calendar;
import java.util.InputMismatchException;
import java.util.Scanner;
```

```
/**
 * Para las estadísticas del parque Clientes, Empleados y Atracciones
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Estadisticas {

    // Declaramos los ArrayList
    public ArrayList<Cliente> clientes;
    public ArrayList<Empleado> empleados;
    public ArrayList<Atracciones> atracciones;

    /**
     * Constructor de objetos para la clase Estadísticas
     *
     * @param clientes
     * @param empleados
     * @param atracciones
     */
    public Estadisticas(ArrayList clientes, ArrayList empleados, ArrayList
atracciones) {
        this.clientes = clientes;
        this.empleados = empleados;
        this.atracciones = atracciones;
    }
```

```
        public void menuEstadisticas() {
            Scanner sn = new Scanner(System.in);
            boolean salir = false;
            int opcion; //Guardaremos la opción del menú
            System.out.println("\n\n\n\n\n\n");
            while (!salir) {
                System.out.println("    MENU ESTADISTICAS ");
                System.out.println("    ==================");
                System.out.println("     [1]. Numero de Visitantes");
                System.out.println("     [2]. Precio Medio de la Entrada");
                System.out.println("     [3]. Visitas medias registradas");
                System.out.println("     [4]. Resumen anual del gasto de
personal");
                System.out.println("     [5]. Salir");

                try {

                    System.out.println("\n\n\n");
                    System.out.println("    Escribe una de las opciones del menu");
                    System.out.print("    > ");
                    opcion = sn.nextInt();

                    switch (opcion) {
                        case 1:
                            numeroDeVisitantes();
                            break;
                        case 2:
```

```
                precioMedioEntrada();
                break;
            case 3:
                visitasMedias();
            case 4:
                gastoPersonal();
                break;
            case 5:
                salir = true;
                break;

            default:
                System.out.println("\n\n\n\n\n\n");
                System.out.println("Solo números entre 1 y 5");
        }
      } catch (InputMismatchException e) {
          System.out.println("\n\n\n\n\n\n");
        System.out.println("Solo números entre 1 y 5");
        sn.next();
      }
   }

}
// El número de visitantes por meses
public void numeroDeVisitantes() {
```

```
            String fechaFormato = "dd/MM/yyyy";
            SimpleDateFormat formatoFecha = new
SimpleDateFormat(fechaFormato);
            Calendar cal = Calendar.getInstance(); // Instanciamos el objeto
Calendar
            for (int i = 0; i < 12; i++) {
               for (Cliente cliente : clientes) {
                  try {
                     // Lo convertimos a formato calendar con parse
                     cal.setTime(formatoFecha.parse(cliente.getFechaEntrada()));
                  } catch (ParseException ex) {
                  }
                  // Nos da el mes empezando de 0 para enero,...
                  if ((cal.get(Calendar.MONTH)) == i) {
                     switch (i + 1) {
                        case 1:
                           System.out.println("Mes de Enero: ");
                           System.out.println("Fecha: " + cliente.getFechaEntrada()
                                 + " Nombre: " + cliente.getNombre());
                           System.out.println();
                           break;
                        case 2:
                           System.out.println("Mes de Febrero: ");
                           System.out.println("Fecha: " + cliente.getFechaEntrada()
                                 + " Nombre: " + cliente.getNombre());
                           System.out.println();
                           break;
```

```
case 3:
   System.out.println("Mes de Marzo: ");
   System.out.println("Fecha: " + cliente.getFechaEntrada()
         + " Nombre: " + cliente.getNombre());
   System.out.println();
   break;
case 4:
   System.out.println("Mes de Abril: ");
   System.out.println("Fecha: " + cliente.getFechaEntrada()
         + " Nombre: " + cliente.getNombre());
   System.out.println();
   break;
case 5:
   System.out.println("Mes de Mayo: ");
   System.out.println("Fecha: " + cliente.getFechaEntrada()
         + " Nombre: " + cliente.getNombre());
   System.out.println();
   break;
case 6:
   System.out.println("Mes de Junio: ");
   System.out.println("Fecha: " + cliente.getFechaEntrada()
         + " Nombre: " + cliente.getNombre());
   System.out.println();
   break;
case 7:
   System.out.println("Mes de Julio: ");
```

```
            System.out.println("Fecha: " + cliente.getFechaEntrada()
                    + " Nombre: " + cliente.getNombre());
            System.out.println();
            break;
        case 8:
            System.out.println("Mes de Agosto: ");
            System.out.println("Fecha: " + cliente.getFechaEntrada()
                    + " Nombre: " + cliente.getNombre());
            System.out.println();
            break;
        case 9:
            System.out.println("Mes de Septiembre: ");
            System.out.println("Fecha: " + cliente.getFechaEntrada()
                    + " Nombre: " + cliente.getNombre());
            System.out.println();
            break;
        case 10:
            System.out.println("Mes de Octubre: ");
            System.out.println("Fecha: " + cliente.getFechaEntrada()
                    + " Nombre: " + cliente.getNombre());
            System.out.println();
            break;
        case 11:
            System.out.println("Mes de Noviembre: ");
            System.out.println("Fecha: " + cliente.getFechaEntrada()
                    + " Nombre: " + cliente.getNombre());
```

```
                    System.out.println();
                    break;
                case 12:
                    System.out.println("Mes de Diciembre: ");
                    System.out.println("Fecha: " + cliente.getFechaEntrada()
                            + " Nombre: " + cliente.getNombre());
                    System.out.println();
                    break;
                default:

                }

            }

        }
    }
}

    // Calcular el precio medio de todos los clientes del parque
    public void precioMedioEntrada() {
        int i = 0;
        double totalPrecioEntrada = 0;
        for (Cliente cliente : clientes) {
            totalPrecioEntrada = totalPrecioEntrada +
cliente.esPrecioEntrada();
            i++;
        }
```

```
            if (i != 0) {
                System.out.println("\n\n\n");
                // Para tener solo 2 decimales con un String.format
                System.out.println("El precio medio de la entrada es: "
                        + String.format("%.2f", (totalPrecioEntrada / i)));
                System.out.println("\n\n\n");
            }
        }

        // Calcular las visitas medias por dias, meses, etc..
        public void visitasMedias() {

        }

        // Calcular el gasto personal por meses y por años
        public void gastoPersonal() {
            int sumaAyudantes = 0;
            int sumaResponsables = 0;
            int atencionCliente = 0;
            int relacionesPublicas = 0;
            double gastoPersonal = 0;

            for (Atracciones atraccion : atracciones) {
                sumaAyudantes = sumaAyudantes
                        + (atraccion.getAyudantesAtraccion() *
atraccion.getNumeroDeAtracciones());
```

```
                sumaResponsables = sumaResponsables +
atraccion.getNumeroDeAtracciones();
            }
            atencionCliente = (int) ((sumaAyudantes + sumaResponsables) *
0.3);
            relacionesPublicas = (int) ((sumaAyudantes + sumaResponsables) *
0.1);
            gastoPersonal = ((sumaAyudantes * 950) + (sumaResponsables *
1092.5)
                    + (atencionCliente * 1045) + (relacionesPublicas * 1140));
            System.out.println("\n\n");
            System.out.println("Gasto Mensual");
            System.out.println("Ayudantes de Atracciones: " + sumaAyudantes
                    + "  Total Ayudantes: " + (sumaAyudantes * 950) + " euros");
            System.out.println("Responsables de Atracciones: " +
sumaResponsables
                    + "  Total Responsables: " + (sumaResponsables * 1092.5 + "
euros"));
            System.out.println("Atencion al Cliente: " + atencionCliente
                    + "  Total Atencion al Cliente: " + (atencionCliente * 1045 + "
euros"));
            System.out.println("Relaciones Publicas: " + relacionesPublicas
                    + "  Total Relaciones Publicas: " + (relacionesPublicas * 1140 + "
euros"));
            System.out.println("El Gasto Mensual de Personal: " + gastoPersonal
+ " euros");
            System.out.println("\n\n");
            System.out.println("Gasto Anual");
```

```
            System.out.println("Ayudantes de Atracciones: " + ((sumaAyudantes
* 950) * 12) + " euros");

            System.out.println("Responsables de Atracciones: " +
((sumaResponsables * 1092.5) * 12) + " euros");

            System.out.println("Atencion al Cliente: " + ((atencionCliente *
1045) * 12) + " euros");

            System.out.println("Relaciones Publicas: " + ((relacionesPublicas *
1140) * 12) + " euros");

            System.out.println("El Gasto Anual de Personal: " + (gastoPersonal *
12) + " euros");

            System.out.println("\n\n");

        }


        @Override

        public String toString() {

            return "Estadisticas{" + "clientes=" + clientes + ", empleados=" +
empleados

                    + ", atracciones=" + atracciones + '}';

        }

    }
```

Clase Menu:

```
    import java.util.ArrayList;

    import java.util.InputMismatchException;

    import java.util.Scanner;
```

```
import java.util.Iterator;

/**
 * Clase Menu para los menús
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Menu {

    private boolean ventaAnticipada; // Controla la venta por internet
    private boolean entradaFamilia; //Controla las entradas familiares

    public ArrayList<Cliente> clientes;
    public ArrayList<Empleado> empleados;
    public ArrayList<Atracciones> atracciones;

     /**
     * Constructor de objetos para la clase Menu
     *
     * @param clientes
     * @param empleados
     * @param atracciones
     * @param ventaAnticipada
     * @param entradaFamilia
     *
```

```
     */

    public Menu(ArrayList<Cliente> clientes, ArrayList<Empleado>
empleados,
            ArrayList<Atracciones> atracciones) {
        this.clientes = clientes;
        this.empleados = empleados;
        this.atracciones = atracciones;
        this.ventaAnticipada = false;
        this.entradaFamilia = false;
    }

    // Menu Principal del Programa
    public void menuPrincipal() {
        cargarDatos(); // Cargamos los datos en los arraysList
        Scanner sn = new Scanner(System.in);
        boolean salir = false;
        int opcion; //Guardaremos la opción del usuario
        limpiarPantalla();
        while (!salir) {
            System.out.println("    MENU PARQUE DE ATRACCIONES");
            System.out.println("    ==========================");
            System.out.println("     [1]. Menu Entradas");
            System.out.println("     [2]. Menu Empleados");
            System.out.println("     [3]. Menu Datos Estadísticos");
            System.out.println("     [4]. Menu Atracciones");
            System.out.println("     [5]. Salir");
```

```
                try {

                    System.out.println("\n\n\n");
                    System.out.println("    Escribe una de las opciones del menú");
                    System.out.print("    > ");
                    opcion = sn.nextInt();

                    switch (opcion) {
                        case 1:
                            menuEntradasPrincipal();
                            break;
                        case 2:
                            menuEmpleados();
                            break;
                        case 3:
                            // Le pasamos los ArrayList para mostrar las estadísticas
                            Estadisticas estadistica = new Estadisticas(clientes,
empleados, atracciones);
                            estadistica.menuEstadisticas();
                            break;
                        case 4:
                            menuAtracciones();
                            break;
                        case 5:
                            salir = true;
                            break;
```

```
            default:
                limpiarPantalla();
                System.out.println("Solo números entre 1 y 5");
            }
        } catch (InputMismatchException e) {
            limpiarPantalla();
            System.out.println("Solo números entre 1 y 5");
            sn.next();
        }
    }
}

public void menuEntradasPrincipal() {
    Scanner sn = new Scanner(System.in);
    boolean salir = false;
    ventaAnticipada = false;
    int opcion; //Guardaremos la opción del menú
    limpiarPantalla();
    while (!salir) {
        System.out.println("    MENU ENTRADAS ");
        System.out.println("    ==============");
        System.out.println("     [1]. Entradas para el Día");
        System.out.println("     [2]. Entradas por Internet");
        System.out.println("     [3]. Regresar al Menú Principal");

        try {
```

```
            System.out.println("\n\n\n");
            System.out.println("    Escribe una de las opciones del menú");
            System.out.print("    > ");
            opcion = sn.nextInt();

            switch (opcion) {
                case 1:
                    ventaAnticipada = false;
                    menuEmitirEntradas();
                    break;
                case 2:
                    ventaAnticipada = true;
                    menuEmitirEntradas();
                    break;
                case 3:
                    salir = true;
                    break;
                default:
                    limpiarPantalla();
                    System.out.println("Solo números entre 1 y 3");
            }
        } catch (InputMismatchException e) {
            limpiarPantalla();
            System.out.println("Solo números del 1 al 3");
            sn.next();
```

```
        }
    }

}

//Menú para las entradas tanto del día como de anticipadas
public void menuEmitirEntradas() {
    Scanner sn = new Scanner(System.in);
    boolean salir = false;
    entradaFamilia = false;
    int opcion; //Guardaremos la opcion del menú
    limpiarPantalla();
    while (!salir) {
        System.out.println("    MENU EMITIR ENTRADAS ");
        System.out.println("    ====================");
        System.out.println("     [1]. Entrada Individual");
        System.out.println("     [2]. Entradas família");
        System.out.println("     [3]. Regresar al Menú anterior");

        try {

            System.out.println("\n\n\n");
            System.out.println("    Escribe una de las opciones del menú");
            System.out.print("    > ");
            opcion = sn.nextInt();
```

```
                switch (opcion) {
                    case 1:
                        limpiarPantalla();
                        // le pasamos si es venta anticipada, si está dentro de una familiar y el arrayList de clientes
                        Entrada entrada = new Entrada(ventaAnticipada, entradaFamilia, clientes);
                        entradaFamilia = false;
                        entrada.menuGenerarEntrada();
                        break;
                    case 2:
                        limpiarPantalla();
                        entradaFamilia = true;  // ponemos verdadera la entrada família,
                                                //creamos el objetos y generamos la entrada
                        Entrada entradafamiliar = new Entrada(ventaAnticipada, entradaFamilia, clientes);
                        entradafamiliar.menuGenerarEntrada();
                        break;
                    case 3:
                        salir = true;
                        break;
                    default:
                        limpiarPantalla();
                        System.out.println("Solo números entre 1 y 3");
                }
            } catch (InputMismatchException e) {
```

```
            limpiarPantalla();
            System.out.println("\n\n");
            System.out.println("Solo números del 1 al 3");
            sn.next();
        }
    }

}

public void limpiarPantalla() {
    System.out.println("\n\n\n\n\n\n\n\n\n\n\n");
}

public void menuEmpleados() {
    Scanner sn = new Scanner(System.in);
    boolean salir = false;
    int opcion;
    while (!salir) {
        System.out.println("\n\n\n\n\n\n");
        System.out.println("    MENU EMPLEADOS ");
        System.out.println("    ===============");
        System.out.println("     [1]. Añadir Empleado");
        System.out.println("     [2]. Borrar Empleado");
        System.out.println("     [3]. Listar Empleado");
        System.out.println("     [4]. Salir");
```

```
try {

    System.out.println("\n\n\n");
    System.out.println("    Escribe una de las opciones del menú");
    System.out.print("    > ");
    opcion = sn.nextInt();

    switch (opcion) {
        case 1:
            agregarEmpleado();
            break;
        case 2:
            borrarEmpleado();
            break;
        case 3:
            listarEmpleados();
            break;
        case 4:
            salir = true;
            break;
        default:
            System.out.println("\n\n");
            System.out.println("Solo números entre 1 y 4");
            System.out.println("\n\n");
    }
} catch (InputMismatchException e) {
```

```
            System.out.println("\n\n");
            System.out.println("Solo números entre 1 y 4");
            System.out.println("\n\n");
            sn.next();
        }
    }
}

public void agregarEmpleado() {

    Scanner sn = new Scanner(System.in);
    int opcion;
    String dni = "";
    String nombre = "";
    String puesto = "";
    int telefono = 0;
    String direccion = "";
    double sueldoMes = 0;
    double sueldoBase = 950;

    System.out.println();
    System.out.println();
    System.out.println("Introducir datos Empleado");
    System.out.println("========================");
    System.out.println();
```

```
System.out.print("Dni: ");
dni = sn.nextLine();
System.out.println();

System.out.print("Nombre: ");
nombre = sn.nextLine();
System.out.println();

System.out.print("Dirección: ");
direccion = sn.nextLine();
System.out.println();
boolean salir = false;
while (!salir) {
   try {
      System.out.print("Teléfono: ");
      telefono = sn.nextInt();
      System.out.println();
      salir = true;
   } catch (InputMismatchException e) {

      System.out.println("\n\n");
      System.out.println("Solo números para el teléfono");
      System.out.println("\n\n");
      sn.next();
   }
```

```
}
opcion = 0;
salir = false;
while (!salir) {
   try {
      System.out.println("      Puesto de trabajo");
      System.out.println("      ================");
      System.out.println("     [1]. Atención al Cliente");
      System.out.println("     [2]. Relaciones Públicas");
      System.out.println("     [3]. Responsables de Atracción");
      System.out.println("     [4]. Ayudante de Atracción");
      System.out.println();
      System.out.println("    Escribe uno de los puestos");
      System.out.print("    > ");
      opcion = sn.nextInt();

      switch (opcion) {
         case 1:
            puesto = "Atención al cliente";
            sueldoMes = sueldoBase * 1.10;
            salir = true;
            break;
         case 2:
            puesto = "Relaciones Públicas";
            sueldoMes = sueldoBase * 1.20;
            salir = true;
```

```
                    break;
                case 3:
                    puesto = "Responsables de Atracción";
                    sueldoMes = sueldoBase * 1.15;
                    salir = true;
                    break;
                case 4:
                    puesto = "Ayudante de Atracción";
                    sueldoMes = sueldoBase;
                    salir = true;
                    break;
                default:
                    System.out.println("\n\n");
                    System.out.println("Solo números entre 1 y 4");
                    System.out.println("\n\n");
            }
        } catch (InputMismatchException e) {
            System.out.println("Solo números");
            sn.next();

        }
    }
    Empleado emple = new Empleado(dni, nombre, direccion, telefono,
puesto, sueldoMes);
    System.out.println(emple.getDni() + " " + emple.getNombre() + " " +
emple.getDireccion()
```

```
                    + " " + emple.getTelefono() + " " + emple.getPuesto() + " " +
emple.getSueldoMes());
                if (empleados.add(emple)) {
                    System.out.println();
                    System.out.println("El empleado " + emple.getNombre() + " ha
sido grabado correctamente");
                } else {
                    System.out.println();
                    System.out.println("El empleado " + emple.getNombre() + " NO
ha sido grabado");
                }
            }

            public void borrarEmpleado() {
                Scanner sn = new Scanner(System.in);
                String dniBorrar = "";
                boolean confirmarBorrar = false;
                System.out.println();
                System.out.print("Introduce DNI del empleado a eliminar: ");
                dniBorrar = sn.nextLine();
                Iterator<Empleado> it = empleados.iterator();
                while (it.hasNext()) {
                    Empleado emplea = it.next();
                    if (dniBorrar.equalsIgnoreCase(emplea.getDni())) {
                        System.out.println();
                        System.out.println("Dni: " + emplea.getDni() + " Nombre: " +
emplea.getNombre());
```

```
                    confirmarBorrar = preguntasVarias("Deseas borrar este
empleado");
                    if (confirmarBorrar) {
                        System.out.println();
                        System.out.println("El empleado " + emplea.getNombre() + "
ha sido borrado");
                        empleados.remove(emplea);
                    } else {
                        System.out.println();
                        System.out.println("El empleado " + emplea.getNombre() + "
NO ha sido borrado");
                    }
                } else {
                    System.out.println("DNI no encontrado");
                }
            }
        }

        public void listarEmpleados() {
            boolean salir = false;
            for (Empleado emplea : empleados) {
                System.out.println();
                System.out.println("Dni: " + emplea.getDni());
                System.out.println("Nombre: " + emplea.getNombre());
                System.out.println("Dirección: " + emplea.getDireccion());
                System.out.println("Teléfono: " + emplea.getTelefono());
                System.out.println("Puesto: " + emplea.getPuesto());
```

```
        System.out.println("Sueldo Mes: " + emplea.getSueldoMes());
        System.out.println();

    }
}

// Menú para gestionar atracciones
public void menuAtracciones() {
    Scanner sn = new Scanner(System.in);
    boolean salir = false;
    int opcion;
    while (!salir) {
        System.out.println("\n\n\n\n\n\n");
        System.out.println("    MENU ATRACCIONES");
        System.out.println("    ================");
        System.out.println("     [1]. Añadir Atracciones");
        System.out.println("     [2]. Borrar Atracciones");
        System.out.println("     [3]. Listar Atracciones");
        System.out.println("     [4]. Salir");

        try {

            System.out.println("\n\n\n");
            System.out.println("    Escribe una de las opciones del menú");
            System.out.print("    > ");
            opcion = sn.nextInt();
```

```
    switch (opcion) {
        case 1:
            agregarAtracciones();
            break;
        case 2:
            borrarAtracciones();
            break;
        case 3:
            listarAtracciones();
            break;
        case 4:
            salir = true;
            break;
        default:
            System.out.println("\n\n");
            System.out.println("Solo números entre 1 y 4");
            System.out.println("\n\n");
    }
} catch (InputMismatchException e) {

    System.out.println("\n\n");
    System.out.println("Solo números entre 1 y 4");
    System.out.println("\n\n");
    sn.next();
}
```

```
    }
}

public void agregarAtracciones() {

    Scanner sn = new Scanner(System.in);
    boolean salir = false;
    int numeroTipoA = 0;
    int numeroTipoB = 0;
    int numeroTipoC = 0;
    int numeroTipoD = 0;
    int numeroTipoE = 0;
    boolean correcto = false;

    try {
        System.out.println("\n\n\n\n\n\n");
        System.out.println("    MENU AGREGAR ATRACCIONES");
        System.out.println("    ========================");
        System.out.println("\n\n\n\n");
        System.out.print("Atracciones Tipo A ¿Cuantas?(4) >");
        numeroTipoA = sn.nextInt();
        System.out.println();
        System.out.print("Atracciones Tipo B ¿Cuantas?(6) >");
        numeroTipoB = sn.nextInt();
        System.out.println();
        System.out.print("Atracciones Tipo C ¿Cuantas?(4) >");
```

```
    numeroTipoC = sn.nextInt();
    System.out.println();
    System.out.print("Atracciones Tipo D ¿Cuantas?(3) >");
    numeroTipoD = sn.nextInt();
    System.out.println();
    System.out.print("Atracciones Tipo E ¿Cuantas?(7) >");
    numeroTipoE = sn.nextInt();
  } catch (InputMismatchException e) {
    limpiarPantalla();
    System.out.println("\n\n");
    System.out.println("Solo números");
    sn.next();
  }
  System.out.println();
  System.out.println("    RESUMEN");
  System.out.println("    =======\n");
  System.out.println("Atracciones Tipo A: " + numeroTipoA);
  System.out.println("Atracciones Tipo B: " + numeroTipoB);
  System.out.println("Atracciones Tipo C: " + numeroTipoC);
  System.out.println("Atracciones Tipo D: " + numeroTipoD);
  System.out.println("Atracciones Tipo E: " + numeroTipoE);
  correcto = preguntasVarias("Son correctas las atracciones");
  if (correcto = true) {
    TipoA atraccionA = new TipoA(numeroTipoA);
    atracciones.add(atraccionA);
    TipoB atraccionB = new TipoB(numeroTipoB);
```

```
        atracciones.add(atraccionB);
        TipoC atraccionC = new TipoC(numeroTipoC);
        atracciones.add(atraccionC);
        TipoD atraccionD = new TipoD(numeroTipoD);
        atracciones.add(atraccionD);
        TipoE atraccionE = new TipoE(numeroTipoE);
        atracciones.add(atraccionE);

    }
}

public void listarAtracciones() {
    int sumaAyudantes = 0;
    int atencionCliente = 0;
    int relacionesPublicas = 0;
    int sumaResponsables = 0;
    double gastoPersonal = 0;

    for (Atracciones atraccion : atracciones) {
        System.out.println("\n\n");
        System.out.println("Atracciones " + atraccion.toString()
                + "  Número de atracciones funcionando "
                + atraccion.getNumeroDeAtracciones());
        if (atraccion.esAdulto() || atraccion.esSenior()) {
            System.out.println("Permitido Adultos");
        }
```

```
                if (atraccion.esNino()) {
                    System.out.println("Permitido Niños");
                }
                if (atraccion.esVip()) {
                    System.out.println("Se permite el pase VIP");
                }
                System.out.println("Número de Ayudantes de Atraccion: "
                        + atraccion.getAyudantesAtraccion());
                sumaAyudantes = sumaAyudantes + 1 +
atraccion.getAyudantesAtraccion();
            }
            // Para decir cuantos empleados se necesitan y el coste de de ellos,
tanto mensual como anual
            if (sumaAyudantes != 0) {
                for (Atracciones atraccion : atracciones) {
                    sumaAyudantes = sumaAyudantes +
atraccion.getAyudantesAtraccion();
                    sumaResponsables++;
                }
                atencionCliente = (int) ((sumaAyudantes + sumaResponsables) *
0.3);
                relacionesPublicas = (int) ((sumaAyudantes + sumaResponsables)
* 0.1);
                System.out.println("Se necesitan " + atencionCliente
                        + " empleados de Atención al Cliente");
                System.out.println("Se necesitan " + relacionesPublicas
                        + " empleados de Relaciones Públicas");
```

```
                gastoPersonal = ((sumaAyudantes * 950) + (sumaResponsables *
1092.5)
                        + (atencionCliente * 1045) + (relacionesPublicas * 1140));
                System.out.println("El Gasto Mensual de Personal: " +
gastoPersonal + " euros");
                System.out.println("El Gasto Anual de Personal: " + (gastoPersonal
* 12) + " euros");
             }


          }


          public void borrarAtracciones() {
             System.out.println("\n\n\n\n\n");
             if (preguntasVarias("Deseas borrar todas las Atracciones")) {
                atracciones.clear();
             }
          }
          // Este método lo he creado para verificar si responde "s/n" y no
diferencie entre mayúsculas y minísculas
          // me devuelve verdadero o falso a la pregunta
          public boolean preguntasVarias(String pregunta) {
             Scanner sn = new Scanner(System.in);
             boolean salir = false;
             String respuesta = "";
             boolean respuestaDevuelta = false;


             try {
```

```
                do {
                    System.out.print(pregunta + " (S/N) :");
                    respuesta = sn.nextLine();
                    if (respuesta.equalsIgnoreCase("s")) {
                        respuestaDevuelta = true;
                        salir = true;
                    }
                    if (respuesta.equalsIgnoreCase("n")) {
                        respuestaDevuelta = false;
                        salir = true;
                    }
                    System.out.println();
                } while (!salir);
            } catch (InputMismatchException e) {

            }
            return respuestaDevuelta;
        }

            // Objetos para cargar los datos a los ArrayList, para pruebas de
estadísticas
        public void cargarDatos() {
            Nino entradaNino1 = new Nino("25369784S", "José Martinez", 8, 95,
"25/02/2019",
                    false, true, false, 50, "36999999Z");
            clientes.add(entradaNino1);
```

```
        Adulto entradaAdulto1 = new Adulto("58988785F", "Pedro Prieto",
35, 185, "11/03/2019",
                true, true, false, 100,
                false, false, false);
        clientes.add(entradaAdulto1);
        Senior entradaSenior1 = new Senior("5846987A", "Rogélio
Gonzalez", 76, 175, "06/08/2019",
                true, false, false, 26.3);
        clientes.add(entradaSenior1);
        Nino entradaNino2 = new Nino("2587456H", "Francisco Trigo", 3,
75, "25/02/2019",
                false, true, false, 50, "3555559Z");
        clientes.add(entradaNino2);
        Adulto entradaAdulto2 = new Adulto("5256475F", "María Gracia",
55, 175, "11/09/2019",
                true, false, true, 85.5,
                false, true, false);
        clientes.add(entradaAdulto2);
        Senior entradaSenior2 = new Senior("5846987A", "Amáncio Silva",
66, 165, "06/01/2019",
                true, true, false, 56.3);
        clientes.add(entradaSenior2);
        Nino entradaNino3 = new Nino("2566997K", "Eva López", 10, 115,
"25/12/2019",
                true, true, false, 45, "369745879Z");
        clientes.add(entradaNino3);
        Adulto entradaAdulto3 = new Adulto("58988785F", "María Prieto",
15, 135, "11/06/2019",
```

```
            true, true, false, 80,
            false, false, false);
        clientes.add(entradaAdulto3);
        Senior entradaSenior3 = new Senior("5846987A", "Gerardo Rivas",
76, 175, "06/11/2019",
            true, false, true, 56.3);
        clientes.add(entradaSenior3);


        Empleado emple1 = new Empleado("66958448T", "María Lopez",
"c/ Nueva, 14", 68792554, "Relaciones Publicas", 1092.5);
        empleados.add(emple1);
        Empleado emple2 = new Empleado("66898544G", "José Martinez",
"c/ Vieja, 89", 68799634, "Ayudante de Atraccion", 950);
        empleados.add(emple2);
        Empleado emple3 = new Empleado("98745684R", "Nuria Blazquez",
"c/ Granvia, 1", 68743554, "Atencion al cliente", 1045);
        empleados.add(emple3);


        TipoA atraccionA = new TipoA(4);
        atracciones.add(atraccionA);
        TipoB atraccionB = new TipoB(6);
        atracciones.add(atraccionB);
        TipoC atraccionC = new TipoC(4);
        atracciones.add(atraccionC);
        TipoD atraccionD = new TipoD(3);
        atracciones.add(atraccionD);
        TipoE atraccionE = new TipoE(7);
```

```
        atracciones.add(atraccionE);
    }
}
```

Clase Atracciones:

```
/*
 * Para generar las Atracciones del parque
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class Atracciones {

    // Declaramos los campos de las atracciones
    private boolean senior;
    private boolean adulto;
    private boolean nino;
    private boolean vip;
    private int alturaMax;
    private int alturaMin;
    private int ayudantesAtraccion;
    private int responsableAtraccion;
    private int numeroDeAtracciones;

    /**
     * Constructor de objetos para la clase Atracciones
```

```
 *
 * @param senior
 * @param adulto
 * @param nino
 * @param vip
 * @param alturaMax
 * @param alturaMin
 * @param ayudantesAtraccion
 * @param responsableAtraccion
 * @param numeroDeAtracciones
 *
 */
public Atracciones() {
    this.senior = false;
    this.adulto = false;
    this.nino = false;
    this.vip = false;
    this.alturaMax = 0;
    this.alturaMin = 0;
    this.ayudantesAtraccion = 0;
    this.responsableAtraccion = 0;
    this.numeroDeAtracciones = 0;
}

    // Métodos Geters y Seters
public boolean esSenior() {
```

```
    return senior;
}

public void setSenior(boolean senior) {
    this.senior = senior;
}

public boolean esAdulto() {
    return adulto;
}

public void setAdulto(boolean adulto) {
    this.adulto = adulto;
}

public boolean esNino() {
    return nino;
}

public void setNino(boolean nino) {
    this.nino = nino;
}

public boolean esVip() {
    return vip;
}
```

```
public void setVip(boolean vip) {
   this.vip = vip;
}

public int getAlturaMax() {
   return alturaMax;
}

public void setAlturaMax(int alturaMax) {
   this.alturaMax = alturaMax;
}

public int getAlturaMin() {
   return alturaMin;
}

public void setAlturaMin(int alturaMin) {
   this.alturaMin = alturaMin;
}

public int getAyudantesAtraccion() {
   return ayudantesAtraccion;
}

public void setAyudantesAtraccion(int ayudantesAtraccion) {
```

```
        this.ayudantesAtraccion = ayudantesAtraccion;
    }

    public int getResponsableAtraccion() {
        return responsableAtraccion;
    }

    public void setResponsableAtraccion(int responsableAtraccion) {
        this.responsableAtraccion = responsableAtraccion;
    }

    public int getNumeroDeAtracciones() {
        return numeroDeAtracciones;
    }

    public void setNumeroDeAtracciones(int numeroDeAtracciones) {
        this.numeroDeAtracciones = numeroDeAtracciones;
    }

    @Override
    public String toString() {
        return "Atracciones{" + "senior=" + senior + ", adulto=" + adulto + ",
nino="
                + nino + ", vip=" + vip + ", alturaMax=" + alturaMax + ",
alturaMin="
                + alturaMin + ", ayudantesAtraccion=" + ayudantesAtraccion +
```

```
                ", responsableAtraccion=" + responsableAtraccion + ",
numeroDeAtracciones="
                + numeroDeAtracciones + '}';
    }

}
```

Clase TipoA:

```
/**
 * Atracción Tipo A.
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class TipoA extends Atracciones {

    /**
     * Constructor de objetos para la clase Tipo A
     *
     * @param senior
     * @param adulto
     * @param nino
     * @param vip
     * @param alturaMax
```

```
     * @param alturaMin
     * @param ayudantesAtraccion
     * @param responsableAtraccion
     * @param numeroDeAtracciones
     *
     */

    public TipoA(int numeroAtracciones) {
        super();
        this.setSenior(true);
        this.setAdulto(true);
        this.setNino(true);
        this.setVip(true);
        this.setAlturaMax(250);
        this.setAlturaMin(120);
        this.setAyudantesAtraccion(6);
        this.setResponsableAtraccion(1);
        this.setNumeroDeAtracciones(numeroAtracciones);
    }

    @Override
    public String toString() {
        return "TipoA{" + '}';
    }

}
```

Clase TipoB:

```
/**
 * Atracción Tipo B.
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class TipoB extends Atracciones {

    /**
    * Constructor de objetos para la clase Tipo B
    *
    * @param senior
    * @param adulto
    * @param nino
    * @param vip
    * @param alturaMax
    * @param alturaMin
    * @param ayudantesAtraccion
    * @param responsableAtraccion
    * @param numeroDeAtracciones
    *
    */
```

```
    public TipoB(int numeroAtracciones) {
        super();
        this.setSenior(true);
        this.setAdulto(true);
        this.setNino(false);
        this.setVip(false);
        this.setAlturaMax(190);
        this.setAlturaMin(120);
        this.setAyudantesAtraccion(5);
        this.setResponsableAtraccion(1);
        this.setNumeroDeAtracciones(numeroAtracciones);
    }

    @Override
    public String toString() {
        return "TipoB{" + '}';
    }

}
```

Clase TipoC:

```
/**
 * Atracción Tipo C.
```

```
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class TipoC extends Atracciones {

    /**
     * Constructor de objetos para la clase Tipo C
     *
     * @param senior
     * @param adulto
     * @param nino
     * @param vip
     * @param alturaMax
     * @param alturaMin
     * @param ayudantesAtraccion
     * @param responsableAtraccion
     * @param numeroDeAtracciones
     *
     */

    public TipoC(int numeroAtracciones) {
        super();
        this.setSenior(false);
        this.setAdulto(false);
        this.setNino(true);
```

```
        this.setVip(false);
        this.setAlturaMax(120);
        this.setAlturaMin(0);
        this.setAyudantesAtraccion(3);
        this.setResponsableAtraccion(1);
        this.setNumeroDeAtracciones(numeroAtracciones);
    }

    @Override
    public String toString() {
        return "TipoC{" + '}';
    }

}
```

Clase TipoD:

```
/**
 * Atracción Tipo D.
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class TipoD extends Atracciones {

    /**
```

```
 * Constructor de objetos para la clase Tipo D
 *
 * @param senior
 * @param adulto
 * @param nino
 * @param vip
 * @param alturaMax
 * @param alturaMin
 * @param ayudantesAtraccion
 * @param responsableAtraccion
 * @param numeroDeAtracciones
 *
 */

public TipoD(int numeroAtracciones) {
   super();
   this.setSenior(true);
   this.setAdulto(true);
   this.setNino(true);
   this.setVip(true);
   this.setAlturaMax(250);
   this.setAlturaMin(0);
   this.setAyudantesAtraccion(5);
   this.setResponsableAtraccion(1);
   this.setNumeroDeAtracciones(numeroAtracciones);
}
```

```
    @Override
    public String toString() {
        return "TipoD{" + '}';
    }

}
```

Clase TipoE:

```
/**
 * Atracción Tipo E.
 *
 * @author Juan Carlos Amor Gutiérrez
 * @version (1.0 05/05/2019)
 */
public class TipoE extends Atracciones {

    /**
     * Constructor de objetos para la clase Tipo E
     *
     * @param senior
     * @param adulto
     * @param nino
     * @param vip
```

```
     * @param alturaMax
     * @param alturaMin
     * @param ayudantesAtraccion
     * @param responsableAtraccion
     * @param numeroDeAtracciones
     *
     */

    public TipoE(int numeroAtracciones) {
        super();
        this.setSenior(true);
        this.setAdulto(true);
        this.setNino(false);
        this.setVip(true);
        this.setAlturaMax(250);
        this.setAlturaMin(0);
        this.setAyudantesAtraccion(7);
        this.setResponsableAtraccion(1);
        this.setNumeroDeAtracciones(numeroAtracciones);
    }

    @Override
    public String toString() {
        return "TipoE{" + '}';
    }
```

```
}
```