Programación Orientada a Objetos

Curso 2018-2019

APELLIDOS Y NOMBRE: **Periscal Poteiro , Juan**

IDENTIFICADOR: **jperiscal1**

DNI: **53306672D**

CENTRO ASOCIADO MATRICULADO: **047000 – A CORUÑA**

CENTRO ASOCIADO DE LA SESIÓN DE CONTROL: **047000 – A CORUÑA**

EMAIL DE CONTACTO: **jperiscal1@alumno.uned.es**

TELÉFONO DE CONTACTO: **625773978**

1 . ESTRUCTURA

Del enunciado se extrae la necesidad de cuatro entidades (clases, tipos, categorías) generales, estas son: trabajador, turista, atracción y entrada. En base a esto, cada clase mencionada junto con sus subclases se incluirán dentro de su correspondiente paquete; por lo que teniendo además un paquete para la clase principal del programa y demás clases no relacionadas con las anteriores la estructura de paquetes se conformará como:

- principal:
  - Clase Parque (con el método *main()*).
  - Clase GUI, se corresponde con lo que será la interfaz gráfica de usuario
  - Clase AtraccionesFuncionanando
  - Persona, de esta clase heredarán trabajador y turista
  - Estadisticas
- trabajadores:
  - Clase Trabajador, de la que heredarán las clases que representarán los diferentes tipos de trabajadores del parque de atracciones
  - Clase TAtencionCliente, empleado que cuya función es la atención al cliente
  - Clase TAyudanteAtr, representa el empleado ayudante en una atracción
  - Clase TRelacionesPublicas, empleado encargado de las relaciones públicas
  - Clases TResponsableAtr, empleado encargado de cada atracción
- turistas:
  - Clase Turista, de la que heredarán clases que representan a los diferentes tipos de turistas concebidos en el parque
  - Clase Nino, representa a aquellos visitantes con no más de 12 años
  - Clase Adulto, representa a aquellos visitantes entre más de 12 y menos de 65 años
  - Clase Senior, representa a aquellos visitantes mayores de 65 años
- atracciones:
  - Clase Atraccion, de la que heredarán clases que representan a los diferentes tipos de atracciones
  - Clase AtraccA: `Adulto` y `Nino`, **AlturaMín**:1,20 , **AlturaMáx**:- ,**VIP**: si, **NúmAyudantes**:6
  - Clase AtraccB: `Adulto`, **AlturaMín**:1,20, **AlturaMáx**:1,90 ,**VIP**:no , **NúmAyudantes**:5
  - Clase AtraccC: `Nino`, **AlturaMín**:-, **AlturaMáx**:1,20,**VIP**:no , **NúmAyudantes**:3
  - Clase AtraccD: `Adulto` y `Nino`, **AlturaMín**:-, **AlturaMáx**:- ,**VIP**:si , **NúmAyudantes**:5
  - Clase AtraccE. `Adulto`, **AlturaMín**:-, **AlturaMáx**:-,**VIP**:si, **NúmAyudantes**:7
- entradas
  - Entrada, de la que heredarán clases que representan a los diferentes tipos de entradas
  - EDiaLaborable
  - EFamiliar

- EGeneral
- ETarde
- EOtra

Las clases Persona, Turista, Trabajador, Entrada y Atraccion son abstractas, impidiendo así que puedan ser instanciadas. Si se desea crear un objeto de alguna de estas clases debe hacer se a través de alguna de sus subclases, ya que estas reúnen características concretas que tienen lugar en el ámbito del parque que deben ser recogidas obligatoriamente.

Dada la necesidad de recoger la información de los diferentes elementos presentes en el parque de atracciones, para estadísticas, gestión del gasto, etc; se hace necesario recoger los objetos creados en listas o tablas Hash para posteriormente acceder a ellos. Para ello, las clases Turista, Trabajador, Entrada y Atraccion disponen de un atributo estático (que pertenece a la clase, no a la instancia), para ir recogiendo las propias instancias que son subclase de estas.

Turista	Clase abstracta que representa a un tipo de turista (cliente) en el parque de atracciones. Las características que se tienen en cuenta para definir todo turista son su nombre y apellidos, y su fecha de nacimiento. Para dar cuenta de todas los turistas, se dispone como atributos de clase un HashMap que almacena todos los turistas que han visitado el parque, además de un contador del total de los mismos. Dado que es necesario conocer el nombre, apellidos y nacimiento de todo visitante; toda clase que herede de Turista debe pasar como parámetros al constructor como mínimo el `nombre`, `apellidos` y fecha de `nacimiento`. Ya que estos serán atributos de esta clase. Además a cada tipo de turista se le asociara un identificador (`idTurista`), y un porcentaje de descuento correspondiente al tipo de turista que sea (`descTipoTurista`) De esta clase heredan las clases: `Nino`, `Adulto` y `Senior`
Entrada	La clase abstracta `Entrada` representa una entrada emitida en el parque de atracciones para cada nuevo turista. Las diferentes características que definen una entrada son: la fecha y hora a la que es emitida, el tipo de turista que la recibe, si incluye bonificación VIP y, por supuesto, su precio. Para dar cuenta de todas las entradas, se dispone como atributos de clase una lista que almacena todas las entradas emitidas, además de un contador del total de las mismas. Las clases que heredan de esta son: `EDiaLaborable`, `EFamiliar`, `EGeneral`, `ETarde` y `EOtra`
Trabajador	Clase abstracta que representa a un trabajador en el parque de atracciones. Las características que se tienen en cuenta para definir todo trabajador. son su nombre y apellidos, y su sueldo. Para dar cuenta de todos los trabajadores, se dispone como atributos de clase un HashMap que almacena todos los trabajadores del parque, además de un contador del total de los mismos. Dado que es necesario conocer el nombre y apellidos; toda clase que herede de Trabajador debe pasar como parámetros al constructor como mínimo el `nombre` y `apellidos`. Ya que estos serán atributos de esta clase. Además a cada tipo de trabajador se le asociara un identificador (`idTrabajador`), y el sueldo correspondiente al puesto vaya a ocupar (`sueldo`). De esta clase heredan las clases: `TAtencionCliente`, `TAyundateAtr`, `TRelacionesPublicas`, `TResponsableAtr`

Atraccion	Clase abstracta que representa una atracción del parque de atracciones. En una atracción se deben definir diversas características como si acepta adultos y/o niños. Cuál debe ser la altura máxima y/o mínima permitida para el acceso a la misma. y, finalmente, si está activa o no. Y, si permite el uso de las ventajas de la bonificación VIP. También, respecto al personal del parque, un objeto Atraccion contiene una lista de los ayudantes de atracción, así como también el trabajador responsable correspondiente. Para dar cuenta de todas las atracciones, se dispone como atributos de clase un HashMap que almacena todas las atracciones existentes, además de un contador del total de las mismas. Las clases que heredan de esta son: AtraccA, AtraccB, AtraccC, AtraccD y AtraccE

❖ Solución a diversos requisitos del enunciado

Respecto a la creación de los diferentes objetos de turistas, la elección de la clase no se deja en manos del usuario, sino que el método de clase Turista (estático) *altaTurista* se encargar de crear el objeto de la clase adecuada en función de la fecha de nacimiento del cliente.

```
public static Turista altaTurista (String nombre, String apellidos, LocalDate nacimiento, LocalDate ahora) {
      Turista t =null;
      int edad = nacimiento.until(ahora).getYears();

      if(edad<=12) t = new Nino(nombre,apellidos,nacimiento);
      else if(edad > 12 && edad<65) t = new Adulto(nombre,apellidos,nacimiento);
      else if (edad >=65) t = new Senior(nombre,apellidos,nacimiento);

      return t;
   }
```

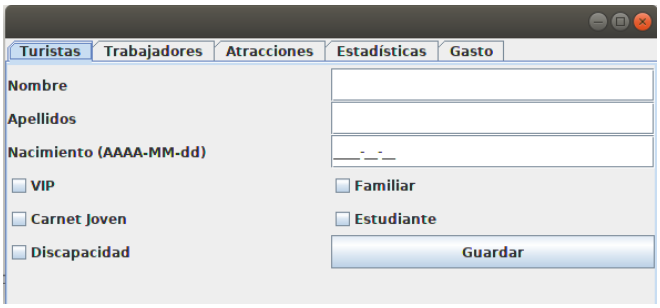
Similar al caso anterior, el tipo de entrada lo decide el método de clase *nuevaEntrada* en la clase abstracta Entrada (a excepción de si se trata de una opción familiar). Entrada de tarde para entradas emitidas después de las 16:00, entrada de día laborable para los lunes, martes, miércoles y jueves; y en otro caso entrada general. Además el método aplica un incremento o decremento del 15% en función de la temporada alta o baja.

```
public static Entrada nuevaEntrada(Turista turista, LocalDateTime ahora, boolean vip, boolean familiar) {
      Entrada e = null;
      // CreaciÃ³n de las entradas segÃºn las caracterÃ-sticas..
      if(familiar) e = new EFamiliar(turista,ahora,vip);

      else if (LocalTime.from(ahora).isAfter(LocalTime.of(16, 0))) e = new ETarde(turista,ahora,vip);

      else if (ahora.getDayOfWeek().equals(DayOfWeek.MONDAY)     ||
            ahora.getDayOfWeek().equals(DayOfWeek.TUESDAY)    ||
            ahora.getDayOfWeek().equals(DayOfWeek.WEDNESDAY)||
            ahora.getDayOfWeek().equals(DayOfWeek.THURSDAY)) e = new EDiaLaborable(turista,ahora,vip);

      else e = new EGeneral(turista,ahora,vip);

      // Aumento o descuento en funciÃ³n de la temporada en las que se emite la entrada
      LocalDate fecha = ahora.toLocalDate();
      float precio = e.precioEntrada();
      if( fecha.isAfter(LocalDate.of(2018, 12, 31)) && fecha.isBefore(LocalDate.of(2019,1,9))
       || fecha.isAfter(LocalDate.of(2019, 4, 14)) && fecha.isBefore(LocalDate.of(2019,4,22))
       || fecha.isAfter(LocalDate.of(2019, 7, 31)) && fecha.isBefore(LocalDate.of(2019,9,1))
       || fecha.isAfter(LocalDate.of(2019, 11, 30)) && fecha.isBefore(LocalDate.of(2020,1,1))) {
         precio+= PRECIO_BASE*(1+15/100);

      }else if(fecha.isAfter(LocalDate.of(2019, 1, 31)) && fecha.isBefore(LocalDate.of(2019,3,1))
         || fecha.isAfter(LocalDate.of(2019, 10, 31)) && fecha.isBefore(LocalDate.of(2019,12,1))) {
         e.decrementar(15);
      }

      //GeneraciÃ³n ticket
      e.ticket ="////////////////////////\n"
            + "Entrada Parque\n\n"
            + "Nombre: "+turista.getNombre()+" "+turista.getApellidos()+"\n"
            + "Fecha: "+ ahora.getDayOfMonth()+"-"+ahora.getMonthValue()+"-"+ahora.getYear()+"\n"
            + "Hora: " + ahora.getHour()+":"+ahora.getMinute()+"\n\n"
            + "Tipo turista: "+turista.getClass().getSimpleName()+"\n"
            + "Tipo entrada: "+ e.getClass().getSimpleName()+"\n"
            + "Precio: ";
```

```
        return e;
    }
```

Respecto de los trabajadores, en el constructor de cada subclase de Trabajador se define el sueldo que corresponde según la clase.

2 . Funcionamiento Programa

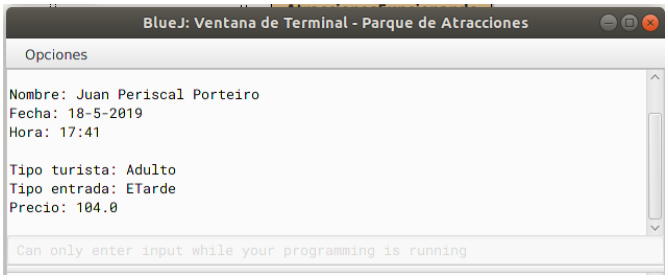


Al arrancar el programa se muestra una interfaz gráfica dee usuario, con cinco pestañas: Turistas, Trabajadores, Atracciones, Estadísticas y Gasto.

La primera de las pestañas es la correspondiente al alta de turistas. Nos permite introducir los campo necesarios para construir un objeto que hereda de clase `Turista`, como son el nombre, apellidos y nacimiento.

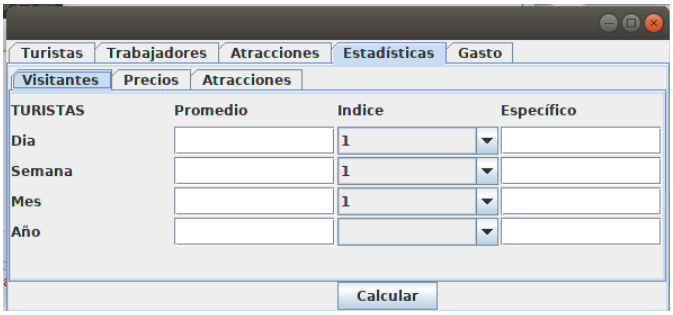
Además, existe la opción de marcar distintas opciones para obtener bonificaciones en la entrada, como son la bonificaciónVIP, carnet joven, discapacidad, familiar, estudiante. El marcado destas distintas casillas afectará al descuento de la entrada en el momento de la generación del objeto que hereda de la clase `Entrada`.

Una vez introducidos los datos, pulsar el botón “Guardar”, da lugar a la creación de un objeto de una subaclase de `Turista` (si es `Nino`, `Adulto` o `Senior` dependerá de la fecha de nacimiento)



El constructor de una subclase de Entrada, llamará a un método *ticket()* que,como se puede ver en la ilustración de la izquierda, imprimirá por consola el ticket de la entrada para el nuevo turista. En este ticket se muestra el nombre del visitante, la fecha y hora en la que se emite dicho ticket, el tipo de entrada correspondiente a la edad del turista, el tipo de entrada correspondiente a las carácterísticas del contexto de su emisión,y finalmente el precio de la entrada con sus descuentos ya aplicados.

La pestaña Trabajador, tiene como fin dar de alta un nuevo empleado, para lo cual solamente es necesario introducir su nombre y apellidos.



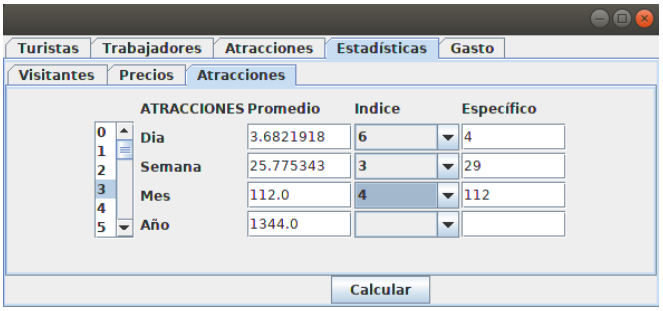
La siguiente pestaña, Estadísticas, tiene como cometido mostrar infomarción sobre los turistas, los precios de la entradas, y atracciones según el día, semana, mes y año; además de los promedios diarios, semanales, mensuales y anuales.

Para solicitar el cálculo de los datos es necesario pulsar el botón “Calcular”, esto es así, poder pedir nuevos datos tras realizar nuevas altas de turistas, atracciones, nuevas entradas, etc.

Dentro de Estadísticas, la subpestaña Visitantes nos permite informarnos del numero de visitantes promedio diario, semanal, mensual y anual. También tenemos la opoción de seleccionar el número de dia, semana y mes (el año, no, siempre serña 2019) en la lista despegable, son es necesario elegir y atomáticamente aparecerá el valor correspondiente a la derecha,en “Específico”.

Toda la información mostrada en este ejemplo del funcionamiento de la aplicación, ha sido previamente **generada aleatoriamente** al iniar el programa. Si se vuelve a correr el programa sse observará que los valores serán distintos.

La estructura de la subpestaña “Precios” de la pestaña “Estadísticas” es análoga a la de “Visitantes”. Mostrándonos los diferentes valores recogidos para los precios de las entradas en diferentes períodos.



La subpestaña Atracciones muestra información sobre las visitas relazadas a las diferentes atracciones del parque según el dia, semana y mes; además de sus promedios.

Si se desean ver los valores una atracción en concreto es necesario seleccionar su número identificar en el panel de la izquierda.

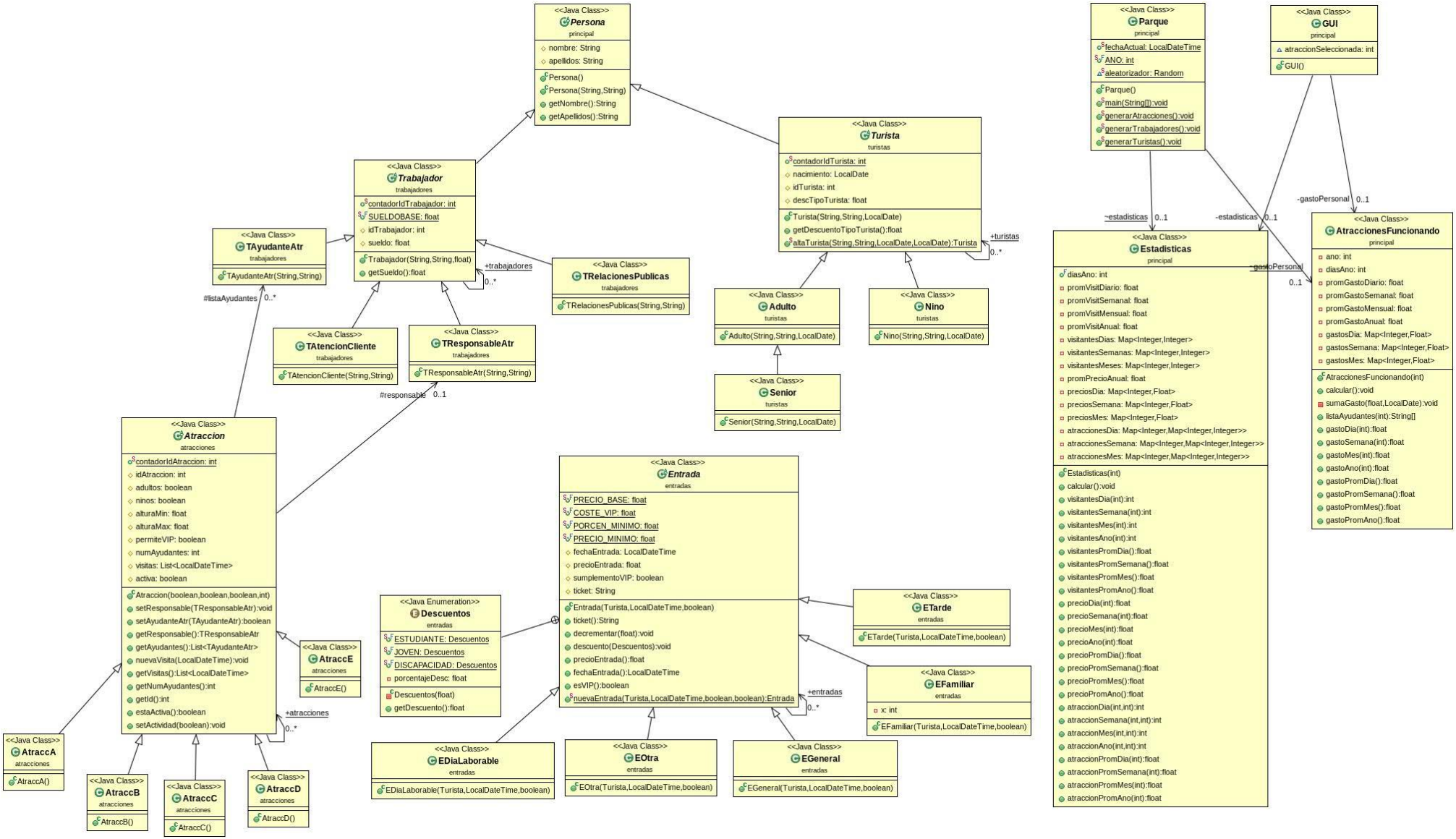
Finalmente la pestaña de “Gasto” nos muestra infomación relativa a los gastos en personal.

Para generar los datos es necesario pulsara el botón “Calcular”.

Seleccionando el número identificador de la atracción en el panel de la izquierda, se mostrará el nombre del responsable (objeto TResponsableAtr) y debajo el nombre de los ayudantes (objeto TAyudanteAtr).

En la parte derecha, tenemos una estructura similar a las subpstañas de estadísticas que nos ofrece información del gasto asociado a la atracción seleccionada según e día, semana, mes y año.

3 . DIAGRAMAS


<<Java Class>>
Persona
principal
nombre: String
apellidos: String
Persona()
Persona(String,String)
getNombre():String
getApellidos():String
<<Java Class>>
Parque
principal
fechaActual: LocalDateTime
ANO: int
aleatorizador: Random
Parque()
main(String[]):void
generarAtracciones():void
generarTrabajadores():void
generarTuristas():void
<<Java Class>>
GUI
principal
atraccionSeleccionada: int
GUI()
<<Java Class>>
Trabajador
trabajadores
contadorIdTrabajador: int
SUELDOBASE: float
idTrabajador: int
sueldo: float
Trabajador(String,String,float)
getSueldo():float
<<Java Class>>
Turista
turistas
contadorIdTurista: int
nacimiento: LocalDate
idTurista: int
descTipoTurista: float
Turista(String,String,LocalDate)
getDescuentoTipoTurista():float
altaTurista(String,String,LocalDate,LocalDate):Turista
<<Java Class>>
TAyudanteAtr
trabajadores
TAyudanteAtr(String,String)
<<Java Class>>
TRelacionesPublicas
trabajadores
TRelacionesPublicas(String,String)
<<Java Class>>
Adulto
turistas
Adulto(String,String,LocalDate)
<<Java Class>>
Nino
turistas
Nino(String,String,LocalDate)
<<Java Class>>
Senior
turistas
Senior(String,String,LocalDate)
<<Java Class>>
TAtencionCliente
trabajadores
TAtencionCliente(String,String)
<<Java Class>>
TResponsableAtr
trabajadores
TResponsableAtr(String,String)
<<Java Class>>
Estadisticas
principal
diasAno: int
promVisitDiario: float
promVisitSemanal: float
promVisitMensual: float
promVisitAnual: float
visitantesDias: Map<Integer,Integer>
visitantesSemanas: Map<Integer,Integer>
visitantesMeses: Map<Integer,Integer>
promPrecioAnual: float
preciosDia: Map<Integer,Float>
preciosSemana: Map<Integer,Float>
preciosMes: Map<Integer,Float>
atraccionesDia: Map<Integer,Map<Integer,Integer>>
atraccionesSemana: Map<Integer,Map<Integer,Integer>>
atraccionesMes: Map<Integer,Map<Integer,Integer>>
Estadisticas(int)
calcular():void
visitantesDia(int):int
visitantesSemana(int):int
visitantesMes(int):int
visitantesAno(int):int
visitantesPromDia():float
visitantesPromSemana():float
visitantesPromMes():float
visitantesPromAno():float
precioDia(int):float
precioSemana(int):float
precioMes(int):float
precioAno(int):float
precioPromDia():float
precioPromSemana():float
precioPromMes():float
precioPromAno():float
atraccionDia(int,int):int
atraccionSemana(int,int):int
atraccionMes(int,int):int
atraccionAno(int,int):int
atraccionPromDia(int):float
atraccionPromSemana(int):float
atraccionPromMes(int):float
atraccionPromAno(int):float
<<Java Class>>
AtraccionesFuncionando
principal
ano: int
diasAno: int
promGastoDiario: float
promGastoSemanal: float
promGastoMensual: float
promGastoAnual: float
gastosDia: Map<Integer,Float>
gastosSemana: Map<Integer,Float>
gastosMes: Map<Integer,Float>
AtraccionesFuncionando(int)
calcular():void
sumaGasto(float,LocalDate):void
listaAyudantes(int):String[]
gastoDia(int):float
gastoSemana(int):float
gastoMes(int):float
gastoAno(int):float
gastoPromDia():float
gastoPromSemana():float
gastoPromMes():float
gastoPromAno():float
<<Java Class>>
Atraccion
atracciones
contadorIdAtraccion: int
idAtraccion: int
adultos: boolean
ninos: boolean
alturaMin: float
alturaMax: float
permiteVIP: boolean
numAyudantes: int
visitas: List<LocalDateTime>
activa: boolean
Atraccion(boolean,boolean,boolean,int)
setResponsable(TResponsableAtr):void
setAyudanteAtr(TAyudanteAtr):boolean
getResponsable():TResponsableAtr
getAyudantes():List<TAyudanteAtr>
nuevaVisita(LocalDateTime):void
getVisitas():List<LocalDateTime>
getNumAyudantes():int
getId():int
estaActiva():boolean
setActividad(boolean):void
<<Java Class>>
AtraccA
atracciones
AtraccA()
<<Java Class>>
AtraccB
atracciones
AtraccB()
<<Java Class>>
AtraccC
atracciones
AtraccC()
<<Java Class>>
AtraccD
atracciones
AtraccD()
<<Java Class>>
AtraccE
atracciones
AtraccE()
<<Java Enumeration>>
Descuentos
entradas
ESTUDIANTE: Descuentos
JOVEN: Descuentos
DISCAPACIDAD: Descuentos
porcentajeDesc: float
Descuentos(float)
getDescuento():float
<<Java Class>>
Entrada
entradas
PRECIO_BASE: float
COSTE_VIP: float
PORCEN_MINIMO: float
PRECIO_MINIMO: float
fechaEntrada: LocalDateTime
precioEntrada: float
sumplementoVIP: boolean
ticket: String
Entrada(Turista,LocalDateTime,boolean)
ticket():String
decrementar(float):void
descuento(Descuentos):void
precioEntrada():float
fechaEntrada():LocalDateTime
esVIP():boolean
nuevaEntrada(Turista,LocalDateTime,boolean,boolean):Entrada
<<Java Class>>
ETarde
entradas
ETarde(Turista,LocalDateTime,boolean)
<<Java Class>>
EFamiliar
entradas
x: int
EFamiliar(Turista,LocalDateTime,boolean)
<<Java Class>>
EDiaLaborable
entradas
EDiaLaborable(Turista,LocalDateTime,boolean)
<<Java Class>>
EOtra
entradas
EOtra(Turista,LocalDateTime,boolean)
<<Java Class>>
EGeneral
entradas
EGeneral(Turista,LocalDateTime,boolean)
+trabajadores
0..*
+turistas
0..*
#listaAyudantes
0..*
#responsable
0..1
+atracciones
0..*
+entradas
0..*
~estadisticas
0..1
-estadisticas
0..1
-gastoPersonal
0..1
-gastoPersonal
0..1

4. ANEXO

Turista

```
package turistas;

import principal.Persona;
import java.util.Map;
import java.util.HashMap;
import java.time.LocalDate;
/**
 * <p> Clase que representa a un tipo de turista (cliente) en el parque de atracciones</p>
 * <p> Las caracterÃ-sticas que se tienen en cuenta para definir todo turista
 *  son su nombre y apellidos, y su fecha de nacimiento</p>
 * <p> Para dar cuenta de todas los turistas, se dispone como atributos de clase un HashMap que
 *  almacena todos los turistas que han visitado el parque, ademÃ¡s de un contador del total de los mismos</p>
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public abstract class Turista extends Persona
{
   // Valores estaticos, iguales para todos las instancias de tipo "Turista"
   public static Map<Integer, Turista> turistas = new HashMap<>();
   public static int contadorIdTurista=0;

   //Atributos para cada Turista
   protected LocalDate nacimiento;
   protected int  idTurista;
   protected float descTipoTurista;

   /**
    * <h1><i>Turista</i></h1>
    * <p><code>public Turista(String nombre, String apellidos, LocalDate nacimiento)</code></p>
    * <p> Construye un objeto Turista </p>
    * @param nombre - nombre del turista
    * @param apellidos - apellidos turista
    * @param nacimiento - fecha nacimiento turista
    */
   public Turista(String nombre, String apellidos, LocalDate nacimiento){
      super(nombre, apellidos);
      this.nacimiento = nacimiento;
      this.idTurista = contadorIdTurista++;
      turistas.put(idTurista,this);
   }

   /**
    * <h1><i>getDescuentoTipoTurista</i></h1>
    * <p><code>public float getDescuentoTipoTurista()</code></p>
    * <p> Indica el tipo de descuento aplicado en funciÃ³n del tipo de turista</p>
    * @return el porecentaje de descuento asociado al tipo de turista
    */
   public float getDescuentoTipoTurista() {return descTipoTurista;}

   /**
    * <h1><i>altaTurista</i></h1>
    * <p><code>public static Turista altaTurista (String nombre, String apellidos, LocalDate nacimiento, LocalDate
ahora)</code></p>
    * <p> Crea y devuelve un objeto que hereda de turista, correspondiente a los parÃ¡metros introducidos.
    * Este puede ser <code>Nino</code>, <code>Adulto</code> o <code>Senior</code></p>
    * @param nombre - nombre del turista
    * @param apellidos - apellidos turista
    * @param nacimiento - fecha nacimiento turista
    * @param ahora - momento en el que se realiza el alta del turista
    * @return un objeto <code>Turista</code>
    */
   public static Turista altaTurista (String nombre, String apellidos, LocalDate nacimiento, LocalDate ahora) {
      Turista t =null;
      int edad = nacimiento.until(ahora).getYears();

      if(edad<=12) t = new Nino(nombre,apellidos,nacimiento);
      else if(edad > 12 && edad<65) t = new Adulto(nombre,apellidos,nacimiento);
      else if (edad >=65) t = new Senior(nombre,apellidos,nacimiento);

      return t;
   }
}
```

Nino

```
package turistas;
```

```java
import java.time.LocalDate;
/**
 * Clase que hereda de <code>Turista</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class Nino extends Turista{
    /**
     * Constructor for objects of class Nino
     */
    public Nino(String nombre, String apellidos, LocalDate nacimiento){
        super(nombre,apellidos,nacimiento);
        descTipoTurista=50;
    }
}
```

Adulto

```java
package turistas;

import java.time.LocalDate;
/**
 * Clase que hereda de <code>Turista</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class Adulto extends Turista
{
    /**
     * Constructor for objects of class Adulto
     */
    public Adulto(String nombre, String apellidos, LocalDate nacimiento){
        super(nombre,apellidos,nacimiento);
        descTipoTurista=0;
    }
}
```

Senior

```java
package turistas;

import java.time.LocalDate;
/**
 * Clase que hereda de <code>Adulto</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class Senior extends Adulto
{
    /**
     * Constructor for objects of class Senior
     */
    public Senior(String nombre, String apellidos, LocalDate nacimiento){
        super(nombre,apellidos,nacimiento);
        descTipoTurista=35;
    }
}
```

Entrada

```java
package entradas;

import turistas.Turista;
import java.time.DayOfWeek;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;
/**
 * <p>La clase <code>Entrada</code> representa una entrada emitida en el parque de atracciones
 * para cada nuevo turista</p>
 * <p>Las diferentes caracterÃ-sticas que definen una entrada son: la fecha y hora a la que es emitida,
 * el tipo de turista que la recibe, si incluye bonificaciÃ³n VIP y, por supuesto, su precio</p>
 * <p> Para dar cuenta de todas las entradas, se dispone como atributos de clase una lista que
 *  almacena todas las entradas emitidas, ademÃ¡s de un contador del total de las mismas</p>
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
```

```java
 */
public abstract class Entrada
{
    // Valores estaticos, iguales para todos las instancias de tipo "Entrada"
    public static List<Entrada> entradas = new ArrayList<>();
    public static final float PRECIO_BASE = 60;
    public static final float COSTE_VIP = 50;
    public static final float PORCEN_MINIMO = 10;
    public static final float PRECIO_MINIMO = PRECIO_BASE*PORCEN_MINIMO/100;

    public static enum Descuentos{
        ESTUDIANTE(10), JOVEN(10), DISCAPACIDAD(20);

        private float porcentajeDesc;

        private Descuentos(float desc) { porcentajeDesc=desc;}
        public float getDescuento() {return porcentajeDesc;}
    }
    //Valores para cada Entrada
    protected LocalDateTime fechaEntrada;
    protected float precioEntrada;
    protected boolean sumplementoVIP;
    protected String ticket="";

    /**
     * <h1><i>Entrada</i></h1>
     * <p><code>public Entrada(Turista p, LocalDateTime ahora, boolean vip)</code></p>
     * <p> Construye un objeto Entrada </p>
     * @param turista - objeto <code>Turista</code> propietario de la entrada
     * @param ahora - momento en el que se emite la entrada
     * @param vip - booleano que indica si el turista escoge la bonificaciÃ³n VIP
     */
    public Entrada(Turista turista, LocalDateTime ahora, boolean vip){
        this.fechaEntrada=ahora;
        this.precioEntrada = PRECIO_BASE;
        if(vip) this.precioEntrada += COSTE_VIP;
        this.precioEntrada*=(1-turista.getDescuentoTipoTurista()/100);
        entradas.add(this);
    }

    /**
     * <h1><i> ticket </i></h1>
     * <p><code> public String ticket()</code></p>
     * <p> Imprime el ticket de la entrada adquirida en la que
     * se indicarÃ¡ su hora, precio, tipo de entrada y tipo de turista</p>
     * @param porcenDesc - porcentaje de decrementar a aplicar al precio
     */
    public String ticket() {
        System.out.println(ticket+precioEntrada);
        return ticket+precioEntrada;
    }

    /**
     * <h1><i> decrementar </i></h1>
     * <p><code> public void decrementar(float porcenDesc)</code></p>
     * <p>Aplica un decrementar al precio de la entrada</p>
     * @param porcenDesc - porcentaje de decrementar a aplicar al precio
     */
    public void decrementar(float porcenDesc){
        float aux;
        aux = precioEntrada-PRECIO_BASE*porcenDesc/100;
        precioEntrada=(aux >= PRECIO_MINIMO)? aux: PRECIO_MINIMO;
    }

    /**
     * <h1><i> descuento </i></h1>
     * <p><code> public void descuento(Descuentos d)</code></p>
     * <p>Aplica uno de los descuentos contemplados en el enumerado Descuentos al precio de la entrada</p>
     * @param tipoDescuento - tipo de descuento
     */

    public void descuento(Descuentos tipoDescuento) {
        decrementar((float) tipoDescuento.getDescuento());
    }
    public float precioEntrada() {return precioEntrada;}
    public LocalDateTime fechaEntrada() {return fechaEntrada;}

    /**
     * <h1><i> esVIP </i></h1>
     * <p><code> public boolean esVIP()</code></p>
     * <p>Indica si la entrada lleva asociado el
     * suplemento VIP de "espera preferente".</p>
     * @return valor de suplementoVIP
```

```
     */
    public boolean esVIP(){return sumplementoVIP;}

    /**
     * <h1><i> nuevaEntrada </i></h1>
     * <p><code> public static Entrada nuevaEntrada(Turista t, LocalDateTime ahora, boolean vip, boolean fami-
liar)</code></p>
     * <p>Crea un tipo de entrada segÃºn las carÃ¡cterÃ-sticas del turista y de la fecha en la que se emite.</p>
     * <p>
     * <table>
     * <tr><th></th><th></th></tr>
     * <tr><td>{@link #EFamiliar}:
     *     <td> En el caso de que la famillia conste de 2 adultos y dos niÃ±os
     * <tr><td>{@link #EDiaLaborable}:
     *     <td> Para las visitas en los lunes, martes, miÃ©rcoles y jueves no festivos
     * <tr><td>{@link #ETarde}:
     *     <td> Para las entradas emitidas a partir de las 16:00, que no sean familiares.
     * <tr><td>{@link #EGeneral}:
     *     <td> Entrada estÃ¡ndar para cualquier turista cualquier dia del aÃ±o.
     *<tr><td>{@link #EOtra}:
     *     <td> Para otras entradas diferentes de los demÃ¡s tipos definidos.
     * </table>
     * </p>
     * @param turista - el turista al que se le emite la entrada
     * @param ahora - la fecha y hora en la que se emite la entrada
     * @param vip - valor booleano que indica si solicita el acceso VIP
     * @param familiar - valor booleano que indica si la entrada encaja en la familia tipo habitual
     */
    public static Entrada nuevaEntrada(Turista turista, LocalDateTime ahora, boolean vip, boolean familiar) {
        Entrada e = null;
        // CreaciÃ³n de las entradas segÃºn las caracterÃ-sticas..
        if(familiar) e = new EFamiliar(turista,ahora,vip);

        else if (LocalTime.from(ahora).isAfter(LocalTime.of(16, 0))) e = new ETarde(turista,ahora,vip);

        else if (ahora.getDayOfWeek().equals(DayOfWeek.MONDAY)      ||
                 ahora.getDayOfWeek().equals(DayOfWeek.TUESDAY)    ||
                 ahora.getDayOfWeek().equals(DayOfWeek.WEDNESDAY)||
                 ahora.getDayOfWeek().equals(DayOfWeek.THURSDAY)) e = new EDiaLaborable(turista,ahora,vip);

        else e = new EGeneral(turista,ahora,vip);

        // Aumento o descuento en funciÃ³n de la temporada en las que se emite la entrada
        LocalDate fecha = ahora.toLocalDate();
        float precio = e.precioEntrada();
        if( fecha.isAfter(LocalDate.of(2018, 12, 31)) && fecha.isBefore(LocalDate.of(2019,1,9))
         || fecha.isAfter(LocalDate.of(2019, 4, 14)) && fecha.isBefore(LocalDate.of(2019,4,22))
         || fecha.isAfter(LocalDate.of(2019, 7, 31)) && fecha.isBefore(LocalDate.of(2019,9,1))
         || fecha.isAfter(LocalDate.of(2019, 11, 30)) && fecha.isBefore(LocalDate.of(2020,1,1))) {
           precio+= PRECIO_BASE*(1+15/100);

        }else if(fecha.isAfter(LocalDate.of(2019, 1, 31)) && fecha.isBefore(LocalDate.of(2019,3,1))
           || fecha.isAfter(LocalDate.of(2019, 10, 31)) && fecha.isBefore(LocalDate.of(2019,12,1))) {
           e.decrementar(15);
        }

        //GeneraciÃ³n ticket
        e.ticket ="//////////////////////////\n"
                + "Entrada Parque\n\n"
                + "Nombre: "+turista.getNombre()+" "+turista.getApellidos()+"\n"
                + "Fecha: "+ ahora.getDayOfMonth()+"-"+ahora.getMonthValue()+"-"+ahora.getYear()+"\n"
                + "Hora: " + ahora.getHour()+":"+ahora.getMinute()+"\n\n"
                + "Tipo turista: "+turista.getClass().getSimpleName()+"\n"
                + "Tipo entrada: "+ e.getClass().getSimpleName()+"\n"
                + "Precio: ";

        return e;
    }
}
```

EDiaLaborable

```
package entradas;

import turistas.Turista;
import java.time.LocalDateTime;
/**
 * Clase que hereda de <code>Entrada</code>.
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class EDiaLaborable extends Entrada
```

```
{
    /**
     * Constructor for objects of class EDiaLaborable
     */
    public EDiaLaborable(Turista p, LocalDateTime fecha,boolean vip){
        super(p,fecha,vip);
    }
}
```

EFamiliar

```
package entradas;

import turistas.Turista;
import java.time.LocalDateTime;
/**
 * Clase que hereda de <code>Entrada</code>.
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class EFamiliar extends Entrada
{
   /**
     * Constructor for objects of class EFamiliar
     */
    public EFamiliar(Turista p, LocalDateTime fecha,boolean vip){
        super(p,fecha,vip);
    }
}
```

EGeneral

```
package entradas;

import turistas.Turista;
import java.time.LocalDateTime;
/**
 * Clase que hereda de <code>Entrada</code>.
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class EGeneral extends Entrada
{
    /**
     * Constructor for objects of class EGeneral
     */
    public EGeneral(Turista p, LocalDateTime fecha,boolean vip){
        super(p,fecha,vip);
    }
}
```

ETarde

```
package entradas;

import turistas.Turista;
import java.time.LocalDateTime;
/**
 * Clase que hereda de <code>Entrada</code>.
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class ETarde extends Entrada
{
    /**
     * Constructor for objects of class ETarde
     */
    public ETarde(Turista p, LocalDateTime fecha,boolean vip){
        super(p,fecha,vip);
    }
}
```

EOtra

```
package entradas;

import turistas.Turista;
import java.time.LocalDateTime;
/**
```

```
 * Clase que hereda de <code>Entrada</code>.
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class EOtra extends Entrada
{
    /**
     * Constructor for objects of class EOtra
     */
    public EOtra(Turista p, LocalDateTime fecha,boolean vip){
        super(p,fecha,vip);
    }
}
```

Trabajador

```
package trabajadores;
import principal.*;
import java.util.Map;
import java.util.HashMap;
/**
 * <p> Clase que representa a un trabajador en el parque de atracciones</p>
 * <p> Las caracterÃ-sticas que se tienen en cuenta para definir todo trabajador
 *  son su nombre y apellidos, y su sueldo</p>
 * <p> Para dar cuenta de todas los trabajadores, se dispone como atributos de clase un HashMap que
 *  almacena todos los trabajadores del parque, ademÃ¡s de un contador del total de los mismos</p>
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public abstract class Trabajador extends Persona
{
    // instance variables - replace the example below with your own
    public static Map<Integer,Trabajador> trabajadores = new HashMap<>();
    public static int contadorIdTrabajador=0;
    public static final float SUELDOBASE = 950;

    protected int  idTrabajador;
    protected float sueldo;

   /**
    * <h1><i>Trabajador</i></h1>
    * <p><code>public Trabajador(String nombre, String apellidos, float sueldo)</code></p>
    * <p> Construye un objeto Trabajador </p>
    * @param nombre - nombre del trabajador
    * @param apellidos - apellidos trabajador
    * @param sueldo - sueldo trabajador
    */
    public Trabajador(String nombre, String apellidos, float sueldo){
        super(nombre,apellidos);
        this.idTrabajador = contadorIdTrabajador++;
        this.sueldo = sueldo;
        trabajadores.put(idTrabajador,this);
    }

    public float getSueldo() {return sueldo;}
}
```

TAtencionCliente

```
package trabajadores;
/**
 * Clase que hereda de <code>Trabajador</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class TAtencionCliente extends Trabajador
{
    /**
     * Constructor for objects of class TAtencionCliente
     */
    public TAtencionCliente(String nombre, String apellidos){
        super(nombre,apellidos,TAyudanteAtr.SUELDOBASE*110/100);
    }
}
```

TAyudanteAtr

```java
package trabajadores;
/**
  * Clase que hereda de <code>Trabajador</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class TAyudanteAtr extends Trabajador
{

    /**
     * Constructor for objects of class TAyudanteAtr
     */
    public TAyudanteAtr(String nombre, String apellidos){
        super(nombre,apellidos,SUELDOBASE);
    }
}
```

TRelacionesPublicas

```java
package trabajadores;
/**
 * Clase que hereda de <code>Trabajador</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class TRelacionesPublicas extends Trabajador
{
    /**
     * Constructor for objects of class TRelacionesPublicas
     */
    public TRelacionesPublicas(String nombre, String apellidos){
        super(nombre,apellidos,TAyudanteAtr.SUELDOBASE*120/100);
    }
}
```

TResponsableAtr

```java
package trabajadores;
/**
 * Clase que hereda de <code>Trabajador</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class TResponsableAtr extends Trabajador
{
    /**
     * Constructor for objects of class TResponsableAtr
     */
    public TResponsableAtr(String nombre, String apellidos){
        super(nombre,apellidos,Trabajador.SUELDOBASE*115/100);
    }
}
```

Atraccion

```java
package atracciones;

import java.util.List;
import java.util.Map;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.HashMap;
import trabajadores.*;
/**
 * <p>Clase que representa una atracciÃ³n del parque de atracciones. </p>
 * <p>En una atracciÃ³n se deben definir diversas caracterÃ-sticas como si acepta adultos y/o niÃ±os.
 * CuÃ¡l debe ser la altura mÃ¡xima y/o mÃ-nima permitida para el acceso a la misma.
 * y, finalmente, si estÃ¡ activa o no.Y, si permite el uso de las ventajas de la bonificaciÃ³n VIP.</p>
 *  <p> TambiÃ©n, respecto al personal del parque, un objeto Atraccion contiene
 *  una lista de los ayudantes de atracciÃ³n, asÃ- como tambiÃ©n el trabajor responsable correspondiente</p>
 *  <p> Para dar cuenta de todas las atracciones, se dispone como atributos de clase un HashMap que almacena
 *  todas las atracciones existentes, ademÃ¡s de un contador del total de las mismas.</p>
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public abstract class Atraccion
{
   // Valores estaticos, iguales para todos las instancias de tipo "Turista"
```

```java
public static Map<Integer,Atraccion> atracciones = new HashMap<>();
public static int contadorIdAtraccion=0;
protected int  idAtraccion;// Todas las atracciones tienen que disponer de un identificador
protected boolean adultos; // Toda atraccion debe especificar si se permiten adultos
protected boolean ninos; // Toda atraccion debe especificar si se permiten niÃ±os
protected float alturaMin; // Altura minima requerida de la atraccion si la hay
protected float alturaMax; // Altura maxima requerida de la atraccion si la hay
protected boolean permiteVIP; //Toda atraccion de especificar si permite usuarios VIP o no
protected TResponsableAtr responsable; //Toda atraccion tiene que disponer de un responsable
protected List<TAyudanteAtr> listaAyudantes; // Toda atraccion tiene al menos un ayudante de atraccion
protected int numAyudantes;
protected List<LocalDateTime> visitas; //Almacemana el momento de cada visita recibida
protected boolean activa;

/**
 * <h1><i>Atraccion</i></h1>
 * <p><code>public Atraccion(boolean adultos, boolean ninos, boolean permiteVIP,int numAyudantes)</code></p>
 * <p> Construye un objeto Atraccion </p>
 * @param adultos - booleano que indica si se permiten personas adultas en la atracciÃ³n
 * @param ninos - booleano que indica si se permiten niÃ±os en la atracciÃ³n
 * @param permiteVIP - booleano que indica si es posible aprovechar las ventajas del bono VIP en esta atracciÃ³n
 * @param numAyudantes - el nÃºmero de ayudantes requerido por la atracciÃ³n para su buen funcionamiento
 */
public Atraccion(boolean adultos, boolean ninos, boolean permiteVIP,int numAyudantes){
   this.adultos = adultos;
   this.ninos = ninos;
   this.permiteVIP = permiteVIP;
   this.numAyudantes = numAyudantes;
   this.idAtraccion = contadorIdAtraccion++;
   listaAyudantes = new ArrayList<>();
   visitas = new ArrayList<>();
   atracciones.put(idAtraccion,this);
   activa=true;
}

/**
 * <h1><i>setResponsable</i></h1>
 * <p><code>public void setResponsable(TResponsableAtr res)</code></p>
 * <p> Especifica quien sera el responsable de la atraccion</p>
 * @param res - trabajador responsable de la atraccion
 */
public void setResponsable(TResponsableAtr res){this.responsable = res;}

/**
 * <h1><i>setAyudanteAtr</i></h1>
 * <p><code>public abstract void setAyudanteAtr()</code></p>
 * <p>AÃ±ade un ayudante de atraccion a la lista de ayudantes sin superar el maximo</p>
 * @param ayudante - trabajador ayudante
 * @return verdadero si ha sido posible aÃ±adir un nuevo ayudante, falso en caso contrario
 */
public boolean setAyudanteAtr(TAyudanteAtr ayudante){
   boolean aux = false;
   if(listaAyudantes.size()<numAyudantes){
      listaAyudantes.add(ayudante);
      aux = true;
   }
   return aux;
}

/**
 * <h1><i>getResponsable</i></h1>
 * <p><code>public TResponsableAtr getResponsable()</code></p>
 * <p>Devuleve el responsable de la atraccion</p>
 * @return el trabajador tipo TResponsableAtr de la atraccion
 */
public TResponsableAtr getResponsable(){return responsable;}

/**
 * <h1><i>getAyudantes</i></h1>
 * <p><code>public List<TAyudanteAtr> getAyudantes()</code></p>
 * <p></p>
 * @return la lista de los ayudantes del responsable de la atracion
 */
public List<TAyudanteAtr> getAyudantes(){return listaAyudantes;}

/**
 * <h1><i>nuevaVisita</i></h1>
 * <p><code>public void nuevaVisita(LocalDateTime visita)</code></p>
 * <p>AÃ±ade a la lista de visitas de la atracciÃ³n el instante
 * que se produce la misma</p>
 * @param visita - momento (fecha y hora) en el que se produce la visita
 */
public void nuevaVisita(LocalDateTime visita) {visitas.add(visita);}
```

```
    /**
     * <h1><i>getVisitas</i></h1>
     * <p><code> public List<LocalDateTime> getVisitas()</code></p>
     * <p>Devuleve una lista con las fechas correspondientes
     * a las visitas recibidas por la atracciÃ³n</p>
     * @return la lista de las visitas recibidas por la atracciÃ³n
     */
    public List<LocalDateTime> getVisitas() { return visitas;}

    /**
     * <h1><i>getNumAyudantes</i></h1>
     * <p><code> public int getNumAyudantes() </code></p>
     * <p>Devuleve el nÃºmero total de ayudantes que debe tener la atracciÃ³n</p>
     * @return nÃºmero entero de ayudantes necesarios
     */
    public int getNumAyudantes() {return numAyudantes;}

    /**
     * <h1><i>getId</i></h1>
     * <p><code> public int getId() </code></p>
     * <p>Devuleve el nÃºmero identificador de la atracciÃ³n</p>
     * @return ID de la atracciÃ³n
     */
    public int getId() {return idAtraccion;}

    /**
     * <h1><i>estaActiva</i></h1>
     * <p><code> pestaActiva() </code></p>
     * <p>Indica si la atracciÃ³n estÃ¡ activa para su uso /p>
     * @return estado de actividad de la atracciÃ³n
     */
    public boolean estaActiva() {return activa;}

    /**
     * <h1><i>setActividad</i></h1>
     * <p><code>public void setActividad(boolean estado) </code></p>
     * <p>Activa o desactiva la atracciÃ³n para su uso</p>
     * @param nuevo estado de actividad de la atracciÃ³n
     */
    public void setActividad(boolean estado) {this.activa=estado;}
}
```

AtraccA

```
package atracciones;
/**
 * Clase que hereda de <code>Atraccion</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class AtraccA extends Atraccion
{
    /**
     * Constructor for objects of class AtraccA
     */
    public AtraccA()
    {
        super(true,true,true,6);
        this.alturaMin=12/10;
    }
}
```

AtraccB

```
package atracciones;
/**
 * Clase que hereda de <code>Atraccion</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class AtraccB extends Atraccion
{
    /**
     * Constructor for objects of class AtraccB
     */
    public AtraccB()
    {
        super(true,true,false,5);
        this.alturaMin=12/10;
        this.alturaMax=19/10;
    }
```

```
}
```

AtraccC

```
package atracciones;
/**
 * Clase que hereda de <code>Atraccion</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class AtraccC extends Atraccion
{
    /**
     * Constructor for objects of class AtraccC
     */
    public AtraccC()
    {
        super(false,true,false,3);
        this.alturaMax=12/10;
    }
}
```

AtraccD

```
package atracciones;
/**
 * Clase que hereda de <code>Atraccion</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class AtraccD extends Atraccion
{
    /**
     * Constructor for objects of class AtraccD
     */
    public AtraccD()
    {
        super(true,true,true,5);
    }
}
```

AtraccE

```
package atracciones;
/**
 * Clase que hereda de <code>Atraccion</code>.
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class AtraccE extends Atraccion
{
    /**
     * Constructor for objects of class AtraccD
     */
    public AtraccE()
    {
        super(true,false,true,7);
    }
}
```

Parque

```
package principal;

import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;
import java.util.Random;
import atracciones.*;
import entradas.*;
import trabajadores.*;
import turistas.*;

/**
 * <p>Clase principal de un parque de atracciones</p>
 *
```

```
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class Parque
{
   public static LocalDateTime fechaActual=LocalDateTime.now();
   public static final int ANO =2019;
   // ===== GeneraciÃ³n aleatoria de numeros
   static Random aleatorizador = new Random();
   static Estadisticas estadisticas= new Estadisticas(ANO);
   static AtraccionesFuncionando gastoPersonal = new AtraccionesFuncionando(ANO);

   public static void main(String[] args){

      generarAtracciones();
      generarTrabajadores();
      generarTuristas();
      //GUI
      GUI gui = new GUI();
      gui.setVisible(true);
   }
   public static void generarAtracciones() {
      // ===== CreaciÃ³n de Atracciones
      //-- Atracciones Tipo A
      for(int i = 0;i<4; i++) new AtraccA();
      //-- Atracciones Tipo B
      for(int i = 0;i<6; i++) new AtraccB();
      //-- Atracciones Tipo C
      for(int i = 0;i<4; i++) new AtraccC();
      //-- Atracciones Tipo D
      for(int i = 0;i<3; i++) new AtraccD();
      //-- Atracciones Tipo E
      for(int i = 0;i<7; i++) new AtraccE();
   }

   public static void generarTrabajadores() {
      for(Atraccion a : Atraccion.atracciones.values()) {

         String nombre ="NombreResponsable_"+Trabajador.contadorIdTrabajador;
         String apellidos="ApellidosResponsable_"+Trabajador.contadorIdTrabajador;
         a.setResponsable(new TResponsableAtr(nombre,apellidos));

         for(int i =0; i<a.getNumAyudantes();i++) {
            nombre ="Ayudante_"+Trabajador.contadorIdTrabajador;
            apellidos="ApellidosAyudante_"+Trabajador.contadorIdTrabajador;
            a.setAyudanteAtr(new TAyudanteAtr(nombre,apellidos));
         }
      }
   }

   public static void generarTuristas() {
      // Se recorre cada dia del aÃ±o 2019 empezando por el 2019-01-01
      LocalDate fecha = LocalDate.of(ANO, 01, 01);
      LocalDate fechafin = fecha.plusYears(1);
      //Mientras la fecha sea anterior a la del aÃ±o siguiente se irÃ¡n recorriendo los dias del aÃ±o
      while(fecha.isBefore(fechafin)) {

         for(int grupo=0; grupo<aleatorizador.nextInt(30); grupo++) {
            List<Turista> grupoTuristas = new ArrayList<>();

            //Hora aleatoria
            LocalTime horaRandom =  LocalTime.of(aleatorizador.nextInt(24), aleatorizador.nextInt(60));

            //Contamos los adultos, niÃ±os y senior que forman el grupo
            int numAdultos =0;
            int numNinos =0;
            int numSenior =0;

            //Siempre irÃ¡ almenos un adulto (incluyendo senior dentro deste)
            String nombre ="Tnombre_"+Turista.contadorIdTurista;
            String apellidos="Tapellidos_"+Turista.contadorIdTurista;
            LocalDate nacimiento=LocalDate.now().minusYears(aleatorizador.nextInt(47)+18);
            Turista tu= Turista.altaTurista(nombre, apellidos, nacimiento, fechaActual.toLocalDate());
            grupoTuristas.add(tu);

            int otrosMiembros = aleatorizador.nextInt(8); //Maximo numero de miembros grupo = 8

            //SelecciÃ³n aleatoria del tipo de turista que serÃ¡ cada miembro del grupo
            for(int miembro=0;miembro<otrosMiembros;miembro++) {
               nombre ="Tnombre_"+Turista.contadorIdTurista;
               apellidos="Tapellidos_"+Turista.contadorIdTurista;
               nacimiento=LocalDate.now().minusYears(aleatorizador.nextInt(100));
               tu= Turista.altaTurista(nombre, apellidos, nacimiento, fecha);
```

```
                grupoTuristas.add(tu);
                if(Adulto.class.isInstance(tu)) numAdultos++;
                else if(Nino.class.isInstance(tu)) numNinos++;
                else if(Senior.class.isInstance(tu)) numSenior++;

                //El turista realiza las visitas a direntes atracciones
                for(int id=0; id<aleatorizador.nextInt(Atraccion.contadorIdAtraccion);id++) {
                   Atraccion.atracciones.get(aleatorizador.nextInt(Atraccion.contadorIdAtraccion)).nuevaVisita(LocalDate-
Time.of(fecha, horaRandom));
                }
             }
             //CreaciÃ³n de las entradas correspondientes a cada grupo de Turistas
             for(Turista t : grupoTuristas) {
                LocalDateTime momento = LocalDateTime.of(fecha, horaRandom);
                // CreaciÃ³n de las entradas segÃºn las caracterÃ-sticas..
                Entrada.nuevaEntrada(t, momento, aleatorizador.nextBoolean(), (grupo ==4 && numAdultos==2 && numNinos==2));
             }
          }
          fecha=fecha.plusDays(1); //Pasamos al siguiente dia del aÃ±o
       }
    }
}
```

Persona

```
package principal;
/**
 * Abstract class Persona - write a description of the class here
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public abstract class Persona
{
    protected String nombre;
    protected String apellidos;
    /**
     */
    public Persona(){}

    public Persona(String nombre, String apellidos)
    {
        this.nombre     = nombre;
        this.apellidos  = apellidos;
    }

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

AtraccionesFuncionando

```
package principal;

import java.time.LocalDate;
import java.time.Year;
import java.time.temporal.IsoFields;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import atracciones.Atraccion;
import trabajadores.Trabajador;

/**
 * La clase AtraccionesFuncionando  indicara, para un aÃ±o natural, que atracciones de las que estan disponibles
 * estan activas (el usuario puede usarlas) o no. Las atracciones que esten inactivas implica que no requeriran
 * de los servicios de los trabajadores correspondientes, lo que hace fluctuar los costes laborales.
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class AtraccionesFuncionando
{
   private int ano;
   private int diasAno;
   // Datos de gastos
   private float promGastoDiario;
   private float promGastoSemanal;
   private float promGastoMensual;
```

```
    private float promGastoAnual;

    private Map<Integer,Float> gastosDia;
    private Map<Integer,Float> gastosSemana;
    private Map<Integer,Float> gastosMes;

    /**
     * <h1><i>AtraccionesFuncionando</i></h1>
     * <p><code>public AtraccionesFuncionando(int ano)</code></p>
     * <p> Construye un objeto AtraccionesFuncionando </p>
     * @param ano - aÃ±o para el que se realiza el analisis del gasto
     */
    public AtraccionesFuncionando(int ano) {
        this.ano=ano;
        //Dias del aÃ±o del cual realizamos las estadisticas
        diasAno = Year.of(ano).length();

        promGastoDiario = 0;
        promGastoSemanal = 0;
        promGastoMensual = 0;
        promGastoAnual = 0;

        gastosDia  = new HashMap<>(diasAno);
        gastosSemana= new HashMap<>(53);
        gastosMes  = new HashMap<>(12);
    }

    public void calcular() {
        gastosDia.clear();
        gastosSemana.clear();
        gastosMes.clear();
        promGastoAnual=0;

        for(Atraccion a: Atraccion.atracciones.values()) {
            //Pagamos a los empleados cada uno de los 12 meses

            for (int mes =1; mes<=12;mes++) {
                sumaGasto(a.getResponsable().getSueldo(), LocalDate.of(ano, mes, 28));

                for(Trabajador t : a.getAyudantes()) sumaGasto(t.getSueldo(), LocalDate.of(ano, mes, 28));
            }
        }
        promGastoMensual = promGastoAnual/12;
        promGastoDiario = promGastoAnual/diasAno;
        promGastoSemanal = promGastoDiario*7;
    }

    private void sumaGasto(float cantidad, LocalDate fecha) {
        int dia = fecha.getDayOfYear();
        int semana =fecha.get(IsoFields.WEEK_OF_WEEK_BASED_YEAR);
        int mes = fecha.getMonthValue();

        gastosDia.put(dia,gastosDia.getOrDefault(dia,(float)0)+cantidad);
        gastosSemana.put(semana,gastosSemana.getOrDefault(semana,(float)0)+cantidad);
        gastosMes.put(mes,gastosMes.getOrDefault(mes,(float)0)+cantidad);
        promGastoAnual +=cantidad;
    }

    public String[] listaAyudantes(int idAtraccion) {
        List<String> listaAyudantes= new ArrayList<>();
        for(Trabajador trabajador : Atraccion.atracciones.get(idAtraccion).getAyudantes()) listaAyudantes.add(trabaja-
dor.getNombre());
        return listaAyudantes.toArray(new String[0]);
    }

    // Metodos de gastos
    public float gastoDia   (int fecha) {return gastosDia.getOrDefault(fecha,(float)0);}

    public float gastoSemana(int fecha) {return gastosSemana.getOrDefault(fecha,(float)0);}

    public float gastoMes   (int fecha) {return gastosMes.getOrDefault(fecha,(float)0);}

    public float gastoAno   (int fecha) {return gastoPromAno();}

    public float gastoPromDia  () {return promGastoDiario;}

    public float gastoPromSemana () {return promGastoSemanal;}

    public float gastoPromMes  () {return promGastoMensual;}

    public float gastoPromAno  () {return promGastoAnual;}

}
```

Estadisticas

```java
package principal;

import java.time.LocalDateTime;
import java.time.Year;
import java.time.temporal.IsoFields;
import java.util.HashMap;
import java.util.Map;
import atracciones.Atraccion;
import entradas.Entrada;

/**
 * La clase <code>Estadisticas</code> representa la informaciÃ³n del parque
 *  de atracciones respecto a los turistas, atracciones y precios de entradas,
 *  analizando valores para los distintos dias, semanas, meses y aÃ±os;
 *  ademÃ¡s de sus promedios
 *
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class Estadisticas
{
   public final int diasAno;

   // Estadisticas de visitantes
   private float promVisitDiario;
   private float promVisitSemanal;
   private float promVisitMensual;
   private float promVisitAnual;
   private Map<Integer,Integer> visitantesDias;
   private Map<Integer,Integer> visitantesSemanas;
   private Map<Integer,Integer> visitantesMeses;

   // Estadisticas de Precios
   private float promPrecioAnual;
   private Map<Integer,Float> preciosDia;
   private Map<Integer,Float> preciosSemana;
   private Map<Integer,Float> preciosMes;

   // Estadisticas de Atracciones
   private Map<Integer, Map<Integer,Integer>>  atraccionesDia;
   private Map<Integer, Map<Integer,Integer>>  atraccionesSemana;
   private Map<Integer, Map<Integer,Integer>>  atraccionesMes;

   /**
    * <h1><i>Estadisticas</i></h1>
    * <p><code>public Estadisticas(int ano)</code></p>
    * <p> Construye un objeto Estadisticas </p>
    * @param ano - aÃ±o para el que se realizan las estadisticas
    */
   public Estadisticas(int ano) {
      //Dias del aÃ±o del cual realizamos las estadisticas
      diasAno = Year.of(ano).length();

      // Estadisticas de visitantes =======================================
      promVisitSemanal=0;
      promVisitMensual=0;
      promVisitAnual=0;
      visitantesDias  = new HashMap<>(diasAno);
      visitantesSemanas= new HashMap<>(53);
      visitantesMeses = new HashMap<>(12);

      // Estadisticas de Precios ==========================================
      preciosDia    = new HashMap<>(diasAno);
      preciosSemana= new HashMap<>(53);
      preciosMes    = new HashMap<>(12);

      // Estadisticas de Atracciones ======================================
      atraccionesDia = new HashMap<>(diasAno);
      atraccionesSemana = new HashMap<>(53);
      atraccionesMes = new HashMap<>(12);
   }

   /**
    * <h1><i>calcular</i></h1>
    * <p><code>public void calcular()</code></p>
    * <p> Realiza todos los cÃ¡lculos estadÃ-sticos en relaciÃ³n a las
    * entradas, turistas, trabajadores y atracciones</p>
    */
   public void calcular() {
      LocalDateTime f; //Fecha entrada
      float p; // Precio entrada
```

```
        // El promedio de visitas anuales es equivalemte a todas las entradas a lo largo del 2019
        promVisitAnual = Entrada.entradas.size();
        promVisitMensual = promVisitAnual/12;
        promVisitDiario = promVisitAnual/diasAno;
        promVisitSemanal = promVisitDiario*7;

        preciosDia.clear();
        preciosSemana.clear();
        preciosMes.clear();
        promPrecioAnual=0;

        int dia=0;
        int semana=0;
        int mes=0;

        for(Entrada e: Entrada.entradas) {
            f = e.fechaEntrada();
            dia = f.getDayOfYear();
            semana =f.get(IsoFields.WEEK_OF_WEEK_BASED_YEAR);
            mes = f.getMonthValue();

            // Estadisticas de visitantes ====================================
            visitantesDias.put(dia,visitantesDias.getOrDefault(dia,0)+1);
            visitantesSemanas.put(semana,visitantesSemanas.getOrDefault(semana,0)+1);
            visitantesMeses.put(mes,visitantesMeses.getOrDefault(mes,0)+1);

            // Estadisticas de Precios =======================================
            p = e.precioEntrada();
            promPrecioAnual +=p;
            preciosDia.put(dia,preciosDia.getOrDefault(dia,(float)0)+p);
            preciosSemana.put(semana,preciosSemana.getOrDefault(semana,(float)0)+p);
            preciosMes.put(mes,preciosMes.getOrDefault(mes,(float)0)+p);

        }
        // Estadisticas de Atracciones ===================================

        for(Atraccion a : Atraccion.atracciones.values()) {
            int idAtraccion = a.getId();
            Map<Integer, Integer> aD =   new HashMap<>();
            Map<Integer, Integer> as =   new HashMap<>();
            Map<Integer, Integer> aM =   new HashMap<>();

            for(LocalDateTime fechaHora : a.getVisitas() ) {
                dia = fechaHora.getDayOfYear();
                semana =fechaHora.get(IsoFields.WEEK_OF_WEEK_BASED_YEAR);
                mes = fechaHora.getMonthValue();

                aD.put(dia,aD.getOrDefault(dia,0)+1);
                as.put(semana,as.getOrDefault(semana,0)+1);
                aM.put(mes,aM.getOrDefault(mes,0)+1);
            }
            atraccionesDia.put(idAtraccion,aD);
            atraccionesSemana.put(idAtraccion,as);
            atraccionesMes.put(idAtraccion,aM);
        }
    }

    // Metodos de visitantes
    public int visitantesDia  (int fecha) {return visitantesDias.getOrDefault(fecha,0);}

    public int visitantesSemana  (int fecha) {return visitantesSemanas.getOrDefault(fecha,0);}

    public int visitantesMes  (int fecha) {return visitantesMeses.getOrDefault(fecha,0);}
    public int visitantesAno  (int fecha) {return (int)promVisitAnual;}

    public float visitantesPromDia   () {return promVisitDiario;}

    public float visitantesPromSemana() {return promVisitSemanal;}

    public float visitantesPromMes   () {return promVisitMensual;}

    public float visitantesPromAno   () {return promVisitAnual;}

    // Metodos de Precios
    public float precioDia  (int fecha) {return preciosDia.getOrDefault(fecha,(float) 0)/visitantesDia(fecha);}

    public float precioSemana(int fecha) {return preciosSemana.getOrDefault(fecha,(float) 0)/visitantesSemana(fecha);}

    public float precioMes  (int fecha) {return preciosMes.getOrDefault(fecha,(float) 0)/visitantesMes(fecha);}

    public float precioAno  (int fecha) {return precioPromAno();}
```

```
   public float precioPromDia     () {return precioPromAno();}

   public float precioPromSemana () {return precioPromAno();}

   public float precioPromMes     () {return precioPromAno();}

   public float precioPromAno     () {return promPrecioAnual/promVisitAnual;}

   // Metodos de Atracciones
   public int atraccionDia    (int fecha, int idAtraccion) { return atraccionesDia.get(idAtraccion).getOrDefault(fe-
cha,0);}

   public int atraccionSemana (int fecha, int idAtraccion) {return atraccionesSemana.get(idAtraccion).getOrDefault(fe-
cha,0);}

   public int atraccionMes    (int fecha, int idAtraccion) {return atraccionesMes.get(idAtraccion).getOrDefault(fecha,0);}

   public int atraccionAno    (int fecha, int idAtraccion) {
      int i = 0;
      for(int k=1; k<=12;k++) i+=atraccionesMes.get(idAtraccion).getOrDefault(k,0);
      return i
            ;}

   public float atraccionPromDia  (int idAtraccion) {return (float) atraccionAno(0,idAtraccion)/diasAno;}

   public float atraccionPromSemana  (int idAtraccion) {return atraccionPromDia(idAtraccion)*7;}

   public float atraccionPromMes  (int idAtraccion) {return (float) atraccionAno(0,idAtraccion)/12;}

   public float atraccionPromAno  (int idAtraccion) {return (float) atraccionAno(0,idAtraccion);}
}
```

GUI

```
package principal;
import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridLayout;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.text.ParseException;
import java.time.LocalDate;
import java.time.Year;
import javax.swing.*;
import javax.swing.text.MaskFormatter;
import atracciones.Atraccion;
import entradas.Entrada;
import turistas.Turista;
/**
 * <p>Interfaz grÃ¡fica para la gestiÃ³n de la aplicaciÃ³n del parque de atracciones</p>
 * <p>Esta cuenta con 5 pestaÃ±as principales. Permitiendo genstionar el alta de
 * nuevos turistas, trabajadores y atracciones; ademÃ¡s del anÃ¡lisis estadÃ-stico y control de gastos </p>
 * @author Periscal Porteiro, Juan
 * @version 17/05/2019
 */
public class GUI extends JFrame{
   private Estadisticas estadisticas;
   private AtraccionesFuncionando gastoPersonal;

   int atraccionSeleccionada=0;

   /**
    * <h1><i>GUI</i></h1>
    * <p><code>public GUI()</code></p>
    * <p> Construye un objeto GUI </p>
    */
   public GUI() {
      /* Almacenamos en la variable objeto "mipantalla" el sistema nativo de ventanas*/
      Toolkit miPantalla = Toolkit.getDefaultToolkit();

      /* Almacenamos en la variable Dimension la resolucion de la pantalla de mi monit*/
      Dimension resolucionPantalla = miPantalla.getScreenSize();

      int alturaPantalla = resolucionPantalla.height;
      int anchoPantalla = resolucionPantalla.width;

      super.setBounds(anchoPantalla /2, alturaPantalla / 2, anchoPantalla / 2, alturaPantalla / 2);

      this.estadisticas = Parque.estadisticas;
      this.gastoPersonal = Parque.gastoPersonal;

      // ------------------------------------------------------------
      Integer[] numDias = new Integer[Year.of(Parque.ANO).length()];
```

```
        Integer[] numSemanas = new Integer [53];
        Integer[] numMeses = new Integer [12];

        for(int dia  = 0; dia<numDias.length;     dia++)       numDias[dia]=dia+1;
        for(int semana  = 0; semana<numSemanas.length;  semana++)    numSemanas[semana]=semana+1;
        for(int mes  = 0; mes<numMeses.length;    mes++)       numMeses[mes]=mes+1;

        //============================================================//
        //------------- PESTAÃ‘A registroCliente ----------------------//
        //============================================================//
        JPanel registroCliente = new JPanel(new GridLayout(7,2));

        registroCliente.add( new JLabel("Nombre"));
        JTextField nombreTu = new JTextField(6);registroCliente.add(nombreTu);

        registroCliente.add( new JLabel("Apellidos"));
        JTextField apellidosTu = new JTextField(6);registroCliente.add(apellidosTu);

        registroCliente.add( new JLabel("Nacimiento (AAAA-MM-dd)"));
        // Mascara para cada cuadro de texto de fecha
        MaskFormatter mascara=new MaskFormatter();
        try {
           mascara.setMask("####-##-##");
        } catch (ParseException e1) {e1.printStackTrace();  }
        mascara.setPlaceholderCharacter('_');
        JTextField nacimiento = new JFormattedTextField(mascara);
        registroCliente.add(nacimiento);

        JCheckBox vip          = new JCheckBox("VIP");          registroCliente.add(vip);
        JCheckBox familiar     = new JCheckBox("Familiar"); registroCliente.add(familiar);
        JCheckBox joven     = new JCheckBox("Carnet Joven");registroCliente.add(joven);
        JCheckBox estudiante   = new JCheckBox("Estudiante");  registroCliente.add(estudiante);
        JCheckBox discFuncional = new JCheckBox("Discapacidad");registroCliente.add(discFuncional);


        JButton botonCliente = new JButton(new AbstractAction() {
           @Override public void actionPerformed(ActionEvent arg0) {
              Turista t= null;
              Entrada e =null;
              t =Turista.altaTurista(nombreTu.getText(), apellidosTu.getText(), LocalDate.parse(nacimiento.getText()), Par-
que.fechaActual.toLocalDate());
              e =Entrada.nuevaEntrada(t, Parque.fechaActual, vip.isSelected(), familiar.isSelected());
              if(estudiante.isSelected()) e.descuento(Entrada.Descuentos.ESTUDIANTE);
              if(joven.isSelected()) e.descuento(Entrada.Descuentos.JOVEN);
              if(discFuncional.isSelected()) e.descuento(Entrada.Descuentos.DISCAPACIDAD);
              e.ticket();
           }
        });
        botonCliente.setText("Guardar");
        registroCliente.add(botonCliente);

        //============================================================//
        //------------- PESTAÃ‘A registroTrabajador --------------------//
        //============================================================//
        JPanel registroTrabajador = new JPanel(new GridLayout(4,2));

        registroTrabajador.add( new JLabel("Nombre"));
        JTextField nombreTra = new JTextField(6);
        registroTrabajador.add(nombreTra );

        registroTrabajador.add( new JLabel("Apellidos"));
        JTextField apellidosTra  = new JTextField(6);
        registroTrabajador.add(apellidosTra );

        JButton botonTrabajador = new JButton("Guardar");
        registroTrabajador.add(botonTrabajador);

        //============================================================//
        //------------- PESTAÃ‘A registroAtraccion --------------------//
        //============================================================//
        JPanel registroAtraccion = new JPanel();

        //============================================================//
        //------------- PESTAÃ‘A estadisticas -------------------------//
        //============================================================//

        //------ PANEL de TURISTAS -------------
        JComboBox<Integer> cvd = new JComboBox<>((Integer[]) numDias);
        JComboBox<Integer> cvs = new JComboBox<>((Integer[]) numSemanas);
        JComboBox<Integer> cvm = new JComboBox<>((Integer[]) numMeses);
        JComboBox<Integer> cva = new JComboBox<>(new Integer[0]);

        JTextField tvd = new JTextField(3);
```

```
        JTextField tvs = new JTextField(3);
        JTextField tvm = new JTextField(3);
        JTextField tva = new JTextField(3);

        cvd.addItemListener( arg0 -> tvd.setText(String.valueOf(estadisticas.visitantesDia((int) cvd.getSelectedItem()))));
        cvs.addItemListener( arg0 -> tvs.setText(String.valueOf(estadisticas.visitantesSemana((int)cvs.getSelectedI-
tem()))));
        cvm.addItemListener( arg0 -> tvm.setText(String.valueOf(estadisticas.visitantesMes((int)cvm.getSelectedItem()))));
        cva.addItemListener( arg0 -> tva.setText(String.valueOf(estadisticas.visitantesAno((int)cva.getSelectedItem()))));

        JPanel visitantes  = new JPanel(new GridLayout(6,3,2,2));
        visitantes.add(new JLabel("TURISTAS"));
        visitantes.add(new JLabel("Promedio"));
        visitantes.add(new JLabel("Indice"));
        visitantes.add(new JLabel("EspecÃ-fico"));

        visitantes.add(new JLabel("Dia"));
        JTextField tpvd = new JTextField(3);visitantes.add(tpvd);visitantes.add(cvd);visitantes.add(tvd);
        visitantes.add(new JLabel("Semana"));
        JTextField tpvs = new JTextField(3);visitantes.add(tpvs);visitantes.add(cvs);visitantes.add(tvs);
        visitantes.add(new JLabel("Mes"));
        JTextField tpvm = new JTextField(3);visitantes.add(tpvm);visitantes.add(cvm);visitantes.add(tvm);
        visitantes.add(new JLabel("AÃ±o"));
        JTextField tpva = new JTextField(3);visitantes.add(tpva);visitantes.add(cva);visitantes.add(tva);

        //------- PANEL  de PRECIOS -------------
        JComboBox<Integer> cpd = new JComboBox<>((Integer[]) numDias);
        JComboBox<Integer> cps = new JComboBox<>((Integer[]) numSemanas);
        JComboBox<Integer> cpm = new JComboBox<>((Integer[]) numMeses);
        JComboBox<Integer> cpa = new JComboBox<>(new Integer[0]);

        JTextField tpd = new JTextField(3);
        JTextField tps = new JTextField(3);
        JTextField tpm = new JTextField(3);
        JTextField tpa = new JTextField(3);

        cpd.addItemListener( arg0 -> tpd.setText(String.valueOf(estadisticas.precioDia((int)cpd.getSelectedItem()))));
        cps.addItemListener( arg0 -> tps.setText(String.valueOf(estadisticas.precioSemana((int)cps.getSelectedItem()))));
        cpm.addItemListener(arg0 -> tpm.setText(String.valueOf(estadisticas.precioMes((int)cpm.getSelectedItem()))));
        cpa.addItemListener( arg0 -> tpa.setText(String.valueOf(estadisticas.precioAno((int)cpa.getSelectedItem()))));

        JPanel precios  = new JPanel(new GridLayout(6,3,2,2));
        precios.add(new JLabel("PRECIOS"));
        precios.add(new JLabel("Promedio"));
        precios.add(new JLabel("Indice"));
        precios.add(new JLabel("EspecÃ-fico"));

        precios.add(new JLabel("Dia"));
        JTextField tppd = new JTextField(3);precios.add(tppd);precios.add(cpd);precios.add(tpd);
        precios.add(new JLabel("Semana"));
        JTextField tpps = new JTextField(3);precios.add(tpps);precios.add(cps);precios.add(tps);
        precios.add(new JLabel("Mes"));
        JTextField tppm = new JTextField(3);precios.add(tppm);precios.add(cpm);precios.add(tpm);
        precios.add(new JLabel("AÃ±o"));
        JTextField tppa = new JTextField(3);precios.add(tppa);precios.add(cpa);precios.add(tpa);


        //------ PANEL de ATRACCIONES -------------
        atraccionSeleccionada=0;

        JComboBox<Integer> cad = new JComboBox<>((Integer[]) numDias);
        JComboBox<Integer> cas = new JComboBox<>((Integer[]) numSemanas);
        JComboBox<Integer> cam = new JComboBox<>((Integer[]) numMeses);
        JComboBox<Integer> caa = new JComboBox<>(new Integer[0]);

        JTextField tad = new JTextField(3);
        JTextField tas = new JTextField(3);
        JTextField tam = new JTextField(3);
        JTextField taa = new JTextField(3);

        cad.addItemListener( arg0 -> tad.setText(String.valueOf(estadisticas.atraccionDia((int) cad.getSelectedItem(),atrac-
cionSeleccionada))));
        cas.addItemListener( arg0 -> tas.setText(String.valueOf(estadisticas.atraccionSemana((int)cas.getSelectedI-
tem(),atraccionSeleccionada))));
        cam.addItemListener( arg0 -> tam.setText(String.valueOf(estadisticas.atraccionMes((int)cam.getSelectedItem(),atrac-
cionSeleccionada))));
        caa.addItemListener( arg0 -> taa.setText(String.valueOf(estadisticas.atraccionAno((int)caa.getSelectedItem(),atrac-
cionSeleccionada))));

        //Se crean 2 paneles. Un panel izquierdo donde se podrÃ¡ seleccionar la la atraccion
        // y otro panel,derecho, donde se visualizarÃ¡ sus estadÃ-sticas
        JPanel atracciones = new JPanel();                              //Panel que contendrÃ¡ los dos paneles
```

```
        JPanel pEstadAtracc = new JPanel(new GridLayout(6,3,2,2));  //Panel que contendrÃ¡ las estadÃ-sticas de la atraccio-
nes seleccionada

        //Conseguimos un array con los identificadores de cada una de las atracciones
        Integer[] a = new Integer[Atraccion.atracciones.size()];
        int i=0;
        for(Atraccion at :Atraccion.atracciones.values()) a[i++]=at.getId();

        JList<Integer> lSelecAtrac = new JList<>(a); // Panel que contendrÃ¡ las diferentes atracciones a seleccionar
        lSelecAtrac.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        lSelecAtrac.setLayoutOrientation(JList.VERTICAL);
        lSelecAtrac.setVisibleRowCount(6);
        JScrollPane listScroller = new JScrollPane(lSelecAtrac);

        lSelecAtrac.addListSelectionListener(arg0 -> atraccionSeleccionada = lSelecAtrac.getSelectedValue());

        pEstadAtracc.add(new JLabel("ATRACCIONES"));
        pEstadAtracc.add(new JLabel("Promedio"));
        pEstadAtracc.add(new JLabel("Indice"));
        pEstadAtracc.add(new JLabel("EspecÃ-fico"));

        pEstadAtracc.add(new JLabel("Dia"));
        JTextField tpad = new JTextField(3);pEstadAtracc.add(tpad);pEstadAtracc.add(cad);pEstadAtracc.add(tad);
        pEstadAtracc.add(new JLabel("Semana"));
        JTextField tpas = new JTextField(3);pEstadAtracc.add(tpas);pEstadAtracc.add(cas);pEstadAtracc.add(tas);
        pEstadAtracc.add(new JLabel("Mes"));
        JTextField tpam = new JTextField(3);pEstadAtracc.add(tpam);pEstadAtracc.add(cam);pEstadAtracc.add(tam);
        pEstadAtracc.add(new JLabel("AÃ±o"));
        JTextField tpaa = new JTextField(3);pEstadAtracc.add(tpaa);pEstadAtracc.add(caa);pEstadAtracc.add(taa);

        atracciones.add(listScroller);
        atracciones.add(pEstadAtracc);
        //-----------------------------------------------------------------

        JPanel panelEstad = new JPanel();
        panelEstad.setLayout(new BoxLayout(panelEstad,BoxLayout.Y_AXIS));
        JTabbedPane pestanasEstad = new JTabbedPane();
        JButton calcularEstad = new JButton(new AbstractAction() {
            @Override
            public void actionPerformed(ActionEvent arg0) {
                estadisticas.calcular();
                tpvd.setText(String.valueOf(estadisticas.visitantesPromDia()));
                tpvs.setText(String.valueOf(estadisticas.visitantesPromSemana()));
                tpvm.setText(String.valueOf(estadisticas.visitantesPromMes()));
                tpva.setText(String.valueOf(estadisticas.visitantesPromAno()));

                tppd.setText(String.valueOf(estadisticas.precioPromDia()));
                tpps.setText(String.valueOf(estadisticas.precioPromSemana()));
                tppm.setText(String.valueOf(estadisticas.precioPromMes()));
                tppa.setText(String.valueOf(estadisticas.precioPromAno()));

                tpad.setText(String.valueOf(estadisticas.atraccionPromDia(atraccionSeleccionada)));
                tpas.setText(String.valueOf(estadisticas.atraccionPromSemana(atraccionSeleccionada)));
                tpam.setText(String.valueOf(estadisticas.atraccionPromMes(atraccionSeleccionada)));
                tpaa.setText(String.valueOf(estadisticas.atraccionPromAno(atraccionSeleccionada)));
                /* */
                pestanasEstad.repaint();
            }
        });


        calcularEstad.setText("Calcular");
        pestanasEstad.add("Visitantes",visitantes);
        pestanasEstad.add("Precios", precios);
        pestanasEstad.add("Atracciones", atracciones);

        panelEstad.add(pestanasEstad);
        panelEstad.add(calcularEstad);

        //=============================================================//
        //------------- PESTAÑA gasto ---------------------------------//
        //=============================================================//

        //------- Informacion de atracciones
        JList<Integer> lSelecAtrac2 = new JList<>(a); // Panel que contendrÃ¡ las diferentes atracciones a seleccionar
        lSelecAtrac2.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        lSelecAtrac2.setLayoutOrientation(JList.VERTICAL);
        lSelecAtrac2.setVisibleRowCount(6);
        JScrollPane listScroller2 = new JScrollPane(lSelecAtrac2);

        JPanel empleados = new JPanel();
        empleados.setLayout(new BoxLayout(empleados,BoxLayout.Y_AXIS));
        empleados.add(new JLabel("Responsable"));
```

```
        JTextField responsable = new JTextField(3);empleados.add(responsable);
        empleados.add(new JLabel("Ayudantes"));

        JList<String> ayudantes = new JList<>();
        ayudantes.setSelectionMode(ListSelectionModel.SINGLE_SELECTION);
        ayudantes.setLayoutOrientation(JList.VERTICAL);
        ayudantes.setVisibleRowCount(6);
        JScrollPane listScroller3 = new JScrollPane(ayudantes);

        lSelecAtrac2.addListSelectionListener(arg0 -> {atraccionSeleccionada = lSelecAtrac2.getSelectedValue();
                                    responsable.setText(Atraccion.atracciones.get(atraccionSeleccionada).getResponsa-
ble().getNombre());
                                    ayudantes.setListData(gastoPersonal.listaAyudantes(atraccionSeleccionada));});
        empleados.add(listScroller3);
        JPanel empleadosAtraccion = new JPanel();
        empleadosAtraccion.add(listScroller2);
        empleadosAtraccion.add(empleados);

        //------- JComboBox  de Gastos
        JComboBox<Integer> cgd = new JComboBox<>((Integer[]) numDias);
        JComboBox<Integer> cgs = new JComboBox<>((Integer[]) numSemanas);
        JComboBox<Integer> cgm = new JComboBox<>((Integer[]) numMeses);
        JComboBox<Integer> cga = new JComboBox<>(new Integer[0]);

        JTextField tgd = new JTextField();
        JTextField tgs = new JTextField();
        JTextField tgm = new JTextField();
        JTextField tga = new JTextField();

        cgd.addItemListener( arg0 -> tgd.setText(String.valueOf(gastoPersonal.gastoDia((int)cgd.getSelectedItem()))));
        cgs.addItemListener( arg0 -> tgs.setText(String.valueOf(gastoPersonal.gastoSemana((int)cgs.getSelectedItem()))));
        cgm.addItemListener( arg0 -> tgm.setText(String.valueOf(gastoPersonal.gastoMes((int)cgm.getSelectedItem()))));
        cga.addItemListener( arg0 -> tga.setText(String.valueOf(gastoPersonal.gastoAno((int)cga.getSelectedItem()))));

        JPanel panelEstadGasto = new JPanel(new GridLayout(6,3,2,2));
        panelEstadGasto.add(new JLabel("GASTO"));
        panelEstadGasto.add(new JLabel("Promedio"));
        panelEstadGasto.add(new JLabel("Indice"));
        panelEstadGasto.add(new JLabel("EspecÃ-fico"));

        panelEstadGasto.add(new JLabel("Dia"));
        JTextField tpgd = new JTextField(3);panelEstadGasto.add(tpgd);panelEstadGasto.add(cgd);panelEstadGasto.add(tgd);
        panelEstadGasto.add(new JLabel("Semana"));
        JTextField tpgs = new JTextField(3);panelEstadGasto.add(tpgs);panelEstadGasto.add(cgs);panelEstadGasto.add(tgs);
        panelEstadGasto.add(new JLabel("Mes"));
        JTextField tpgm = new JTextField(3);panelEstadGasto.add(tpgm);panelEstadGasto.add(cgm);panelEstadGasto.add(tgm);
        panelEstadGasto.add(new JLabel("AÃ±o"));
        JTextField tpga = new JTextField(3);panelEstadGasto.add(tpga);panelEstadGasto.add(cga);panelEstadGasto.add(tga);

        //---------------------------------------------------------------

        JPanel panelGasto = new JPanel();
        panelGasto.setLayout(new BorderLayout());

        JButton calcularGasto = new JButton(new AbstractAction() {
           @Override
           public void actionPerformed(ActionEvent arg0) {
              gastoPersonal.calcular();

              tpgd.setText(String.valueOf(gastoPersonal.gastoPromDia()));
              tpgs.setText(String.valueOf(gastoPersonal.gastoPromSemana()));
              tpgm.setText(String.valueOf(gastoPersonal.gastoPromMes()));
              tpga.setText(String.valueOf(gastoPersonal.gastoPromAno()));

              panelGasto.repaint();
           }
        });
        calcularGasto.setText("Calcular");

        panelGasto.add(empleadosAtraccion, BorderLayout.WEST);
        panelGasto.add(panelEstadGasto, BorderLayout.CENTER );
        panelGasto.add(calcularGasto, BorderLayout.SOUTH);
        //============================================================//
        //------------------------------------------------------------//
        //============================================================//
        super.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JTabbedPane pestanas = new JTabbedPane();
        pestanas.add("Turistas",registroCliente);
        pestanas.add("Trabajadores",registroTrabajador);
        pestanas.add("Atracciones",registroAtraccion);
        pestanas.add("EstadÃ-sticas",panelEstad);
        pestanas.add("Gasto",panelGasto);
        super.getContentPane().add(pestanas);
```

```
        super.pack();
    }
}
```