

Programación 2

**Trabajo Práctico Integrador**

**Profesores:** Adriana Gaudiani, Daniel Bertaccini.

**Alumno:** Alan Fernandez.

**Mail:** [alanfernandez97@hotmail.com](mailto:alanfernandez97@hotmail.com)

**Nro-Libreta**: 20-40230718

**Alumno:** Alan Veron.

**Mail:** [alanfveron@gmail.com](mailto:alanfveron@gmail.com)

**Nro-Libreta:** 19-40376634

**Comision:** Com-4 tarde

**Año:** 2021

**TAD AsignaciónTurnos**

*Atributos:*

***String nombre***

**mesas:** diccionario<int numDeMesa, mesa m >

**personas:** diccionario <int DNI, persona p>

**IREP**:

No se puede tener dos mesas con mismo número de mesa, al usar un diccionario para mesas donde la clave es el número de mesa esto se cumple ya que no es valido tener dos claves con mismo valor.

No se puede tener registradas dos personas con mismo dni, al usar un diccionario para personas donde la clave es el dni esto se cumple ya que no es valido tener dos claves con mismo valor.

*Operaciones:*

**public void** registrarVotante(**int** dni, String nombre, **int** edad, **boolean** enfPrevia, **boolean** trabaja)– Recibe una persona y la ingresa al sistema. Tira excepción si el dni ya fue ingresado o si la persona es menor a 16 años.

**public** **int** agregarMesa(String tipoMesa, **int** dni) - Crea una mesa del tipo que se recibe como parametro y asignandole como presidente la persona que le corresponde el dni.El presidente asignado se le asigna un turno automáticamente en la mesa creada. Tira excepción en caso de que el dni recibido no esté ingresado en el sistema o ya tenga un turno asignado, y también en caso de que el tipo de mesa sea invalido.

**public** Turno<Integer, Integer> asignarTurno(**int** dni)– Recibe un dni, le asigna un turno y lo retorna. Tira excepción en caso de que la persona no esté ingresada en el sistema.

**public** **int** asignarTurnos() – Asigna turnos automáticamente a las personas registradas en el sistema que no tengan turno.

**public** **boolean** votar(Integer dni) – La persona vota. Tira excepcion si la persona no esta registrada en el sistema o bien no tiene turno, también si ya voto.

**public** **int** votantesConTurno(String tipoDeMesa) - Devuelve la cantidad de votantes asignados, sin importar si votaron o no, del tipo de mesa pasado por parámetro. Tira excepcion si el tipo de mesa recibido es invalido.

**public** Turno<Integer, Integer> consultaTurno(**int** dni) - Devuelve el turno de la persona, en caso de no tener turno devuelve null. Tira excepcion en caso de que la persona no este registrada al sistema.

**public** Map <Integer,List <Integer>> asignadosAMesa (Integer numMesa) –Devuelve un map detallando las franjas horarias de la mesa recibida. Cada llave va a ser un horario y el valor una lista con los dnis de los votantes en dicha hora. Tira excepcion en caso de que la mesa recibida no sea válida.

**public** List<Tupla<String,Integer>> sinTurnoSegunTipoMesa() –Devuelve una lista con tuplas donde sus datos van a ser: una mesa de dada caracteristica y la cantidad de votantes sin turno para dicha mesa.

**TAD Mesa (Abstact)**

*Atributos:*

**private int** numero

**private Votante** presidente

**private int** cupo

**private int static** contador

**private int** horario

**private Votantes**: diccionario <Integer, Votante> Key = dni, Value = datos del votante

**IREP**:

* El numero de mesa debe ser mayor a 0.
* El cupo debe ser mayor a 0.
* El contador debe ser mayor a 0.
* No se puede tener dos personas con igual dni en votantes, esto se cumple ya que usamos un diccionario para votantes donde la clave es el dni

*Operaciones:*

Mesa(Votante presidente)– Crea una asignado como presidente la persona recibida por parametro.

**public** **abstract** Turno<Integer, Integer> agregarTurno(Integer dni) – agrega un turno a una persona según el dni recibido.

**public** **abstract** **boolean** cumpleCondicionDeLaMesa (Votante v);- Determina si la persona pasada como parámetro cumple las condiciones para votar en la mesa actual

**public** Map<Integer, Integer> mostrarListaVotantes() devuelve la lista de votantes de la respectiva mesa.

**public** **void** agregarVotante(**int** dni,Votante votante) – Agrega el votante a la lista votantes de la mesa

**public** **int** cantVotantes()- Devuelve cantidad de votantes de la mesa

**public** **void** agregarTurnoPresidente(Votante v)- Asigna turno automatico al presidente de la mesa

**MesaMayor65**

*Atributos:*

**private int** numero

**private Votante** presidente

**private int** cupo

**private int static** contador

**private int** horario

**private Votantes**: diccionario <Integer, Votante> Key = dni, Value = datos del votante

**IREP**: El cupo tiene que pertencer al rango [ 0 y 100 ].

*Operaciones:*

Mesa(Votante presidente,String tipoMesa)– Crea una mesa.

**public** Turno<Integer, Integer> agregarTurno(Integer dni) agrega un turno a una persona según el dni recibido.

**public** **boolean** cumpleCondicionDeLaMesa (Votante v);- Determina si la persona pasada como parámetro cumple las condiciones para votar en la mesa actual

**public** **void** agregarVotante(**int** dni,Votante votante) – Agrega el votante a la lista votantes de la mesa

**public** **int** cantVotantes()- Devuelve cantidad de votantes de la mesa

**public** Map<Integer, Integer> mostrarListaVotantes() devuelve la lista de votantes de la respectiva mesa.

**public** **void** agregarTurnoPresidente(Votante v)- Asigna turno automatico al presidente de la mesa

**MesaEnfPreex**

*Atributos:*

**private int** numero

**private Votante** presidente

**private int** cupo

**private int static** contador

**private int** horario

**private Votantes**: diccionario <Integer, Votante> Key = dni, Value = datos del votante

**IREP**: El cupo tiene que pertencer al rango [ 0 y 200 ].

Operaciones:

Mesa(Votante presidente,String tipoMesa)– Crea una mesa.

**public** Turno<Integer, Integer> agregarTurno(Integer dni) agrega un turno a una persona según el dni recibido.

**public** **boolean** cumpleCondicionDeLaMesa (Votante v);- Determina si la persona pasada como parámetro cumple las condiciones para votar en la mesa actual

**public** **void** agregarVotante(**int** dni,Votante votante) – Agrega el votante a la lista votantes de la mesa

**public** **int** cantVotantes()- Devuelve cantidad de votantes de la mesa

**public** Map<Integer, Integer> mostrarListaVotantes() devuelve la lista de votantes de la respectiva mesa.

**public** **void** agregarTurnoPresidente(Votante v)- Asigna turno automatico al presidente de la mesa

**MesaTrabajador**

*Atributos:*

**private int** numero

**private Votante** presidente

**private int** cupo

**private int static** contador

**private int** horario

**private Votantes**: diccionario <Integer, Votante> Key = dni, Value = datos del votante

**IREP**: El cupo tiene que ser mayor a 0.

Operaciones:

Mesa(Votante presidente,String tipoMesa)– Crea una mesa.

**public** Turno<Integer, Integer> agregarTurno(Integer dni) agrega un turno a una persona según el dni recibido.

**public** **boolean** cumpleCondicionDeLaMesa (Votante v);- Determina si la persona pasada como parámetro cumple las condiciones para votar en la mesa actual

**public** **void** agregarVotante(**int** dni,Votante votante) – Agrega el votante a la lista votantes de la mesa

**public** **int** cantVotantes()- Devuelve cantidad de votantes de la mesa

**public** Map<Integer, Integer> mostrarListaVotantes() devuelve la lista de votantes de la respectiva mesa.

**public** **void** agregarTurnoPresidente(Votante v)- Asigna turno automatico al presidente de la mesa

**MesaGeneral**

*Atributos:*

**private int** numero

**private Votante** presidente

**private int** cupo

**private int static** contador

**private int** horario

**private Votantes**: diccionario <Integer, Votante> Key = dni, Value = datos del votante

**IREP**: El cupo tiene que pertencer al rango [ 0 y 300 ].

Operaciones:

Mesa(Votante presidente,String tipoMesa)– Crea una mesa.

**public** Turno<Integer, Integer> agregarTurno(Integer dni) agrega un turno a una persona según el dni recibido.

**public** **boolean** cumpleCondicionDeLaMesa (Votante v);- Determina si la persona pasada como parámetro cumple las condiciones para votar en la mesa actual

**public** **void** agregarVotante(**int** dni,Votante votante) – Agrega el votante a la lista votantes de la mesa

**public** **int** cantVotantes()- Devuelve cantidad de votantes de la mesa

**public** Map<Integer, Integer> mostrarListaVotantes() devuelve la lista de votantes de la respectiva mesa.

**public** **void** agregarTurnoPresidente(Votante v)- Asigna turno automatico al presidente de la mesa

**TAD Votante**

*Atributos:*

**private String** nombre

**private Int** dni

**private Int** edad

**private Boolean** enfPrevia

**private Boolean** trabaja

**private Boolean** voto

**private** Turno <Integer,Integer> turno; clave número de mesa y valor horario del turno

**IREP**:

* El dni debe ser mayor que 0.
* La edad debe ser mayor igual que 0.

*Operaciones:*

**Votante** (String nombre,int dni,int edad,boolean enfermedad,boolean trabaja\_dia\_votacion) –Constructor de la clase Persona.

**public** **int** obtenerHorarioTurno()- Obtiene horario del turno del votante

**public** **boolean** tieneTurno()- Indica si la persona tiene un turno asignado

public boolean votar()- Si la persona ya voto devuelve false, si no voto setea el voto en true y devuelve true

**TAD Turno**

*Atributos:*

**final** Integer numMesa;

**final** Integer horario

**IREP:** Numero de mesa y horario deben ser mayor a 0

Operaciones:

**public** Turno(Integer numMesa, Integer horario) – Constructor del turno