Práctica 2

Introducción a POO

Objetivo. Realizar programas que instancien objetos, a partir de clases existentes, y se le envíen mensajes a estos objetos. Manipulación de objetos Strings. Comprender los conceptos: clase, objeto, estado, método, mensaje, referencia.

Nota: Trabajar sobre la carpeta "tema2" del proyecto

1- Se dispone de la clase Persona (en la carpeta tema2). Un objeto persona puede crearse sin valores iniciales o enviando en el mensaje de creación el nombre, DNI y edad (en ese orden). Un objeto persona responde a los siguientes mensajes:

`getNombre()`	retorna el nombre (String) de la persona
`getDNI()`	retorna el dni (int) de la persona
`getEdad()`	retorna la edad (int) de la persona
`setNombre(X)`	modifica el nombre de la persona al "String" pasado por parámetro (X)
`setDNI(X)`	modifica el DNI de la persona al "int" pasado por parámetro (X)
`setEdad(X)`	modifica la edad de la persona al "int" pasado por parámetro (X)
`toString()`	retorna un String que representa al objeto. Ej: "Mi nombre es **Mauro**, mi DNI es **11203737** y tengo **70** años"

Realice un programa que cree un objeto persona con datos leídos desde teclado. Luego muestre en consola la representación de ese objeto en formato String.

2- Utilizando la clase Persona. Realice un programa que almacene en un vector **a lo sumo** 15 personas. La información (nombre, DNI, edad) se debe generar aleatoriamente hasta obtener edad 0. Luego de almacenar la información:

- Informe la cantidad de personas mayores de 65 años.
- Muestre la representación de la persona con menor DNI.

3- Se realizará un casting para un programa de TV. El casting durará a lo sumo 5 días y en cada día se entrevistarán a 8 personas en distinto turno.

a) Simular el proceso de inscripción de personas al casting. A cada persona se le pide nombre, DNI y edad y se la debe asignar en un día y turno de la siguiente manera: las personas primero completan el primer día en turnos sucesivos, luego el segundo día y así siguiendo. La inscripción finaliza al llegar una persona con nombre "ZZZ" o al cubrirse los 40 cupos de casting.

Una vez finalizada la inscripción:

b) Informar para cada día y turno asignado, el nombre de la persona a entrevistar.

NOTA: utilizar la clase Persona. Pensar en la estructura de datos a utilizar. Para comparar Strings use el método `equals`.

4- Sobre un nuevo programa, modifique el ejercicio anterior para considerar que:

a) Durante el proceso de inscripción se pida a cada persona sus datos (nombre, DNI, edad) *y el día* en que se quiere presentar al casting. La persona debe ser inscripta en ese día, en el siguiente turno disponible. En caso de no existir un turno en ese día, informe la situación. La inscripción finaliza al llegar una persona con nombre "ZZZ" o al cubrirse los 40 cupos de casting.

Una vez finalizada la inscripción:

b) Informar para cada día: la cantidad de inscriptos al casting ese día y el nombre de la persona a entrevistar en cada turno asignado.

5- Se dispone de la clase Partido (en la carpeta tema2). Un objeto partido representa un encuentro entre dos equipos (local y visitante). Un objeto partido puede crearse sin valores iniciales o enviando en el mensaje de creación el nombre del equipo local, el nombre del visitante, la cantidad de goles del local y del visitante (en ese orden). Un objeto partido sabe responder a los siguientes mensajes:

`getLocal()`	retorna el nombre (String) del equipo local
`getVisitante()`	retorna el nombre (String) del equipo visitante
`getGolesLocal()`	retorna la cantidad de goles (int) del equipo local
`getGolesVisitante()`	retorna la cantidad de goles (int) del equipo visitante
`setLocal(X)`	modifica el nombre del equipo local al "String" X
`setVisitante(X)`	modifica el nombre del equipo visitante al "String" X
`setGolesLocal(X)`	modifica la cantidad de goles del equipo local al "int" X
`setGolesVisitante(X)`	modifica la cantidad de goles del equipo visitante al "int" X
`hayGanador()`	retorna un boolean que indica si hubo (true) o no hubo (false) ganador
`getGanador()`	retorna el nombre (String) del ganador del partido (si no hubo retorna un String vacío).
`hayEmpate()`	retorna un boolean que indica si hubo (true) o no hubo (false) empate

Implemente un programa que cargue un vector con **a lo sumo 20** partidos disputados en el campeonato. La información de cada partido se lee desde teclado hasta ingresar uno con nombre de visitante "ZZZ" o alcanzar los 20 partidos. Luego de la carga:

- Para cada partido, armar e informar una representación String del estilo:

  {EQUIPO-LOCAL *golesLocal* **VS** EQUIPO-VISITANTE *golesVisitante* }

- Calcular e informar la cantidad de partidos que ganó River.

- Calcular e informar el total de goles que realizó Boca jugando de local.