

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- Responda: ¿Qué imprimen los siguientes programas? ¿Qué efecto tiene la asignación utilizada con objetos? ¿Qué efecto tiene la comparación con == y != utilizada con objetos? ¿Qué retorna el mensaje equals cuando se le envía a un String?

```
public class Ej03QueImprimeA {
    public static void main(String[] args) {
        String saludo1=new String("hola");
        String saludo2=new String("hola");
        System.out.println(saludo1 == saludo2);
        System.out.println(saludo1 != saludo2);
        System.out.println(saludo1.equals(saludo2));
    }
}
```

```
public class Ej03QueImprimeB {
    public static void main(String[] args) {
        Persona p1;
        Persona p2;
        p1 = new Persona();
        p1.setNombre("Pablo Sotile");
        p1.setDNI(11200413);
        p1.setEdad(40);
        p2 = new Persona();
        p2.setNombre("Julio Toledo");
        p2.setDNI(22433516);
        p2.setEdad(51);
        p1 = p2;
        p1.setEdad( p1.getEdad() + 1 );
        System.out.println(p2.toString());
        System.out.println(p1.toString());
        System.out.println( p1 == p2 );
    }
}
```

Taller de Programación 2022 – Módulo POO

4- 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 el nombre de la persona a entrevistar.

NOTA: utilizar la clase Persona y pensar en la estructura de datos a utilizar.

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:

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