Práctica 1

Introducción a Java. Matrices.

Objetivo. Realizar programas simples que lean datos desde teclado, generen datos aleatorios, muestren datos en consola y manipulen variables de tipos simples, String y arreglos. Familiarizarse con el entorno Netbeans.

Nota: Trabajar sobre la carpeta "tema1" del proyecto

1- En el programa Demo08Vector escriba las instrucciones necesarias para:
- generar enteros aleatorios hasta obtener el número 11. Para cada número muestre el resultado de multiplicarlo por 2 (accediendo al vector).

2- Escriba un programa que lea las alturas de los 15 jugadores de un equipo de básquet y las almacene en un vector. Luego informe:
- la altura promedio
- la cantidad de jugadores con altura por encima del promedio

NOTA: Dispone de un esqueleto para este programa en Ej02Jugadores.java

3- Escriba un programa que defina una matriz de enteros de tamaño 10x10. Inicialice la matriz con números aleatorios entre 0 y 200.
Luego realice las siguientes operaciones:
- Mostrar el contenido de la matriz en consola.
- Calcular e informar la suma de todos los elementos almacenados entre las filas 2 y 9 y las columnas 0 y 3
- Generar un vector de 10 posiciones donde cada posición i contiene la suma de los elementos de la columna i de la matriz.
- Lea un valor entero e indique si se encuentra o no en la matriz. En caso de encontrarse indique su ubicación (fila y columna) en caso contrario imprima "No se encontró el elemento".

NOTA: Dispone de un esqueleto para este programa en Ej03Matrices.java

4- Un edificio de oficinas está conformado por 8 pisos y 4 oficinas por piso. Realice un programa que permita informar la cantidad de personas que concurrieron a cada oficina de cada piso. Para esto, simule la llegada de personas al edificio de la siguiente manera: a cada persona se le pide el nro. de piso y nro. de oficina a la cual quiere concurrir. La llegada de personas finaliza al indicar un nro. de piso 9. Al finalizar la llegada de personas, informe lo pedido.

5- El dueño de un restaurante entrevista a cinco clientes y les pide que califiquen (con puntaje de 1 a 10) los siguientes aspectos: (0) Atención al cliente (1) Calidad de la comida (2) Precio (3) Ambiente.
Escriba un programa que lea desde teclado las calificaciones de los cinco clientes para cada uno de los aspectos y almacene la información en una estructura. Luego imprima la calificación promedio obtenida por cada aspecto.

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) );
    }
}
```

4- Realice un programa que cargue un vector con 10 Strings leídos desde teclado. El vector generado tiene un mensaje escondido que se forma a partir de la primera letra de cada string. Genere un nuevo string con el mensaje escondido y muéstrelo en consola.

NOTA: La primer letra de un String se obtiene enviando el mensaje charAt(0) al objeto. Ingrese: humo oso lejos ala menos usado nene de ocho ! Debería imprimir: holamundo!

5- 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.

6- 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 informar:

- La cantidad de partidos que ganó River.
- El total de goles que realizó Boca jugando de local.
- El porcentaje de partidos finalizados con empate.

Práctica 3

Desarrollo de Clases (parte I)

Objetivo. Definir clases para representar objetos del mundo real. Concepto de clase, estado (variables de instancia) y comportamiento (métodos). Instanciación. Mensajes.

Nota: Trabajar sobre la carpeta "tema3" del proyecto

1-A- Definir una clase para representar triángulos. Un triángulo se caracteriza por el tamaño de sus 3 lados (`double`), el color de relleno (`String`) y el color de línea (`String`). El triángulo debe saber:

- Devolver/modificar el valor de cada uno de sus atributos (métodos `get` y `set`)
- Calcular el área y devolverla (método `calcularArea`)
- Calcular el perímetro y devolverlo (método `calcularPerimetro`)

B- Realizar un programa que instancie un triángulo, le cargue información leída desde teclado e informe en consola el perímetro y el área.

NOTA: Calcular el área con la fórmula $\text{Área} = \sqrt{s(s-a)(s-b)(s-c)}$, donde a, b y c son los lados y $s = \frac{a+b+c}{2}$. La función raíz cuadrada es `Math.sqrt(#)`

2-A- Definir una clase para representar balanzas comerciales (para ser utilizadas en verdulerías, carnicerías, etc). Una balanza comercial sólo mantiene el monto y la cantidad de items correspondientes a la compra actual (es decir, no almacena los ítems de la compra). La balanza debe responder a los siguientes mensajes:

- `iniciarCompra()`: inicia el monto y la cantidad de ítems para la compra actual.
- `registrarProducto(pesoEnKg, precioPorKg)`: recibe el "*peso en kilos*" del ítem comprado y su "*precio por kilo*", y actualiza el estado de la balanza.
- `getMonto()`: retorna el monto a pagar por la compra actual.
- `getResumenDeCompra()`: retorna un String del siguiente estilo "`Total a pagar X por la compra de Y productos`" , donde `X` es el monto e `Y` es la cantidad de ítems de la compra.

B- Realizar un programa que instancie una balanza e inicie una compra. Lea desde teclado información de los ítems comprados (peso en kg y precio por kg) hasta ingresar peso 0. Registre cada producto en la balanza. Al finalizar, informe el resumen de la compra.

3-A- Definir una clase para representar entrenadores de un club. Un entrenador se caracteriza por su nombre, sueldo básico, antigüedad y cantidad de campeonatos ganados.

- Defina métodos para obtener/modificar el valor de cada atributo.
- Defina el método `calcularEfectividad` que devuelve la efectividad del entrenador, que es el promedio de campeonatos ganados por año de antigüedad.
- Defina el método `calcularSueldoACobrar` que devuelve el sueldo a cobrar por el entrenador. El sueldo a cobrar es el sueldo básico agregando un 10% del básico por año de antigüedad, y además se adiciona un plus por campeonatos ganados (5000$ si ha ganado entre 1 y 4 campeonatos; $30.000 si ha ganado entre 5 y 10 campeonatos; 50.000$ si ha ganado más de 10 campeonatos).

B- Realizar un programa que instancie un entrenador, cargándole datos leídos desde teclado. Informe el sueldo a cobrar y la efectividad del entrenador.

4-A- Definir una clase para representar círculos. Los círculos se caracterizan por su radio (`double`), el color de relleno (`String`) y el color de línea (`String`). El círculo debe saber:

- Devolver/modificar el valor de cada uno de sus atributos (métodos `get` y `set`)
- Calcular el área y devolverla (método `calcularArea`)
- Calcular el perímetro y devolverlo (método `calcularPerimetro`)

B- Realizar un programa que instancie un círculo, le cargue información leída de teclado e informe en consola el perímetro y el área.

NOTA: la constante PI es `Math.PI`

5-A- Se dispone de la clase Producto (en la carpeta tema3). Un objeto producto puede crearse sin valores iniciales o enviando en el mensaje de creación el "*peso en kg*" y "*descripción*" (en ese orden). Un objeto producto responde a los siguientes mensajes:

`getDescripcion()`	retorna la descripción (String) del producto
`getPesoEnKg()`	retorna el peso en kg (double) del producto
`setDescripcion(X)`	modifica la descripción del producto al "String" X
`setPesoEnKg(X)`	modifica el peso del producto al "double" X

Usando la clase Producto. Realice las siguientes modificaciones a la clase del ejercicio 2-A. Ahora la balanza debe generar un resumen de compra más completo. Agregue a la balanza la característica *resumen* (`String`) y modifique los métodos como se indica a continuación:

- `iniciarCompra` para que además inicie el *resumen* en el String vacío.
- `registrarProducto` para que reciba un objeto Producto y su "*precio por kg*". La operación debe actualizar el monto y cantidad de ítems de la balanza, y concatenar al *resumen* la descripción y el monto a pagar por este producto.
- `getResumenDeCompra` para que retorne un String del siguiente estilo: "Naranja 100$ – Banana 40$ – Lechuga 50$ – Total a pagar **190** pesos por la compra de **3** productos" donde la parte subrayada es el contenido de *resumen*.

B- Realice las modificaciones necesarias en el programa principal solicitado en 2-B para corroborar el funcionamiento de la balanza.

Práctica 4

Desarrollo de Clases (parte II)

Objetivo. Iniciar objetos a partir de constructores. Continuar trabajando los conceptos de la POO. Relacionar clases por asociación/conocimiento.

Nota: Trabajar sobre la carpeta "tema4" del proyecto. Las modificaciones planteadas del ejercicio 1 deben realizarse en la carpeta "tema3"

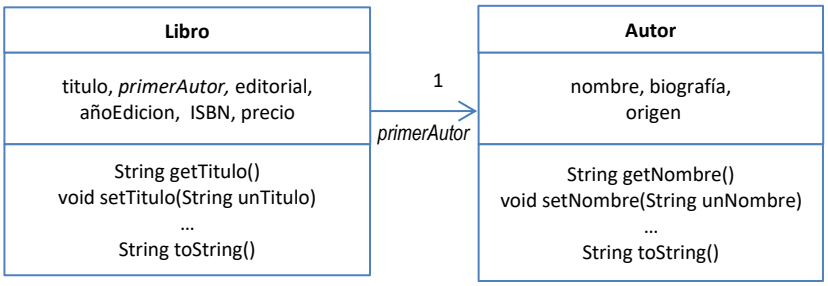
1-A– Defina constructores para las clases Triángulo, Círculo y Entrenador (en la carpeta tema3). En los tres casos declare:

- Un *primer constructor* que reciba todos los datos necesarios para iniciar el objeto.

- Un *segundo constructor* que no posea parámetros ni código (constructor nulo).

B- Incluya en los programas respectivos (de la carpeta tema3) el código necesario para instanciar triángulos, círculos y entrenadores usando en cada caso el *primer constructor.*

2-A- Modifique la clase Libro (carpeta tema 4) para ahora considerar que el primer autor es un objeto instancia de la clase Autor. Implemente la clase Autor, sabiendo que éstos se caracterizan por nombre, biografía y origen y que deben permitir devolver/modificar el valor de sus atributos y devolver una representación String formada por nombre, biografía y origen.



B- Modifique el programa Demo01Constructores (carpeta tema 4) para instanciar los libros con su autor, considerando las modificaciones realizadas. A partir de uno de los libros, obtenga e imprima la representación del autor de ese libro.

3-A- Definir una clase para representar estantes. Un estante almacena a lo sumo 20 libros. Implemente un constructor que permita iniciar el estante sin libros. Provea métodos para:

(i) devolver la cantidad de libros que almacenados **(ii)** devolver si el estante está lleno **(iii)** agregar un libro al estante **(iv)** devolver el libro con un título particular que se recibe.

B- Realice un programa que instancie un estante. Cargue varios libros. A partir del estante, busque e informe el autor del libro "Mujercitas".

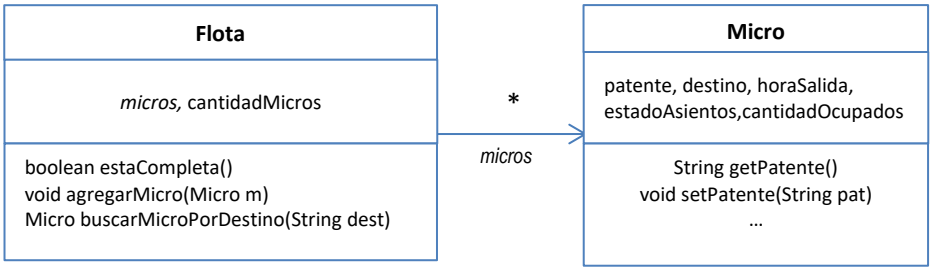
C- Piense: ¿Qué modificaría en la clase definida para ahora permitir estantes que almacenen como máximo N libros? ¿Cómo instanciaría el estante?

4-A- Definir una clase para representar micros. Un micro conoce su patente, destino, hora de salida, el estado de sus 20 asientos (boolean: true *ocupado*, false *libre*) y la cantidad de asientos ocupados al momento. Lea detenidamente a) y b) y luego implemente.

a) Implemente un constructor que permita iniciar el micro con una patente, un destino y una hora de salida (que se reciben) y con sus asientos *libres*.
b) Implemente métodos para:
   i. devolver/modificar patente, destino y hora de salida
   ii. devolver la cantidad de asientos ocupados
   iii. devolver si el micro está lleno
   iv. validar un nro. de asiento que se recibe (es decir, devolver si está en rango o no)
   v. devolver el estado de un nro. de asiento *válido* que se recibe
   vi. ocupar un nro. de asiento *válido* que se recibe
   vii. liberar un nro. de asiento *válido* que se recibe
   viii. devolver el nro. del primer asiento libre

B- Realice un programa que cree un micro con patente "ABC123", destino "Mar del Plata" y hora de salida "5:00". Lea 5 nros. de asiento, que corresponden a pedidos de personas. Para cada nro. ingresado debe: validar el nro.; en caso que esté libre dicho asiento, ocuparlo e informar el éxito de la operación; en caso que esté ocupado, mostrar el nro. del primer asiento libre. Al finalizar, informe la cantidad de asientos ocupados del micro.

5-A- Definir una clase para representar flotas de micros. Una flota conoce a los micros que la componen (a lo sumo 15). Provea un constructor para crear una flota vacía (sin micros). Implemente métodos para:
   i. devolver si la flota está completa (es decir, si tiene 15 micros o no).
   ii. agregar a la flota un micro que se recibe.
   iii. buscar en la flota un micro con destino igual a uno que se recibe y retornarlo (en caso de no existir dicho micro, retornar null).



B- Realice un programa que cree una flota vacía. Cargue varios micros (sin pasajeros) a la flota. Luego, lea un destino e informe si en la flota hay un micro que va a ese destino.

Práctica 5

Herencia

Objetivo. Trabajar con el concepto de herencia y polimorfismo.

Nota: Trabajar sobre la carpeta "tema5" del proyecto

1-A- Incluya la clase Triángulo a la jerarquía de figuras vista (carpeta tema5). Triángulo debe *heredar* de Figura todo lo que es común y *definir* su constructor y sus atributos y métodos propios. Además debe *redefinir* el método `toString`.

B- De igual manera, incluya la clase Círculo a la jerarquía de figuras.

C- Añada a la representación String el valor del perímetro. Piense ¿qué método `toString` debe modificar: el de cada subclase o el de Figura?

D- Añada el método `despintar` que establece los colores de la figura a línea "negra" y relleno "blanco". Piense ¿dónde debe definir el método: en cada subclase o en Figura?

E- Realizar un programa que instancie un triángulo y un círculo. Muestre en consola la representación String de cada uno. Pruebe el funcionamiento del método `despintar`.

2- Queremos representar a los empleados de un club: jugadores y entrenadores.

- Cualquier *empleado* se caracteriza por su nombre, sueldo básico y antigüedad.
- Los *jugadores* son empleados que se caracterizan por el número de partidos jugados y el número de goles anotados.
- Los *entrenadores* son empleados que se caracterizan por la cantidad de campeonatos ganados.

A- Implemente la jerarquía de clases declarando atributos, métodos para obtener/modificar su valor y *constructores* que reciban los datos necesarios.

B- Cualquier empleado debe responder al mensaje `calcularEfectividad`. La efectividad del entrenador es el promedio de campeonatos ganados por año de antigüedad, mientras que la del jugador es el promedio de goles por partido.

C- Cualquier empleado debe responder al mensaje `calcularSueldoACobrar`. El sueldo a cobrar es el sueldo básico más un 10% del básico por año de antigüedad y además:

- Para los *jugadores*: si el promedio de goles por partido es superior a 0,5 se adiciona un plus de otro sueldo básico.
- Para los *entrenadores*: se adiciona un plus por campeonatos ganados (5000$ si ha ganado entre 1 y 4 campeonatos; $30.000 si ha ganado entre 5 y 10 campeonatos; 50.000$ si ha ganado más de 10 campeonatos).

D- Cualquier empleado debe responder al mensaje `toString`, que devuelve un String que lo representa, compuesto por nombre, sueldo a cobrar y efectividad.

F- Realizar un programa que instancie un jugador y un entrenador. Informe la representación String de cada uno.

NOTA: Reutilice el código de la clase Entrenador (definida en tema 3).

3-A- Implemente las clases para el siguiente problema. Una garita de seguridad quiere identificar los distintos tipos de personas que entran a un barrio cerrado. Al barrio pueden entrar: *personas*, que se caracterizan por nombre, DNI y edad; y *trabajadores*, estos son personas que se caracterizan además por la tarea realizada en el predio.

Implemente constructores, getters y setters para las clases. Además tanto las personas como los trabajadores deben responder al mensaje `toString` siguiendo el formato:

- Personas "Mi nombre es **Mauro,** mi DNI es **11203737** y tengo **70** años"
- Trabajadores "Mi nombre es **Mauro,** mi DNI es **11203737** y tengo **70** años. Soy **jardinero.**"

B- Realice un programa que instancie una persona y un trabajador y muestre la representación de cada uno en consola.

NOTA: Reutilice la clase Persona (carpeta tema2).

4- Un objeto *visor de figuras* se encarga de mostrar en consola cualquier figura que reciba y también mantiene cuántas figuras mostró. Analice y ejecute el siguiente programa y responda: ¿Qué imprime? ¿Por qué?

```
public class VisorFiguras {
  private int mostradas;

  public VisorFiguras(){
    mostradas=0;
  }

  public void mostrar(Figura f){
    System.out.println(f.toString());
    mostradas++;
  }

  public int getMostradas() {
    return mostradas;
  }
}
```

```
public class MainVisorFiguras {
 public static void main(String[] args) {
    VisorFiguras visor = new VisorFiguras();

    Cuadrado c1 = new Cuadrado(10,"Violeta","Rosa");
    Rectangulo r= new Rectangulo(20,10,"Azul","Celeste");
    Cuadrado c2= new Cuadrado(30,"Rojo","Naranja");

    visor.mostrar(c1);
    visor.mostrar(r);
    visor.mostrar(c2);

    System.out.println(visor.getMostradas());
 }
}
```

5-A- Modifique la clase VisorFiguras: ahora debe permitir guardar las figuras a mostrar (a lo sumo 5) y también mostrar todas las figuras guardadas. Use la siguiente estructura.

```
public class VisorFigurasModificado {
    private int guardadas;
    private int capacidadMaxima=5;
    private Figura [] vector;

    public VisorFigurasModificado(){
        //completar
    }

    public void guardar(Figura f){
        //completar
    }

    //sigue a la derecha ->
```

```
    public boolean quedaEspacio(){
        //completar
    }

    public void mostrar(){
        //completar
    }

    public int getGuardadas() {
        return guardadas;
    }

}
```

B- Realice un programa que instancie el visor, guarde dos cuadrados y un rectángulo en el visor y por último haga que el visor muestre sus figuras almacenadas.