Práctica 1

Primera Parte: 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- Escriba un programa que imprima en consola el factorial de un número N (ingresado por teclado, N > 0). Ejemplo: para N=5 debería imprimir **5! = 120**

2- Escriba un programa que imprima en consola el factorial de todos los números entre 1 y 9. ¿Qué modificación debe hacer para imprimir el factorial de los números impares solamente?

3- 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 Ej03Jugadores.java

4- 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 Ej04Matrices.java

5- 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.

6- Escriba un programa que simule el ingreso de personas a un banco. Cada persona que ingresa indica la operación que desea realizar (0: "cobro de cheque" 1: "depósito/ extracción en cuenta" 2: "pago de impuestos o servicios" 3: "cobro de jubilación" 4: "cobro de planes"). La recepción de personas culmina cuando un empleado ingresa la operación 5 (cierre del banco). Informar la cantidad de personas atendidas por cada operación y la operación más solicitada.

Segunda Parte: Introducción a POO.

Objetivo. Realizar programas donde se 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 una clase Persona (ya implementada 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.

Piense y responda: ¿Qué datos conforman el estado del objeto persona? ¿Cómo se implementan dichos datos? ¿Qué ocurre cuando se le envía un mensaje al objeto?

2- Utilizando la clase Persona (ya implementada). Realice un programa que almacene en un vector 15 personas. La información de cada persona debe leerse de teclado. Luego de almacenar la información:

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

3- Indique qué imprimen los siguientes programas. Responda: ¿Qué efecto tiene la *asignación* utilizada con objetos? ¿Qué se puede concluir acerca de 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- 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- 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. Muestre el mensaje escondido en consola.

NOTA: La primera letra de un string se obtiene enviándole el mensaje charAt(0) al objeto string. Probar con: humo oso lejos ala menos usado nene de ocho ! Debería imprimir: holamundo!

6- Se dispone de la clase Partido (ya implementada 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" pasado por parámetro (X)
setVisitante(X)	modifica el nombre del equipo visitante al "String" pasado por parámetro (X)
setGolesLocal(X)	modifica la cantidad de goles del equipo local "int" pasado por parámetro (X)
setGolesVisitante(X)	modifica la cantidad de goles del equipo visitante "int" pasado por parámetro (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 2

Primera parte: Desarrollo de Clases

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

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

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

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(): inicializa el monto y cantidad de ítems de la compra actual.
- registrarProducto(*pesoEnKg*, *precioPorKg*): recibe el peso en kg del ítem comprado y su precio por kg, debiendo realizar las actualizaciones en el estado de la balanza.
- devolverMontoAPagar(): retorna el monto de la compra actual.
- devolverResumenDeCompra(): 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 - Genere un programa principal que cree una balanza e inicie una compra. Lea información desde teclado correspondiente a los ítems comprados (peso en kg y precio por kg) hasta que se ingresa uno con 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 de fútbol. Un *entrenador* se caracteriza por su nombre, sueldo básico y la cantidad de campeonatos ganados.

- Defina métodos para obtener/modificar el valor de cada atributo.
- Defina el método *calcularSueldoACobrar* que calcula y devuelve el sueldo a cobrar por el *entrenador*. El sueldo se compone del sueldo básico, al cual se le 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 principal que instancie un *entrenador,* cargándole datos leídos desde teclado. Pruebe el correcto funcionamiento de cada método implementado.

4-A- Generar 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 (get# y set#)
- Calcular el área y devolverla. (método *calcularArea*)
- Calcular el perímetro y devolverlo. (método *calcularPerimetro*)

NOTA: la constante PI es *Math.PI*

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

5-A- Modifique el ejercicio 2-A. Ahora la balanza debe poder generar un resumen de compra más completo. Para eso agregue a la balanza la característica *resumen* (String). Modifique los métodos:

- iniciarCompra para que además inicie el *resumen* en el String vacío.
- registrarProducto para que reciba un objeto Producto (que se caracteriza por peso en kg y descripción) y su precio por kg. La operación debe realizar las actualizaciones en monto /cantidad de ítems y adicionar al *resumen* (string) la descripción y el monto pagado por este producto.
- devolverResumenDeCompra() para que retorne un String del siguiente estilo: "Naranja 100 pesos – Banana 40 pesos – Lechuga 50 pesos – Total a pagar 190 pesos por la compra de 3 productos" . La sección subrayada es el contenido de *resumen*.

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

NOTA: dispone en la carpeta tema 3 de la clase Producto ya implementada. Para adicionar la información del producto recibido al *resumen* use concatenación de Strings (operación +).

Segunda parte: Desarrollo de Clases (continuación)

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

Nota: Las modificaciones planteadas en los ejercicios 1 y 2 deben realizarse sobre los archivos .java de las clases ya definidas en la carpeta "tema 3". Para el resto de los ejercicios, trabajar sobre la carpeta "tema4" del proyecto.

1-A– i) Defina constructores para la clase Triángulo (definida anteriormente): el *primer constructor* debe recibir un valor para cada lado y para los colores de línea y relleno; el *segundo constructor* no debe poseer parámetros ni código (constructor nulo).

ii) Realice un programa que instancie un triángulo mediante los distintos constructores.

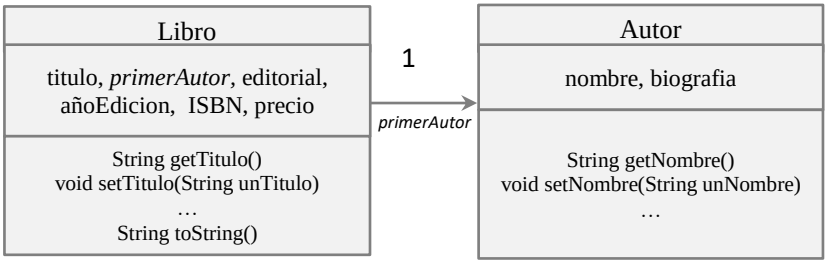
1-B- i) Defina constructores para la clase Círculo (definida anteriormente): el *primer constructor* debe recibir un valor para el radio y para los colores de línea y relleno; el *segundo constructor* no debe poseer parámetros ni código (constructor nulo).

ii) Realice un programa que instancie un círculo mediante los distintos constructores.

2 – i) Defina un constructor para la clase Entrenador (definida anteriormente) que reciba los datos necesarios (nombre, sueldo básico, cantidad de campeonatos ganados). Además defina un constructor nulo.

ii) Realice un programa que instancie un entrenador mediante el primer constructor.

3-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, considerando que éstos se caracterizan por nombre y biografía. El autor debe poder devolver/modificar el valor de sus atributos.



B- Modifique el programa ppal. (carpeta tema 4) para instanciar un libro con su autor, considerando las modificaciones realizadas en A). Los datos se ingresan por teclado.

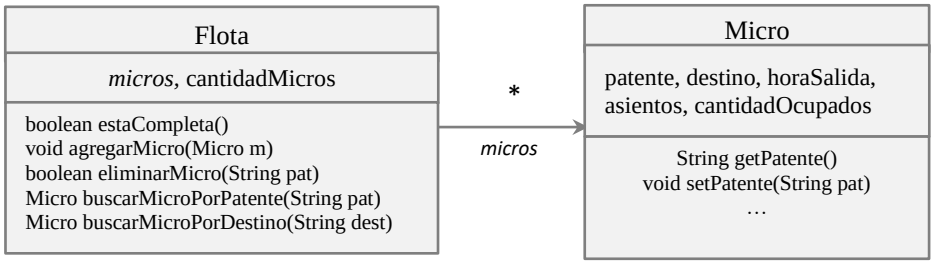
4-A- Definir una clase para representar micros. Un micro conoce su patente, destino, hora salida, el estado de sus 20 asientos (es decir si está o no ocupado) y la cantidad de asientos ocupados al momento. Lea detenidamente a) y b) y luego implemente.

a) Implemente un constructor que permita iniciar el micro con una patente, un destino y una hora de salida (recibidas por parámetro) y sin pasajeros.
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 número de asiento recibido como parámetro (es decir, devolver si está en rango o no)
   v. devolver el estado de un nro. de asiento válido recibido como parámetro
   vi. ocupar un nro. de asiento válido recibido como parámetro
   vii. liberar un nro. de asiento válido recibido como parámetro
   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. Cargue pasajeros al micro de la siguiente manera. Leer nros. de asientos desde teclado que corresponden a pedidos de personas. La lectura finaliza cuando se ingresa el nro. de asiento -1 o cuando se llenó el micro. Para cada nro. de asiento leído debe: validar el nro; en caso que esté libre, ocuparlo e informar a la persona el éxito de la operación; en caso que esté ocupado informar a la persona la situación y 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 se caracteriza por conocer a los micros que la componen (a lo sumo 15). Defina 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 recibido como parámetro.
   iii. eliminar de la flota el micro con patente igual a una recibida como parámetro, y retornar si la operación fue exitosa.

iv. buscar en la flota un micro con patente igual a una recibida como parámetro y retornarlo (en caso de no existir dicho micro, retornar null).
v. buscar en la flota un micro con destino igual a uno recibido como parámetro y retornarlo (en caso de no existir dicho micro, retornar null).



B- Genere un programa que cree una flota vacía. Cargue micros (sin pasajeros) a la flota con información leída desde teclado (hasta que se ingresa la patente "ZZZ000" o hasta completar la flota). Luego lea una patente y elimine de la flota el micro con esa patente; busque el micro con dicha patente para comprobar que ya no está en la flota. Para finalizar, lea un destino e informe la patente del micro que va a dicho destino.

Práctica 3

Herencia (utilizando Java)

Objetivo. Trabajar con el concepto de herencia y polimorfismo (utilizando Java).

NOTA: Trabajar sobre la carpeta "tema5" del proyecto

1 – A- Agregar la clase Triángulo a la jerarquía de figuras vista en clase (paquete tema5 del proyecto). 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, agregar la clase Círculo a la jerarquía de figuras.

C- Escriba un programa que instancie un triángulo, un círculo y un cuadrado, con información leída desde teclado. Luego muestre en consola el área y perímetro de cada uno y su representación en String.

2- Queremos representar la información de empleados de un club: jugadores y entrenadores.

- Cualquier *empleado* se caracteriza por su nombre y sueldo básico.
- 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, con los atributos de cada clase y métodos para obtener/modificar el valor de los mismos.

B- Implemente *constructores* para los jugadores y entrenadores, que reciban toda la información necesaria para inicializar el objeto en cuestión.

C- Cualquier empleado (jugador / entrenador) debe saber responder al mensaje c*alcularSueldoACobrar* (que calcula y devuelve el sueldo a cobrar) pero de manera diferente:

- Para los *jugadores*: el sueldo a cobrar es el sueldo básico y si el promedio de goles por partido es superior a 0,5 se adiciona un plus de otro sueldo básico.
- Para los *entrenadores*: el sueldo a cobrar es el sueldo básico al cual se le 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. La representación de cualquier empleado está compuesta por su nombre y sueldo a cobrar.

E- Escriba un programa principal que instancie un *jugador* y un *entrenador* con datos leídos desde teclado. Pruebe el correcto funcionamiento de cada método implementado.

NOTA: Tomar como base la clase Entrenador definida con anterioridad.

3- A- Modele e 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 su nombre, DNI y edad. Además pueden entrar trabajadores, estos son personas que se caracterizan además por la tarea que realizan en el predio.

Implemente constructores, getters y setters para las clases. Además tanto las personas como los trabajadores deben responder al mensaje toString(). A continuación se ejemplifica la representación a retornar por cada uno:

- 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 **Corta césped.**"

B- Genere un programa que instancie una persona y un trabajador con datos leídos de teclado y muestre la representación de cada uno en consola.

NOTA: reutilice la clase Persona (tema 2).

4- Dada la siguiente jerarquía, indique qué imprime el programa.

```
public class ClaseA {

   public int dos(){
      return 2;
   }

   public int tres(){
      return this.dos() + this.siete();
   }

   public int siete(){
      return 9;
   }
}
```

```
public class ClaseB extends ClaseA{

   public int dos(){
      return 5;
   }

   public int cuatro(){
      return this.dos() + super.tres();
   }

    public int seis(){
      return this.dos();
   }
}
```

```
public class ClaseC extends ClaseB{

   public int uno(){
      return this.cuatro();
   }

   public int dos(){
      return 9;
   }

   public int cinco(){
      return super.seis();
   }
}
```

```
public class QueImprime {
   public static void main(String[] args) {
      ClaseC objC=new ClaseC();
      System.out.println(objC.cinco());
      System.out.println(objC.uno());
   }
}
```

5- 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());
   }
}
```

6- Modificar la clase Visor Figuras: ahora debe permitir guardar las figuras a mostrar (a lo sumo 5) y también mostrar todas las figuras guardadas en forma conjunta. Usar la siguiente estructura.

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

   public VisorFigurasModificado(){
      //completar
   }

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

   public boolean quedaEspacio(){
      //completar
   }

   public void mostrar(){
      //completar
   }

   public int getGuardadas() {
      return guardadas;
   }

}
```

Luego realice un programa que instancie un visor, guarde dos cuadrados y un rectángulo en el visor y por último haga que el visor muestre sus figuras.

Práctica de Repaso

Objetivo. Repasar los temas de POO vistos en el módulo. Repasar el uso de Netbeans.

Nota: Para cada ejercicio cree un nuevo proyecto y cargue el paquete de lectura. Puede encontrar las instrucciones en la guía de uso rápida de Netbeans disponible en Medioteca.

1- La UNLP desea administrar sus proyectos, investigadores y subsidios. Un proyecto tiene: nombre, código, nombre y apellido del director y los investigadores que participan en el proyecto (50 como máximo). De cada investigador se tiene: nombre y apellido, categoría (1 a 5) y su especialidad. Además, cualquier investigador puede pedir hasta un máximo de 5 subsidios. De cada subsidio se conoce: el monto pedido, el motivo y si fue otorgado o no.

i) Implemente el modelo de clases teniendo en cuenta:
   a. Un proyecto sólo debería poder construirse con el nombre y el código.
   b. Un investigador sólo debería poder construirse con nombre y apellido, categoría y especialidad.
   c. Un subsidio sólo debería poder construirse con el monto solicitado y el motivo. Un subsidio siempre se crea en estado no-otorgado.

ii) Implemente los métodos necesarios (en las clases donde corresponda) que permitan:
   a. void agregarInvestigador(unInvestigador);
      // agrega un investigador a un proyecto.
   b. void agregarSubsidio(unSubsidio);
      // agrega un subsidio a un investigador.
   c. double dineroTotalOtorgado();
      //devuelve la cantidad de dinero total de todos los subsidios otorgados a todos los investigadores de un proyecto.
   d. int cantidadDeSubsidios(String nombre_y_apellido);
      // devuelve la cantidad de subsidios (otorgados o no) solicitados por el investigador llamado "nombre_y_apellido".
   e. void otorgarTodos(String nombre_y_apellido);
      // otorga todos los subsidios pendientes que tiene el investigador llamado “nombre_y_apellido”
   f. String toString();
      // Devuelve un string que tiene el nombre del proyecto, su código, el nombre y apellido del director, el total de dinero otorgado y el nombre y apellido de cada investigador. Para cada investigador, además, se debe agregar la categoría del mismo y el dinero de sus subsidios otorgados.

iii) Escriba un programa que instancie un proyecto con un director y dos investigadores. Asigne dos subsidios a cada investigador y otorgue los subsidios del primero de ellos, luego imprima todos los datos del proyecto en pantalla.

2- Un manager de bandas de música desea un sistema para manejar los recitales que organiza. El manager maneja dos tipos de recitales: eventos ocasionales y giras.

- Todo recital se caracteriza por el nombre de la banda y la lista de temas que tocarán durante el recital.
- Un evento ocasional es un recital que además tiene el motivo (a beneficio, show de TV o show privado), el nombre de la persona que contrata el recital y el día del evento.
- Una gira es el mismo recital que se repite varias veces en distintas ciudades. Una gira tiene un nombre, y las "fechas", de cada fecha se conoce la ciudad y el día. Además una gira deberá guardar la información de la fecha "*actual*" a tocar.

a) Realice el modelo de clases. Implemente las clases con sus atributos y métodos para obtener/modificar el valor de los mismos.

b) Implemente los constructores en todas las clases. El constructor de recitales recibe el nombre de la banda y la cantidad de temas que tendrá el recital. El constructor de eventos ocasionales además recibe el motivo, la persona que lo contrata y día del evento. El constructor de giras además recibe el nombre de la gira y la cantidad de fechas que tendrá la gira.

c) Incorpore los métodos listados a continuación:

i. Cualquier recital debe saber responder a los mensajes:

- **agregarTema** que recibe el nombre de un tema y lo almacena adecuadamente.
- **actuar** que imprime por consola y para cada tema, la leyenda "y ahora tocaremos…" seguido por el nombre del tema.

ii. La gira debe saber responder a los mensajes:

- **agregarFecha** que recibe la fecha (con ciudad y día) y la almacena adecuadamente.
- La gira debe responder al mensaje **actuar** de manera distinta. Imprime la leyenda "Buenas noches …" seguido del nombre de la ciudad "actual". Luego debe imprimir el listado de temas como lo hace cualquier recital. Además debe setearse correctamente el siguiente recital de la gira como el "actual".

iii. El evento ocasional debe saber responder al mensaje actuar de manera distinta:

- Si es un show de beneficencia se imprime la leyenda "Recuerden colaborar con…" seguido del nombre de quien contrató el evento.
- Si es un show de TV se imprime "Saludos amigos televidentes"
- Si es un show privado se imprime "Un feliz cumpleaños para…" seguido del nombre de quien contrató el evento.

Independientemente del motivo del evento, luego se imprime el listado de temas como lo hace cualquier recital.

iv. Todo recital debe saber responder a los mensajes:

- **finalizado:** Si es un evento ocasional devuelve true si el recital ya se llevó a cabo o false en caso contrario. Si es una gira devuelve true si ya se actuaron todas las fechas o false en caso contrario.

- **calcularCosto:** Si es un evento ocasional devuelve 0 si es a beneficio, 50000 si es un show de TV y 150000 si es privado. Las giras deben devolver 30000 por cada fecha de la misma.

d) Realice un programa que instancie un evento ocasional. Lea desde el teclado el día del evento, el nombre de quien lo contrata, el motivo, la banda y el listado de temas a tocar. Luego imprima el costo del evento e invoque el mensaje actuar del evento.

A continuación instancie una gira, leyendo desde el teclado el nombre de la gira, la banda, el listado de temas y el listado de ciudades con sus correspondientes días. Luego imprima el costo de la gira e invoque al mensaje actuar de la misma hasta finalizar la gira.

3- Una escuela de artes musicales arma coros con algunos de sus alumnos para participar de ciertos eventos. Un coro tiene un director y alumnos como coristas. De los coristas se conoce el nombre, el dni, la edad y el tono fundamental (un número entero). Del director se conoce el nombre, el dni, la edad y la antigüedad (un número entero). Los coros poseen un nombre y están formados por un director y una serie de coristas. Asimismo pueden formarse de dos formas: o bien los coristas se colocan en el escenario uno al lado del otro formando un *semicírculo*, o bien conforman *hileras* de la misma cantidad de coristas.

a. Implemente el modelo de clases teniendo en cuenta que los coros deberían crearse con un director y sin ningún corista, pero sí sabiendo cuantos coristas va a tener el coro.

b. Implemente métodos (en las clases donde corresponda) que permitan:
   - agregar un corista a un coro.
     - En el coro *semicircular* los coristas se deben ir agregando de izquierda a derecha
     - En el coro *por hileras* los coristas se deben ir agregando de izquierda a derecha completando la hilera antes de pasar a la siguiente, comenzando por la hilera de adelante.
   - determinar si un coro está lleno o no. Devuelve true si el coro tiene a todos sus coristas asignados o false en caso contrario.
   - determinar si un coro (se supone que está lleno) está bien formado. Un coro está bien formado si:
     - En el caso del coro *semicircular*, de izquierda a derecha los coristas están ordenados de mayor a menor en cuanto a tono fundamental.
     - En el caso del coro *por hileras*, desde adelante hacia atrás los coristas están ordenados de mayor a menor en cuanto a tono fundamental y todos los miembros de una misma hilera tienen el mismo tono fundamental.
   - devolver la representación de un coro formada por el nombre del coro, todos los datos del director y todos los datos de todos los coristas.

c. Escriba un programa que instancie cuatro coros y que los vaya almacenando en un arreglo. El tipo de cada uno de los cuatro coros es leído por teclado. Una vez leído el tipo de coro se deberá leer o bien la cantidad de coristas (en el caso del coro *semicircular*) o la cantidad de hileras e integrantes por hilera (en el caso del coro por *hileras*). Luego se deberá crear la cantidad de coristas necesarios, leyendo sus datos, y almacenándolos en el coro. Finalmente imprima toda la información de los cuatro coros indicando si están bien formados o no.