

Trabajo Práctico N° 6: **Módulo Objetos (Introducción a Java. Matrices).**

Ejercicio 1.

Analizar el programa Ej01Tabla2.java, que carga un vector que representa la tabla del 2. Luego, escribir las instrucciones necesarias para generar enteros aleatorios hasta obtener el número 11. Para cada número, mostrar el resultado de multiplicarlo por 2 (accediendo al vector).

Ejercicio 2.

Escribir un programa que lea las alturas de los 15 jugadores de un equipo de básquet y las almacene en un vector. Luego, informar:

- *La altura promedio.*
- *La cantidad de jugadores con altura por encima del promedio.*

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

Ejercicio 3.

Escribir un programa que defina una matriz de enteros de tamaño 5x5. Inicializar la matriz con números aleatorios entre 0 y 30. Luego, realizar las siguientes operaciones:

- *Mostrar el contenido de la matriz en consola.*
- *Calcular e informar la suma de los elementos de la fila 1.*
- *Generar un vector de 5 posiciones donde cada posición j contiene la suma de los elementos de la columna j de la matriz. Luego, imprimir el vector.*
- *Leer un valor entero e indicar si se encuentra o no en la matriz. En caso de encontrarse, indicar su ubicación (fila y columna), en caso contrario imprimir “No se encontró el elemento”.*

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

Ejercicio 4.

Un edificio de oficinas está conformado por 8 pisos (1..8) y 4 oficinas por piso (1..4). Realizar un programa que permita informar la cantidad de personas que concurrieron a cada oficina de cada piso. Para esto, simular 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, informar lo pedido.

Ejercicio 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. Escribir 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.

Trabajo Práctico N° 7: **Módulo Objetos (Introducción a POO).**

Ejercicio 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. Ejemplo: “Mi nombre es Mauro, mi DNI es 11203737 y tengo 70 años”.*

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

Ejercicio 2.

Utilizando la clase Persona, realizar 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:

- *Informar la cantidad de personas mayores de 65 años.*
- *Mostrar la representación de la persona con menor DNI.*

Ejercicio 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.

(b) Una vez finalizada la inscripción, 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, usar el método equals.

Ejercicio 4.

Sobre un nuevo programa, modificar 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, informar la situación. La inscripción finaliza al llegar una persona con nombre “ZZZ” o al cubrirse los 40 cupos de casting.

(b) Una vez finalizada la inscripción, informar para cada día: la cantidad de inscriptos al casting ese día y el nombre de la persona a entrevistar en cada turno asignado.

Ejercicio 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;

Implementar 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.*

Trabajo Práctico N° 8: **Módulo Objetos (Desarrollo de Clases).**

Ejercicio 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). Proveer un constructor que reciba todos los datos necesarios para iniciar el objeto. Proveer métodos para:

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

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

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

Ejercicio 2.

(a) *Modificar la clase Libro.java (carpeta tema3) para, ahora, considerar que el primer autor es un objeto instancia de la clase Autor. Implementar la clase Autor, sabiendo que 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. Luego, realizar las modificaciones necesarias en la clase Libro.*

(b) *Modificar el programa Demo01Constructores (carpeta tema3) para instanciar los libros con su autor, considerando las modificaciones realizadas. Luego, a partir de uno de los libros instanciados, obtener e imprimir la representación del autor de ese libro.*

Ejercicio 3.

(a) Definir una clase para representar estantes. Un estante almacena, a lo sumo, 20 libros. Implementar un constructor que permita iniciar el estante sin libros. Proveer métodos para:

- Devolver la cantidad de libros que hay almacenados.
- Devolver si el estante está lleno.
- Agregar un libro al estante.
- Devolver el libro con un título particular que se recibe.

(b) Realizar un programa que instancie un estante. Cargar varios libros. A partir del estante, buscar e informar el autor del libro “Mujercitas”.

(c) ¿Qué se modificaría en la clase definida para, ahora, permitir estantes que almacenen como máximo N libros? ¿Cómo se instanciaría el estante?

Ejercicio 4.

(a) Un hotel posee N habitaciones. De cada habitación, se conoce costo por noche, si está ocupada y, en caso de estarlo, guarda el cliente que la reservó (nombre, DNI y edad).

(i) Generar las clases necesarias. Para cada una, proveer métodos getters/setters adecuados.

(ii) Implementar los constructores necesarios para iniciar: los clientes a partir de nombre, DNI, edad; el hotel para N habitaciones, cada una desocupada y con costo aleatorio entre 2000 y 8000.

(iii) Implementar, en las clases que corresponda, todos los métodos necesarios para:

- Ingresar un cliente C en la habitación número X . Asumir que X es válido (es decir, está en el rango $1..N$) y que la habitación está libre.
- Aumentar el precio de todas las habitaciones en un monto recibido.
- Obtener la representación String del hotel, siguiendo el formato:
{Habitación 1: costo, libre u ocupada, información del cliente si está ocupada},
...
{Habitación N : costo, libre u ocupada, información del cliente si está ocupada}.

(b) Realizar un programa que instancie un hotel, ingrese clientes en distintas habitaciones, muestre el hotel, aumente el precio de las habitaciones y vuelva a mostrar el hotel.

NOTAS: Reusar la clase Persona. Para cada método solicitado, pensar a qué clase debe delegar la responsabilidad de la operación.

Ejercicio 5.

(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*). Proveer un constructor que reciba todos los datos necesarios para iniciar el objeto. Proveer métodos para:

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

(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*.

Trabajo Práctico N° 9: **Módulo Objetos (Herencia).**

Ejercicio 1.

(a) Incluir la clase Triángulo a la jerarquía de figuras vista (carpeta tema4). Triángulo debe heredar de Figura todo lo que es común y definir su constructor y sus atributos y métodos propios. Además, se debe redefinir el método toString.

(b) De igual manera, incluir la clase Círculo a la jerarquía de figuras.

(c) Añadir, a la representación String, el valor del perímetro. Pensar: ¿qué método toString se debe modificar: el de cada subclase o el de Figura?

(d) Añadir el método despintar que establece los colores de la figura a línea “negra” y relleno “blanco”. Pensar: ¿dónde se debe definir el método: en cada subclase o en Figura?

(e) Realizar un programa que instancie un triángulo y un círculo. Mostrar en consola la representación String de cada uno. Probar el funcionamiento del método despintar.

Ejercicio 2.

Se quiere 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) Implementar 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 cada 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 (\$5.000 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. Informar la representación String de cada uno.

NOTA: Para cada método a implementar, pensar en qué clase/s se debe definir el método.

Ejercicio 3.

(a) Implementar 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. Implementar 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) Realizar un programa que instancie una persona y un trabajador y mostrar la representación de cada uno en consola.

NOTA: Reutilizar la clase Persona (carpeta tema2).

Ejercicio 4.

Un objeto visor de figuras se encarga de mostrar en consola cualquier figura que reciba y también mantiene cuántas figuras mostró. Analizar y ejecutar el siguiente programa y responder: ¿Qué imprime? ¿Por qué?

Ejercicio 5.

(a) Modificar la clase VisorFiguras: ahora, debe permitir guardar las figuras a mostrar (a lo sumo, 5) y también mostrar todas las figuras guardadas. Usar la siguiente estructura.

(b) Realizar 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.

Ejercicio 6.

El Servicio Meteorológico Nacional necesita un sistema que permita registrar, para una determinada estación meteorológica, la temperatura promedio mensual de N años consecutivos a partir de un año A dado. Además, necesita dos versiones del sistema: una que permita reportar el promedio histórico por años y otra que permita reportar el promedio histórico por meses. Esto se detalla más adelante. De la estación, interesa conocer: nombre y latitud y longitud donde se encuentra. Implementar las clases, constructores y métodos que se considere necesarios para:

(a) *Crear el sistema de registro/reporte, que funcionará en una determinada estación, para N años consecutivos a partir de un año A. Iniciar cada temperatura en un valor muy alto.*

(b) *Registrar la temperatura de un mes y año recibidos por parámetro. Nota: El mes está en rango 1..12 y el año está en rango A..A+N-1.*

(c) *Obtener la temperatura de un mes y año recibidos por parámetro. Nota: El mes está en rango 1..12 y el año está en rango A..A+N-1. En caso de no haberse registrado temperatura para ese mes/año, se retorna el valor muy alto.*

(d) *Devolver un String que concatena el mes y año en que se registró la mayor temperatura. Nota: Suponer que ya están registradas las temperaturas de todos los meses y años.*

(e) *Devolver un String con el nombre de la estación, su latitud y longitud y los promedios mensuales o anuales según corresponda:*

- *La versión del sistema que reporta por años deberá calcular el promedio para cada año (el promedio del año X se calcula con los datos mensuales de ese año).*
- *La versión del sistema que reporta por meses deberá calcular el promedio para cada mes (el promedio del mes M se calcula con los datos de todos los años en ese mes).*

Nota: Suponer que ya están registradas las temperaturas de todos los meses y años. Utilizar el carácter \n para concatenar un salto de línea.

(f) *Realizar un programa principal que cree un Sistema con reporte anual para 3 años consecutivos a partir del 2021, para la estación La Plata (latitud -34.921 y longitud -57.955). Cargar todas las temperaturas (para todos los meses y años). Informar los promedios anuales y el mes y año en que se registró la mayor temperatura. Luego, crear un Sistema con informe mensual para 4 años a partir de 2020, para la estación Mar del Plata (latitud -38.002 y longitud -57.556). Cargar todas las temperaturas (para todos los meses y años). Informar los promedios mensuales y el mes y año en que se registró la mayor temperatura.*

NOTA: *Prestar atención de no violar el encapsulamiento al resolver el ejercicio.*

Trabajo Práctico N° 10: **Módulo Objetos (Repaso).**

Ejercicio 1.

La UNLP desea administrar sus proyectos, investigadores y subsidios. Un proyecto tiene: nombre, código, nombre completo del director y los investigadores que participan en el proyecto (50 como máximo). De cada investigador, se tiene: nombre completo, categoría (1 a 5) y 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.

(a) *Implementar el modelo de clases teniendo en cuenta:*

- *Un proyecto sólo debería poder construirse con el nombre, código, nombre del director.*
- *Un investigador sólo debería poder construirse con nombre, categoría y especialidad.*
- *Un subsidio sólo debería poder construirse con el monto pedido y el motivo. Un subsidio siempre se crea en estado no-otorgado.*

(b) *Implementar los métodos necesarios (en las clases donde corresponda) que permitan:*

- *void agregarInvestigador(Investigador unInvestigador);
// agregar un investigador al proyecto.*
- *void agregarSubsidio(Subsidio unSubsidio);
// agregar un subsidio al investigador.*
- *double dineroTotalOtorgado();
// devolver el monto total otorgado en subsidios del proyecto (tener en cuenta todos los subsidios otorgados de todos los investigadores).*
- *void otorgarTodos(String nombre_completo);
// otorgar todos los subsidios no-otorgados del investigador llamado nombre_completo.*
- *String toString();
// devolver un string con: nombre del proyecto, código, nombre del director, el total de dinero otorgado del proyecto y la siguiente información de cada investigador: nombre, categoría, especialidad y el total de dinero de sus subsidios otorgados.*

(c) *Escribir un programa que instancie un proyecto con tres investigadores. Agregar dos subsidios a cada investigador y otorgar los subsidios de uno de ellos. Luego, imprimir todos los datos del proyecto en pantalla.*

Ejercicio 2.

Se quiere un sistema para gestionar estacionamientos. Un estacionamiento conoce su nombre, dirección, hora de apertura, hora de cierre y almacena, para cada número de piso (1..N) y número de plaza (1..M), el auto que ocupa dicho lugar. De los autos, se conoce nombre del dueño y patente.

(a) *Generar las clases, incluyendo getters y setters adecuados.*

(b) *Implementar constructores. En particular, para el estacionamiento:*

- *Un constructor debe recibir nombre y dirección e iniciar el estacionamiento con hora de apertura "8:00", hora de cierre "21:00" y para 5 pisos y 10 plazas por piso. El estacionamiento inicialmente no tiene autos.*
- *Otro constructor debe recibir nombre, dirección, hora de apertura, hora de cierre, el número de pisos (N) y el número de plazas por piso (M) e iniciar el estacionamiento con los datos recibidos y sin autos.*

(c) *Implementar métodos para:*

- *Dado un auto A, un número de piso X y un número de plaza Y, registrar al auto en el estacionamiento en el lugar (X, Y). Suponer que (X, Y) son válidos (es decir, están en rango 1..N y 1..M, respectivamente) y que el lugar está desocupado.*
- *Dada una patente, obtener un String que contenga el número de piso y plaza donde está dicho auto en el estacionamiento. En caso de no encontrarse, retornar el mensaje "Auto Inexistente".*
- *Obtener un String con la representación del estacionamiento.
Ejemplo: "Piso 1 Plaza 1: libre; Piso 1 Plaza 2: representación del auto; ...; Piso 2 Plaza 1: libre; ..."*
- *Dado un número de plaza Y, obtener la cantidad de autos ubicados en dicha plaza (teniendo en cuenta todos los pisos).*

(d) *Realizar un programa que instancie un estacionamiento con 3 pisos y 3 plazas por piso. Registrar 6 autos en el estacionamiento en distintos lugares. Mostrar la representación String del estacionamiento en consola. Mostrar la cantidad de autos ubicados en la plaza 1. Leer una patente por teclado e informar si dicho auto se encuentra en el estacionamiento o no. En caso de encontrarse, la información a imprimir es el piso y plaza que ocupa.*

Ejercicio 3.

Un productor musical desea administrar los recitales que organiza, que pueden ser: eventos ocasionales y giras.

- *De todo recital, se conoce 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 del contratante del recital y el día del evento.*
- *Una gira es un recital que, además, tiene un nombre y las “fechas” donde se repetirá la actuación. De cada “fecha”, se conoce la ciudad y el día. Además, la gira guarda el número de la fecha en la que se tocará próximamente (actual).*

(a) Generar las clases necesarias. Implementar métodos getters/setters adecuados.

(b) Implementar los constructores. 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, el nombre del contratante y día del evento. El constructor de giras, además, recibe el nombre de la gira y la cantidad de fechas que tendrá.

(c) Implementar los métodos listados a continuación:

(i) Cualquier recital debe saber responder a los mensajes:

- ***agregarTema** que recibe el nombre de un tema y lo agrega a la lista de temas.*
- ***actuar** que imprime (por consola), 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 una “fecha” y la agrega adecuadamente.*
- *La gira debe responder al mensaje **actuar** de manera distinta. Imprimir la leyenda “Buenas noches...” seguido del nombre de la ciudad de la fecha “actual”. Luego, debe imprimir el listado de temas como lo hace cualquier recital. Además, debe establecer la siguiente fecha de la gira como la nueva “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 del contratante.*
- *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 del contratante.*

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

(iv) *Todo recital debe saber responder al mensaje **calcularCosto** teniendo en cuenta lo siguiente. Si es un evento ocasional, devuelve 0 si es a beneficio, 50.000 si es un show de TV y 150.000 si es privado. Las giras deben devolver 30.000 por cada fecha de la misma.*

(d) *Realizar un programa que instancie un evento ocasional y una gira, cargando la información necesaria. Luego, para ambos, imprimir el costo e invocar al mensaje “actuar”.*

Ejercicio 4.

Una escuela de música arma coros para participar de ciertos eventos. Los coros poseen un nombre y están formados por un director y una serie de coristas. Del director, se conoce el nombre, DNI, edad y la antigüedad (un número entero). De los coristas, se conoce el nombre, DNI, edad y el tono fundamental (un número entero). Asimismo, hay dos tipos de coros: coro semicircular en el que los coristas se colocan en el escenario uno al lado del otro y coro por hileras donde los coristas se organizan en filas de igual dimensión.

(a) *Implementar las clases necesarias teniendo en cuenta que los coros deberían crearse con un director y sin ningún corista, pero sí sabiendo las dimensiones del coro.*

(b) *Implementar métodos (en las clases donde corresponda) que permitan:*

- *agregar un corista al 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.*
- *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, todos los miembros de una misma hilera tienen el mismo tono fundamental y, además, todos los primeros miembros de cada hilera están ordenados de mayor a menor en cuanto a 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) *Escribir un programa que instancie un coro de cada tipo. 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, crear la cantidad de coristas necesarios, leyendo sus datos y almacenándolos en el coro. Finalmente, imprimir toda la información de los coros indicando si están bien formados o no.*