Módulo Imperativo
Práctica Inicial

1.– Implementar un programa que procese la información de los alumnos de la Facultad de Informática.
a) Implementar un módulo que lea y retorne, en una estructura adecuada, la información de todos los alumnos. De cada alumno se lee su apellido, número de alumno, año de ingreso, cantidad de materias aprobadas (a lo sumo 36) y nota obtenida (sin contar los aplazos) en cada una de las materias aprobadas. La lectura finaliza cuando se ingresa el número de alumno 11111, el cual debe procesarse.
b) Implementar un módulo que reciba la estructura generada en el inciso a) y retorne número de alumno y promedio de cada alumno.
c) Analizar: ¿qué cambios requieren los puntos a y b, si no se sabe de antemano la cantidad de materias aprobadas de cada alumno, y si además se desean registrar los aplazos? ¿cómo puede diseñarse una solución modularizada que requiera la menor cantidad de cambios?

2.– Implementar un programa que procese información de propiedades que están a la venta en una inmobiliaria.
Se pide:
a) Implementar un módulo para almacenar en una estructura adecuada, las propiedades agrupadas por zona. Las propiedades de una misma zona deben quedar almacenadas ordenadas por tipo de propiedad. Para cada propiedad debe almacenarse el código, el tipo de propiedad y el precio total. De cada propiedad se lee: zona (1 a 5), código de propiedad, tipo de propiedad, cantidad de metros cuadrados y precio del metro cuadrado. La lectura finaliza cuando se ingresa el precio del metro cuadrado -1.
b) Implementar un módulo que reciba la estructura generada en a), un número de zona y un tipo de propiedad y retorne los códigos de las propiedades de la zona recibida y del tipo recibido.

3.– Implementar un programa que procese las ventas de un supermercado. El supermercado dispone de una tabla con los precios y stocks de los 1000 productos que tiene a la venta.
a) Implementar un módulo que retorne, en una estructura de datos adecuada, los tickets de las ventas. De cada venta se lee código de venta y los productos vendidos. Las ventas finalizan con el código de venta -1. De cada producto se lee código y cantidad de unidades solicitadas. Para cada venta, la lectura de los productos a vender finaliza con cantidad de unidades vendidas igual a 0. El ticket debe contener:
- Código de venta
- Detalle (código de producto, cantidad y precio unitario) de los productos que se pudieron vender. En caso de no haber stock suficiente, se venderá la máxima cantidad posible.
- Monto total de la venta.

c) Implementar un módulo que reciba la estructura generada en el inciso a) y un código de producto y retorne la cantidad de unidades vendidas de ese código de producto.

Módulo Imperativo
Práctica Ordenación

1.- Descargar el programa **ImperativoEjercicioClase1.pas** que contiene parte del código del siguiente enunciado a resolver. Implementar lo indicado en el código.

Se desea procesar la información de las ventas de productos de un comercio (como máximo 50).

Implementar un programa que invoque los siguientes módulos:

a. Un módulo que retorne la información de las ventas en un vector. De cada venta se conoce el día de la venta, código del producto (entre 1 y 15) y cantidad vendida (como máximo 99 unidades). El código debe generarse automáticamente (random) y la cantidad se debe leer. El ingreso de las ventas finaliza con el día de venta 0 (no se procesa).

b. Un módulo que muestre el contenido del vector resultante del punto a).

c. Un módulo que ordene el vector de ventas por código.

d. Un módulo que muestre el contenido del vector resultante del punto c).

e. Un módulo que elimine, del vector ordenado, las ventas con código de producto entre dos valores que se ingresan como parámetros.

f. Un módulo que muestre el contenido del vector resultante del punto e).

g. Un módulo que retorne la información (ordenada por código de producto de menor a mayor) de cada código par de producto junto a la cantidad total de productos vendidos.

h. Un módulo que muestre la información obtenida en el punto g).

2.- El administrador de un edificio de oficinas cuenta, en papel, con la información del pago de las expensas de dichas oficinas.

Implementar un programa que invoque a módulos para cada uno de los siguientes puntos:

a. Genere un vector, sin orden, con a lo sumo las 300 oficinas que administra. De cada oficina se ingresa el código de identificación, DNI del propietario y valor de la expensa. La lectura finaliza cuando se ingresa el código de identificación -1, el cual no se procesa.

b. Ordene el vector, aplicando el método de inserción, por código de identificación de la oficina.

c. Ordene el vector aplicando el método de selección, por código de identificación de la oficina.

3.- Netflix ha publicado la lista de películas que estarán disponibles durante el mes de diciembre de 2022. De cada película se conoce: código de película, código de género (1: acción, 2: aventura, 3: drama, 4: suspenso, 5: comedia, 6: bélico, 7: documental y 8: terror) y puntaje promedio otorgado por las críticas.

Implementar un programa que invoque a módulos para cada uno de los siguientes puntos:

a. Lea los datos de películas, los almacene por orden de llegada y agrupados por código de género, y retorne en una estructura de datos adecuada. La lectura finaliza cuando se lee el código de la película -1.

b. Genere y retorne en un vector, para cada género, el código de película con mayor puntaje obtenido entre todas las críticas, a partir de la estructura generada en a)..

c. Ordene los elementos del vector generado en b) por puntaje utilizando alguno de los dos métodos vistos en la teoría.

d. Muestre el código de película con mayor puntaje y el código de película con menor puntaje, del vector obtenido en el punto c).

4.- Una librería requiere el procesamiento de la información de sus productos. De cada producto se conoce el código del producto, código de rubro (del 1 al 8) y precio.

Implementar un programa que invoque a módulos para cada uno de los siguientes puntos:

a. Lea los datos de los productos y los almacene ordenados por código de producto y agrupados por rubro, en una estructura de datos adecuada. El ingreso de los productos finaliza cuando se lee el precio 0.

b. Una vez almacenados, muestre los códigos de los productos pertenecientes a cada rubro.

c. Genere un vector (de a lo sumo 30 elementos) con los productos del rubro 3. Considerar que puede haber más o menos de 30 productos del rubro 3. Si la cantidad de productos del rubro 3 es mayor a 30, almacenar los primeros 30 que están en la lista e ignore el resto.

d. Ordene, por precio, los elementos del vector generado en c) utilizando alguno de los dos métodos vistos en la teoría.

e. Muestre los precios del vector resultante del punto d).

f. Calcule el promedio de los precios del vector resultante del punto d).

Módulo Imperativo
Práctica Recursión

1.- Descargar el programa **ImperativoEjercicioClase2.pas** que contiene parte del código del siguiente enunciado a resolver. Implementar lo indicado en el código.

Implementar un programa que invoque a los siguientes módulos.

a. Un módulo recursivo que retorne un vector de a lo sumo 15 números enteros “random” mayores a 10 y menores a 155 (incluidos ambos). La carga finaliza con el valor 20.

b. Un módulo no recursivo que reciba el vector generado en a) e imprima el contenido del vector.

c. Un módulo recursivo que reciba el vector generado en a) e imprima el contenido del vector.

d. Un módulo recursivo que reciba el vector generado en a) y devuelva la suma de los valores pares contenidos en el vector.

e. Un módulo recursivo que reciba el vector generado en a) y devuelva el máximo valor del vector.

f. Un módulo recursivo que reciba el vector generado en a) y un valor y devuelva verdadero si dicho valor se encuentra en el vector o falso en caso contrario.

g. Un módulo que reciba el vector generado en a) e imprima, para cada número contenido en el vector, sus dígitos en el orden en que aparecen en el número. Debe implementarse un módulo recursivo que reciba el número e imprima lo pedido. Ejemplo si se lee el valor 142, se debe imprimir 1 4 2.

2.- Escribir un programa que:

a. Implemente un módulo recursivo que genere y retorne una lista de números enteros “random” en el rango 100-200. Finalizar con el número 100.

b. Un módulo recursivo que reciba la lista generada en a) e imprima los valores de la lista en el mismo orden que están almacenados.

c. Implemente un módulo recursivo que reciba la lista generada en a) e imprima los valores de la lista en orden inverso al que están almacenados.

d. Implemente un módulo recursivo que reciba la lista generada en a) y devuelva el mínimo valor de la lista.

e. Implemente un módulo recursivo que reciba la lista generada en a) y un valor y devuelva verdadero si dicho valor se encuentra en la lista o falso en caso contrario.

3.- Implementar un programa que invoque a los siguientes módulos.

a. Un módulo recursivo que retorne un vector de 20 números enteros “random” mayores a 300 y menores a 1550 (incluidos ambos).

b. Un módulo que reciba el vector generado en a) y lo retorne ordenado. (*Utilizar lo realizado en la práctica anterior)*

c. Un módulo que realice una búsqueda dicotómica en el vector, utilizando el siguiente encabezado:

Procedure busquedaDicotomica (v: vector; ini,fin: indice; dato:integer; var pos: indice);

Nota: El parámetro "pos" debe retornar la posición del dato o -1 si el dato no se encuentra en el vector.

Desafío…

4.- Realizar un programa que lea números y que utilice un módulo recursivo que escriba el equivalente en binario de un número decimal. El programa termina cuando el usuario ingresa el número 0 (cero).

Ayuda: Analizando las posibilidades encontramos que: Binario (N) es N si el valor es menor a 2. ¿Cómo obtenemos los dígitos que componen al número? ¿Cómo achicamos el número para la próxima llamada recursiva? Ejemplo: si se ingresa 23, el programa debe mostrar: 10111.

Módulo Imperativo
Práctica Árboles 1

1. Descargar el programa **ImperativoEjercicioClase3.pas** que contiene parte del código del siguiente enunciado a resolver. Implementar lo indicado en el código.

Escribir un programa que:

a. Implemente un módulo que almacene información de socios de un club en un árbol binario de búsqueda. De cada socio se debe almacenar número de socio, nombre y edad. La carga finaliza con el número de socio 0 y el árbol debe quedar ordenado por número de socio. La información de cada socio debe generarse aleatoriamente.

b. Una vez generado el árbol, realice módulos independientes que reciban el árbol como parámetro y que :

i. Informar los datos de los socios en orden creciente.

ii. Informar los datos de los socios en orden decreciente.

iii. Informar el número de socio con mayor edad. Debe invocar a un módulo recursivo que retorne dicho valor.

iv. Aumentar en 1 la edad de los socios con edad impar e informar la cantidad de socios que se les aumentó la edad.

vi. Leer un nombre e informar si existe o no existe un socio con ese nombre. Debe invocar a un módulo recursivo que reciba el nombre leído y retorne verdadero o falso.

vii. Informar la cantidad de socios. Debe invocar a un módulo recursivo que retorne dicha cantidad.

viii. Informar el promedio de edad de los socios. Debe invocar a un módulo recursivo que retorne el promedio de las edades de los socios.

2. Escribir un programa que:

a. Implemente un módulo que genere aleatoriamente información de ventas de un comercio. Para cada venta generar código de producto, fecha y cantidad de unidades vendidas. Finalizar con el código de producto 0. Un producto puede estar en más de una venta. Se pide:

i. Generar y retornar un árbol binario de búsqueda de ventas ordenado por código de producto. Los códigos repetidos van a la derecha.

ii. Generar y retornar otro árbol binario de búsqueda de productos vendidos ordenado por código de producto. Cada nodo del árbol debe contener el código de producto y la cantidad total de unidades vendidas.

iii. Generar y retornar otro árbol binario de búsqueda de productos vendidos ordenado por código de producto. Cada nodo del árbol debe contener el código de producto y la lista de las ventas realizadas del producto.

Nota: El módulo debe retornar TRES árboles.

b. Implemente un módulo que reciba el árbol generado en i. y una fecha y retorne la cantidad total de productos vendidos en la fecha recibida.

c. Implemente un módulo que reciba el árbol generado en ii. y retorne el código de producto con mayor cantidad total de unidades vendidas.

c. Implemente un módulo que reciba el árbol generado en iii. y retorne el código de producto con mayor cantidad de ventas.

3. Implementar un programa que contenga:

a. Un módulo que lea información de los finales rendidos por los alumnos de la Facultad de Informática y los almacene en una estructura de datos. La información que se lee es legajo, código de materia, fecha y nota. La lectura de los alumnos finaliza con legajo 0. La estructura generada debe ser eficiente para la búsqueda por número de legajo y para cada alumno deben guardarse los finales que rindió en una lista.

b. Un módulo que reciba la estructura generada en a. y retorne la cantidad de alumnos con legajo impar.

c. Un módulo que reciba la estructura generada en a. e informe, para cada alumno, su legajo y su cantidad de finales aprobados (nota mayor o igual a 4).

c. Un módulo que reciba la estructura generada en a. y un valor real. Este módulo debe retornar los legajos y promedios de los alumnos cuyo promedio supera el valor ingresado.

Módulo Imperativo
Práctica Árboles 2

1. Descargar el programa ImperativoEjercicioClase4.pas. y completar los módulos indicados en el código.

a. Almacenar los productos vendidos en una estructura eficiente para la búsqueda por código de producto. De cada producto deben quedar almacenados su código, la cantidad total de unidades vendidas y el monto total. De cada venta se cargan código de venta, código del producto vendido, cantidad de unidades vendidas y precio unitario. El ingreso de las ventas finaliza cuando se lee el código de venta 0.

b. Imprimir el contenido del árbol ordenado por código de producto.

c. Retornar el menor código de producto.

d. Retornar la cantidad de códigos que existen en el árbol que son menores que un valor que se recibe como parámetro.

e. Retornar el monto total entre todos los códigos de productos comprendidos entre dos valores recibidos (sin incluir) como parámetros.

2. Descargar el programa ImperativoEjercicioClase3.pas de la clase anterior e incorporar lo necesario para:

i. Informar el número de socio más grande. Debe invocar a un módulo recursivo que retorne dicho valor.

ii. Informar los datos del socio con el número de socio más chico. Debe invocar a un módulo recursivo que retorne dicho socio.

iii. Leer un valor entero e informar si existe o no existe un socio con ese valor. Debe invocar a un módulo recursivo que reciba el valor leído y retornar verdadero o falso.

iv. Leer e informar la cantidad de socios cuyos códigos se encuentran comprendidos entre los valores leídos. Debe invocar a un módulo recursivo que reciba los valores leídos y retorne la cantidad solicitada.

3. Implementar un programa modularizado para una librería. Implementar módulos para:

a. Almacenar los productos vendidos en una estructura eficiente para la búsqueda por código de producto. De cada producto deben quedar almacenados su código, la cantidad total de unidades vendidas y el monto total. De cada venta se lee código de venta, código del producto vendido, cantidad de unidades vendidas y precio unitario. El ingreso de las ventas finaliza cuando se lee el código de venta -1.

b. Imprimir el contenido del árbol ordenado por código de producto.

c. Retornar el código de producto con mayor cantidad de unidades vendidas.

d. Retornar la cantidad de códigos que existen en el árbol que son menores que un valor que se recibe como parámetro.

e. Retornar el monto total entre todos los códigos de productos comprendidos entre dos valores recibidos (sin incluir) como parámetros.

4. Una biblioteca nos ha encargado procesar la información de los préstamos realizados durante el año 2021. De cada préstamo se conoce el ISBN del libro, el número de socio, día y mes del préstamo y cantidad de días prestados. Implementar un programa con:

a. Un módulo que lea préstamos y retorne 2 estructuras de datos con la información de los préstamos. La lectura de los préstamos finaliza con ISBN 0. Las estructuras deben ser eficientes para buscar por ISBN.

   i. En una estructura cada préstamo debe estar en un nodo. Los ISBN repetidos insertarlos a la derecha.
   ii. En otra estructura, cada nodo debe contener todos los préstamos realizados al ISBN. (prestar atención sobre los datos que se almacenan).

b. Un módulo recursivo que reciba la estructura generada en i. y retorne el ISBN más grande.

c. Un módulo recursivo que reciba la estructura generada en ii. y retorne el ISBN más pequeño.

d. Un módulo recursivo que reciba la estructura generada en i. y un número de socio. El módulo debe retornar la cantidad de préstamos realizados a dicho socio.

e. Un módulo recursivo que reciba la estructura generada en ii. y un número de socio. El módulo debe retornar la cantidad de préstamos realizados a dicho socio.

f. Un módulo que reciba la estructura generada en i. y retorne una nueva estructura ordenada ISBN, donde cada ISBN aparezca una vez junto a la cantidad total de veces que se prestó.

g. Un módulo que reciba la estructura generada en ii. y retorne una nueva estructura ordenada ISBN, donde cada ISBN aparezca una vez junto a la cantidad total de veces que se prestó.

h. Un módulo recursivo que reciba la estructura generada en g. y muestre su contenido.

i. Un módulo recursivo que reciba la estructura generada en i. y dos valores de ISBN. El módulo debe retornar la cantidad total de préstamos realizados a los ISBN comprendidos entre los dos valores recibidos (incluidos).

j. Un módulo recursivo que reciba la estructura generada en ii. y dos valores de ISBN. El módulo debe retornar la cantidad total de préstamos realizados a los ISBN comprendidos entre los dos valores recibidos (incluidos).

Módulo Imperativo
Práctica Adicionales

1. El administrador de un edificio de oficinas tiene la información del pago de las expensas de dichas oficinas. Implementar un programa con:

a) Un módulo que retorne un vector, sin orden, con a lo sumo las 300 oficinas que administra. Se deben cargar, para cada oficina, el código de identificación, DNI del propietario y valor de la expensa. La lectura finaliza cuando llega el código de identificación 0.

b) Un módulo que reciba el vector retornado en a) y retorne dicho vector ordenado por código de identificación de la oficina. Ordenar el vector aplicando uno de los métodos vistos en la cursada.

c) Un módulo que realice una búsqueda dicotómica. Este módulo debe recibir el vector generado en b) y un código de identificación de oficina. En el caso de encontrarlo, debe retornar la posición del vector donde se encuentra y en caso contrario debe retornar 0. Luego el programa debe informar el DNI del propietario o un cartel indicando que no se encontró la oficina.

d) Un módulo recursivo que retorne el monto total de las expensas.

2. Una agencia dedicada a la venta de autos ha organizado su stock y, tiene la información de los autos en venta. Implementar un programa que:

a) Genere la información de los autos (patente, año de fabricación (2010..2018), marca y modelo, finalizando con marca 'MMM') y los almacene en dos estructuras de datos:

   i. Una estructura eficiente para la búsqueda por patente.

   ii. Una estructura eficiente para la búsqueda por marca. Para cada marca se deben almacenar todos juntos los autos pertenecientes a ella.

b) Invoque a un módulo que reciba la estructura generado en a) i y una marca y retorne la cantidad de autos de dicha marca que posee la agencia.

c) Invoque a un módulo que reciba la estructura generado en a) ii y una marca y retorne la cantidad de autos de dicha marca que posee la agencia.

d) Invoque a un módulo que reciba el árbol generado en a) i y retorne una estructura con la información de los autos agrupados por año de fabricación.

e) Invoque a un módulo que reciba el árbol generado en a) i y una patente y devuelva el modelo del auto con dicha patente.

f) Invoque a un módulo que reciba el árbol generado en a) ii y una patente y devuelva el modelo del auto con dicha patente.

3. Un supermercado requiere el procesamiento de sus productos. De cada producto se conoce código, rubro (1..10), stock y precio unitario. Se pide:

a) Generar una estructura adecuada que permita agrupar los productos por rubro. A su vez, para cada rubro, se requiere que la búsqueda de un producto por código sea lo más eficiente posible. El ingreso finaliza con el código de producto igual a 0.
b) Implementar un módulo que reciba la estructura generada en a), un rubro y un código de producto y retorne si dicho código existe o no para ese rubro.
c) Implementar un módulo que reciba la estructura generada en a), y retorne, para cada rubro, el código y stock del producto con mayor código.
d) Implementar un módulo que reciba la estructura generada en a), dos códigos y retorne, para cada rubro, la cantidad de productos con códigos entre los dos valores ingresados.

4. Una oficina requiere el procesamiento de los reclamos de las personas. De cada reclamo se ingresa código, DNI de la persona, año y tipo de reclamo. El ingreso finaliza con el código de igual a 0. Se pide:

a) Un módulo que retorne estructura adecuada para la búsqueda por DNI. Para cada DNI se deben tener almacenados cada reclamo y la cantidad total de reclamos que realizó.
b) Un módulo que reciba la estructura generada en a) y un DNI y retorne la cantidad de reclamos efectuados por ese DNI.
c) Un módulo que reciba la estructura generada en a) y dos DNI y retorne la cantidad de reclamos efectuados por todos los DNI comprendidos entre los dos DNI recibidos.
d) Un módulo que reciba la estructura generada en a) y un año y retorne los códigos de los reclamos realizados en el año recibido.

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- Analice el programa Ej01Tabla2.java, que carga un vector que representa la tabla del 2.
Genere 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 5x5. Inicialice la matriz con números aleatorios entre 0 y 30.
Luego realice 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, imprima el vector.
- Leer un valor entero e indicar 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 (1..8) y 4 oficinas por piso (1..4). 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- 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.

Práctica 3

Desarrollo de Clases

Objetivo. Definir clases para representar objetos del mundo real. Concepto de clase, estado (variables de instancia) y comportamiento (métodos). Constructores. Relacionar clases por asociación/conocimiento. Referencia this.

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`).

Provea un constructor que reciba todos los datos necesarios para iniciar el objeto.

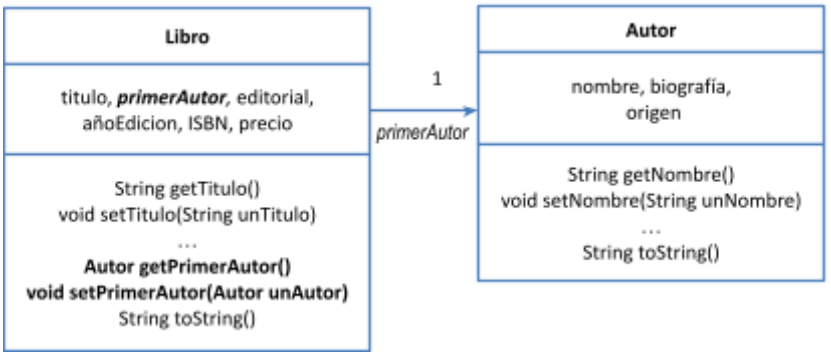
Provea métodos para:

- 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 triángulo, le cargue información leída desde teclado e informe en consola el perímetro y el área.

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(#)`

2-A- Modifique la clase Libro.java (carpeta tema3) para ahora considerar que el *primer autor* es un objeto instancia de la clase Autor.
Implemente 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 realice las modificaciones necesarias en la clase Libro.



B- Modifique 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, obtenga e imprima la representación del autor de ese libro.

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

(i) Genere las clases necesarias. Para cada una provea métodos getters/setters adecuados.

(ii) Implemente 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 e/ 2000 y 8000.

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

- Ingresar un cliente C en la habitación número X. Asuma 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- Realice 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: Reúse la clase Persona. Para cada método solicitado piense a qué clase debe delegar la responsabilidad de la operación.

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`).

Provea un constructor que reciba todos los datos necesarios para iniciar el objeto.

Provea 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`

Práctica 4

Herencia

Objetivo. Trabajar con el concepto de herencia y polimorfismo.

Nota: Trabajar sobre la carpeta "tema4" del proyecto

1- Nos piden una aplicación estilo Paint, para ello necesitamos representar figuras geométricas (cuadrados, rectángulos, círculos, triángulos). Todas las figuras tienen color de relleno y color de línea. Además, los triángulos guardan el valor de sus tres lados, los cuadrados el valor de su lado, los círculos el valor del radio, y los rectángulos el valor de la base y de la altura.

Las figuras deben incluir funcionalidad para: crearla a partir de los datos necesarios (constructor), modificar/obtener el valor de los atributos (métodos `get` y `set`), calcular el área y devolverla (método `calcularArea`), calcular el perimetro y devolverlo (método `calcularPerimetro`), y mostrar la representación String de la figura (método `toString`) concatenando toda la información.

A- Analice la jerarquía de figuras (carpeta tema4).

B- Incluya la clase Triángulo a la jerarquía de figuras. 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`.

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

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

E- 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?

F- 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 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 (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: para cada método a implementar piense en que clase/s debe definir el método.

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- 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 tenga funcionalidad para reportar el promedio histórico por años y otra que tenga funcionalidad para reportar el promedio histórico por meses. *Esto se detalla más adelante.*

De la estación, interesa conocer: nombre, y latitud y longitud donde se encuentra.

Implemente las clases, constructores y métodos que 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**. Inicie 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: Suponga 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).

Ej: *"La Plata (34,921 S - 57,955 O):*
- Año 2020: 23,8 ºC;
- Año 2021: 26,1 ºC;
- Año 2022: 25,3 ºC. "

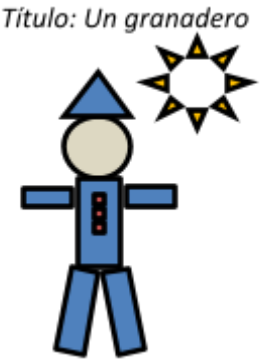
- La versión del sistema que reporta por meses deberá calcular el promedio *para cada mes* (el promedio del mes M se calcula con los datos de todos los años en ese mes).

Ej: *"La Plata (34,921 S - 57,955 O):*
- Enero: 28,2 ºC;
- Febrero: 26,8 ºC;
- Marzo: 24.3 ºC
- ….."

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

f) Realice 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). Cargue todas las temperaturas (para todos los meses y años). Informe los promedios anuales, y el mes y año en que se registró la mayor temperatura.
Luego cree 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). Cargue todas las temperaturas (para todos los meses y años). Informe los promedios mensuales, y el mes y año en que se registró la mayor temperatura.

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



5 A- Queremos representar dibujos. Un dibujo guarda su título y las figuras que lo componen (círculos, triángulos, cuadrados, rectángulos, etc). Piense, con lo visto hasta ahora (no es necesario implementar):

i- ¿Dónde almacenará las figuras que componen el dibujo?. ¿Cuántas estructuras se necesitarán?.
ii- ¿Cómo agregará las distintas figuras al dibujo?. ¿Cuántos métodos *agregar* necesita implementar?
iii- ¿Qué problema surge a medida que aumentan las posibles figuras en la jerarquía?

B- Implemente la clase Dibujo usando un ***array genérico de Figuras***. Dicho array puede guardar objetos creados a partir de cualquier subclase de Figura. Siga el molde mostrado.

```
public class Dibujo {
  private String titulo;
  private Figura [] vector;
  private int guardadas;
  private int capacidadMaxima=10;

  //inicia el dibujo, sin figuras
  public Dibujo (String titulo){
      //completar
  }

  //agrega la figura al dibujo
  public void agregar(Figura f){
      //completar
      System.out.println("la figura "+
                         f.toString() +
                    " se ha guardado");
  }

  //calcula el área del dibujo:
  //suma de las áreas de sus figuras
  public double calcularArea(){
      //completar
  }

  //sigue a la derecha ->
```

```
  //imprime el título, representación
  //de cada figura, y área del dibujo
  public void mostrar(){
      //completar
  }

  //retorna está lleno el dibujo
  public boolean estaLleno() {
    return (guardadas == capacidadMaxima);
  }
}
```

```
public class MainDibujos {
 public static void main(String[] args) {
  Dibujo d = new Dibujo("un dibujo");

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

  d.agregar (c1);
  d.agregar (r);
  d.agregar (c2);

  d.mostrar();
 }
}
```

B- Analice y ejecute programa `MainDibujos`. Responda: ¿Qué imprime? ¿Por qué?

Piense: ¿qué ventaja tiene esta implementación a medida que aumentan las posibles figuras en la jerarquía? ¿cuál es la ventaja del polimorfismo? ¿dónde se observa en este ejercicio?

Práctica de Repaso

Objetivo. Repasar los temas de POO vistos en el módulo. Repasar el uso de Netbeans: crear proyecto, cargar paquete de lectura y exportar/comprimir proyecto.

Nota: Resuelva cada ejercicio **en un nuevo proyecto** y si lo necesita **cargue el paquete de lectura**. Finalizado el desarrollo, **exporte/comprima su proyecto (a .zip).** Puede encontrar las instrucciones en la Guía de uso rápida de Netbeans.

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.

i) Implemente el modelo de clases teniendo en cuenta:
   a. Un proyecto sólo debería poder construirse con el nombre, código, nombre del director.
   b. Un investigador sólo debería poder construirse con nombre, categoría y especialidad.
   c. Un subsidio sólo debería poder construirse con el monto pedido 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(Investigador unInvestigador);
      // agregar un investigador al proyecto.
   b. void agregarSubsidio(Subsidio unSubsidio);
      // agregar un subsidio al investigador.
   c. double dineroTotalOtorgado();
      //devolver el monto total otorgado en subsidios del proyecto (tener en cuenta todos los subsidios *otorgados* de todos los investigadores)
   d. void otorgarTodos(String nombre_completo);
      //otorgar todos los subsidios no-otorgados del investigador llamado *nombre_completo*
   e. 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.

iii) Escriba un programa que instancie un proyecto con tres investigadores. Agregue dos subsidios a cada investigador y otorgue los subsidios de uno de ellos. Luego imprima todos los datos del proyecto en pantalla.

2- Queremos 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) Genere las clases, incluyendo getters y setters adecuados.

b) Implemente 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) Implemente 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. Suponga 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* … etc"
- Dado un número de plaza Y, obtener la cantidad de autos ubicados en dicha plaza (teniendo en cuenta todos los pisos).

d) Realice un programa que instancie un estacionamiento con 3 pisos y 3 plazas por piso.
Registre 6 autos en el estacionamiento en distintos lugares.
Muestre la representación String del estacionamiento en consola.
Muestre la cantidad de autos ubicados en la plaza 1.
Lea una patente por teclado e informe 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.

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) Genere las clases necesarias. Implemente métodos getters/setters adecuados.

b) Implemente 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) Implemente 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. Imprime 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, 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 y una gira, cargando la información necesaria. Luego, para ambos, imprima el costo e invoque al mensaje actuar.

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. Implemente 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. Implemente 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.

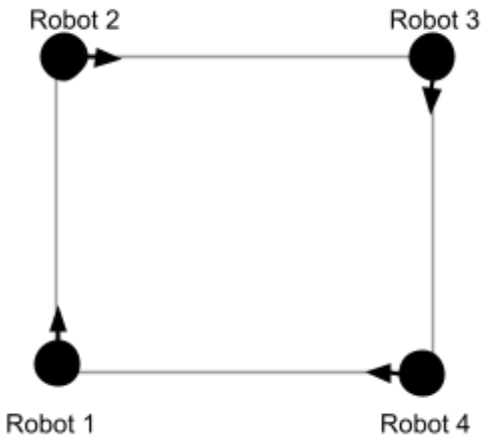
c. Escriba un programa que instancie un coro de cada tipo. Lea o bien la cantidad de coristas (en el caso del *coro semicircular*) o la cantidad de hileras e integrantes por hilera (en el caso del *coro por hileras*). Luego cree la cantidad de coristas necesarios, leyendo sus datos, y almacenándolos en el coro. Finalmente imprima toda la información de los coros indicando si están bien formados o no.

Práctica 1

Objetivo:
Realizar programas en R-info en los que múltiples robots realizan tareas. Diseñar soluciones con robots del mismo tipo y con robots de diferentes tipos. Analizar situaciones de posibles colisiones.

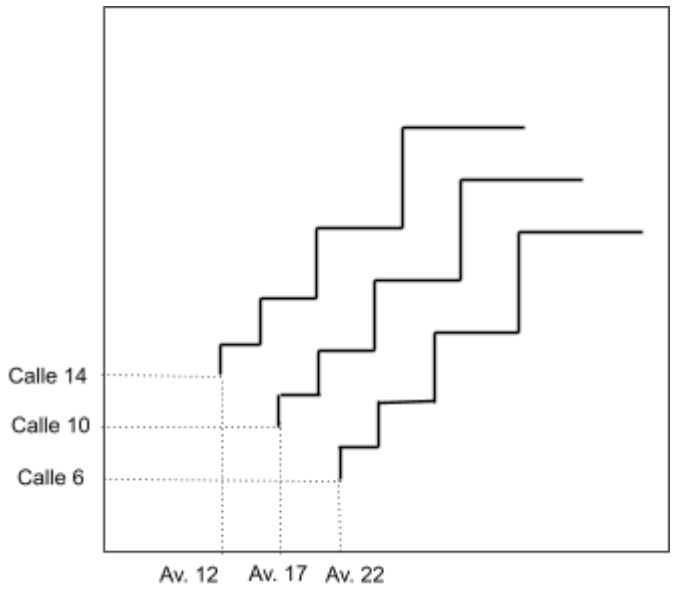
1) Realice un programa para que un robot junte todas las flores de la avenida 1 y las deposite al final de dicha avenida. Al finalizar, debe informar la cantidad de flores depositadas y la cantidad de esquinas sin flores que encontró durante el recorrido.
   a) Modifique el programa anterior para que el mismo robot realice lo mismo en las avenidas 1, 3 y 5.
   b) Modifique el programa anterior para que el trabajo sea realizado por 3 robots: uno realiza la avenida 1, otro realiza la avenida 3 y otro la avenida 5. Cada robot debe iniciar en las esquina (1,1), (3,1) y (5,1) respectivamente.

2) Realice un programa en el que 4 robots limpien de papeles el perímetro de un cuadrado de lado 20 en sentido horario, como se muestra en la figura:



El vértice inferior izquierdo del cuadrado se ubica en la esquina (10,10).
Al finalizar, cada robot debe informar la cantidad de papeles juntados en su lado.
Al realizar este programa, analizar:
   a) ¿Cómo deben declararse la o las áreas?
   b) ¿Existe riesgo de colisión?

3) Realice un programa en el que 3 robots realicen escaleras de 4 escalones. El tamaño de cada escalón se incrementa en un 1 respecto al escalón anterior. El primer escalón será de 1x1, el segundo de 2x2, y así sucesivamente, como se muestra a continuación:



Al finalizar el recorrido, cada robot debe informar la cantidad de escalones en los que la cantidad de papeles superó en 1 a la cantidad de flores. Las esquinas deben quedar sin modificar.

4) Realice un programa en el que dos robots se encargan de limpiar las ciudad. La ciudad se dividió en 4 áreas: las impares (1 y 3) deben limpiarse de flores; y las pares (2 y 4) deben limpiarse de papeles. Un robot debe encargarse de las áreas impares y otro robot de las pares. Modularice el recorrido de cada área

- Área 1: desde la avenida 1 hasta la avenida 25
- Área 2: desde la avenida 26 hasta la avenida 50
- Área 3: desde la avenida 51 hasta la avenida 75
- Área 4: desde la avenida 76 hasta la avenida 100

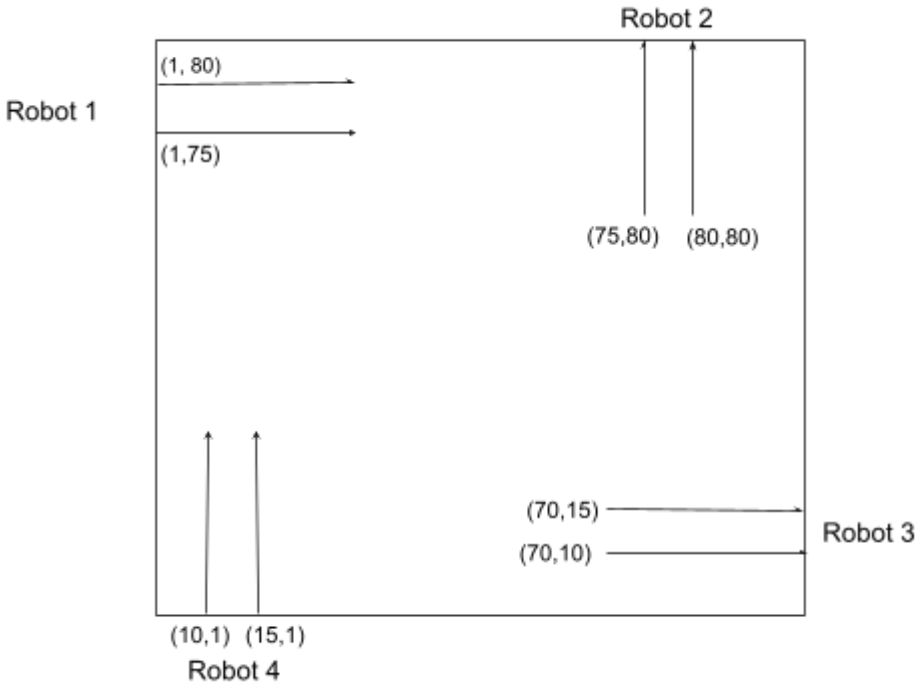
a) Analizar (no es necesario implementar) qué se debería modificar si ahora se pide que la ciudad se divida en 20 áreas:

- Área 1: Avenidas 1 a 5
- Área 2: Avenidas 6 a 10
- …

- Área 19: Avenidas 91 a 95
- Área 20: Avenidas 96 a 100

5) Realice un programa en el que cuatro robots realizan las siguientes actividades:

- El robot 1 debe limpiar de flores las primeras 15 esquinas de las calles 75 y 80. Al finalizar cada calle, debe depositar todas las flores en la última esquina.
- El robot 2 debe limpiar de papeles las últimas 20 esquinas de las avenidas 75 y 80. Al finalizar cada avenida debe depositar todos los papeles en la primera esquina.
- El robot 3 debe limpiar de flores las últimas 30 esquinas de las calles 10 y 15. Al finalizar cada calle, debe depositar todas las flores en la última esquina.
- El robot 4 debe limpiar de papeles las primeras 10 esquinas de las avenidas 10 y 15. Al finalizar cada avenida debe depositar todos los papeles en la primera esquina.



Práctica 2 - Programación Concurrente

Objetivo:
Realizar programas en R-info con distintos tipos de robots. Utilizar el pasaje de mensajes para la comunicación entre robots. Usar la función random para generar valores aleatorios.

1. Dos robots compiten para ver cuál junta más flores. El primer robot recoge todas las flores de la avenida 1 entre las calles 1 y 10. El segundo robot recoge todas las flores de la avenida 2, entre las calles 11 y 20.
Al finalizar el recorrido, el robot que recogió mayor cantidad de flores debe informar la diferencia de flores que obtuvo respecto al robot perdedor (el que obtuvo menos flores).
Los robots inician en las esquinas (1, 1) y (2, 11) respectivamente.

b. Modifique el ejercicio anterior, considerando que ahora habrá un robot fiscalizador, que será responsable de informar la diferencia de flores que obtuvo el ganador con respecto al perdedor. El robot fiscalizador se ubica en la esquina (2,1)

c. Modifique el ejercicio anterior para que ahora participen 6 robots

- Robot 1: Avenida 1, entre las calles 1 y 10
- Robot 2: Avenida 2, entre las calles 11 y 20
- Robot 3: Avenida 3, entre las calles 21 y 30
- Robot 4: Avenida 4, entre las calles 31 y 40
- Robot 5: Avenida 5, entre las calles 41 y 50
- Robot 6: Avenida 6, entre las calles 51 y 60
- Fiscalizador: Avenida 2, calle 1

El fiscalizador deberá informar la cantidad de flores que juntó el robot ganador.

d. Modifique el inciso anterior para que ahora el fiscalizador informe también, cuál fue el robot ganador.

e. Analizar (no es necesario implementar): ¿cómo se puede implementar el inciso 1.c. sin robot fiscalizador?
→ ¿qué cantidad de robots participarán del juego?
→ ¿qué cantidad de mensajes deben enviarse?

2. Realice un programa en el que 3 robots realizan una escalera de 4 escalones cada uno. Todos los escalones tienen un ancho fijo de 1, y un alto aleatorio entre 1 y 5. Al finalizar el recorrido, cada robot deberá enviar al robot jefe la cantidad de escalones que tenían más flores que papeles. Una vez que los tres robots finalizaron, el robot jefe deberá informar la suma de las cantidades enviadas por los 3 robots.

- El robot jefe inicia en la esquina (1,1)
- El robot 1 inicia en la esquina (2,1)
- El robot 2 inicia en la esquina (7,1)
- El robot 3 inicia en la esquina (12,1)

3. Realice un programa con 2 equipos:
- El equipo A, compuesto por los robots A1 y A2, debe juntar papeles de las primeras 20 esquinas de la calle 1
- El equipo B, compuesto por los robots B1 y B2, debe juntar flores de las primeras 20 esquinas de la calle 5

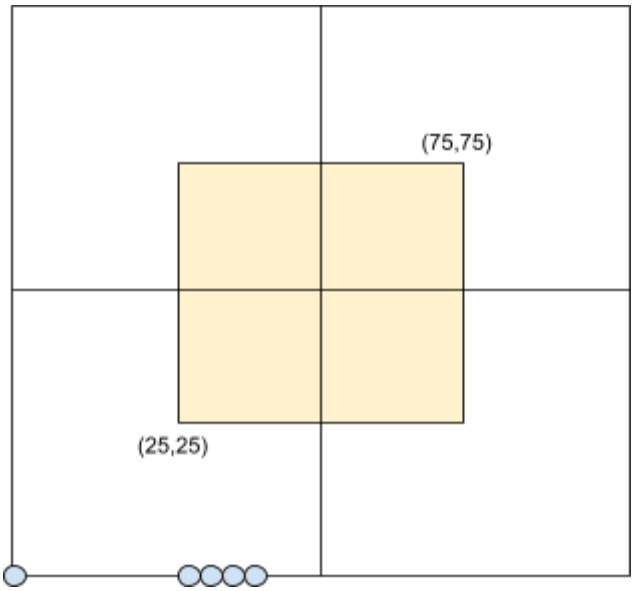
Los robots A1 y B1 deberán realizar las 10 primeras esquinas de su recorrido, y al finalizar avisarán a los robots A2 y B2 respectivamente para que continúen con las siguientes 10 esquinas. El segundo robot de cada equipo debe informar la cantidad de elementos recogidos en las 20 esquinas.

Inicialice los 4 robots en las esquinas que considere más apropiadas según el trayecto que le corresponde realizar a cada uno.

b. Modifique el programa anterior para que cada equipo repita el recorrido con las siguientes 20 esquinas de sus correspondientes calles.

c. Analice (no es necesario implementar) cómo implementaría el inciso **b** si ahora cada equipo debe realizar 8 segmentos de 20 esquinas.

4. Realice un programa en el que un robot fiscalizador controla el acceso de 4 robots recolectores al cuadrante encerrado entre las esquinas (25,25) y (75,75) . Para ello, el robot fiscalizador avisa a un robot recolector aleatorio que puede ingresar al área. El robot que recibe la autorización de acceso calcula una esquina aleatoria dentro del área, limpia dicha esquina de flores y papeles, regresa a su esquina y avisa al robot fiscalizador que ya finalizó.



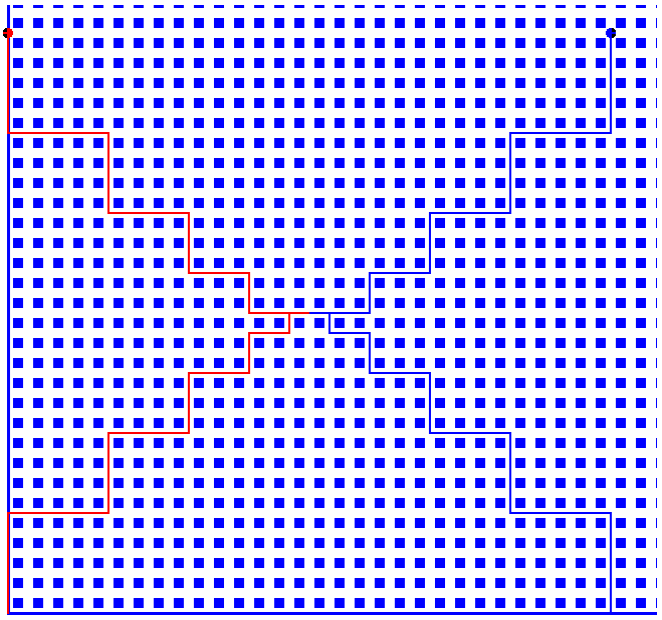
Se realizarán en total 10 accesos al cuadrante entre los 4 robots recolectores. Al finalizar, el robot fiscalizador deberá indicar al robot ganador que se posicione en la esquina (50,50).
El robot fiscalizador inicia en la esquina (1,1) y los robots recolectores inician en las esquinas (25,1) (30,1) (35,1) y (40,1) respectivamente.

Práctica 3 - Programación Concurrente

Objetivo:
Realizar programas en R-info con distintos tipos de robots. Utilizar memoria compartida para la comunicación y sincronización entre robots. Combinar problemas con memoria compartida y pasaje de mensajes.

1- Realice un programa con 2 robots recolectores de flores (floreros) y 2 robots recolectores de papeles (papeleros).
Los floreros comparten área y tienen 5 intentos cada uno para juntar las flores de una esquina, dentro de dicha área, elegida al azar en cada intento. Del mismo modo, los papeleros comparten área y tienen 3 intentos cada uno para juntar los papeles. En cada intento cada robot va a la esquina al azar, junta todos los elementos (flores o papeles según le corresponda) y vuelve a su esquina original. Al finalizar sus intentos cada robot debe acceder a la esquina (10, 10) y depositar los elementos recogidos de a uno.
- Área de floreros: (1,1) a (5, 10)
- Área de papeleros: (6, 1) a (10, 9)
- Esquinas de inicio de floreros: (6,10) y (7,10)
- Esquinas de inicio de papeleros: (8,10) y (9,10)

2- Realice un programa en el cual 2 robots corren una carrera. El recorrido realizado por cada uno es el que se muestra en la siguiente figura. Durante el recorrido el robot 1 debe juntar todas las flores que encuentre en los vértices de cada escalón, y el robot 2 debe juntar todos los papeles que encuentre en los vértices de cada escalón. Al finalizar deben informar la cantidad de elementos recogidos.
El robot 1 debe iniciar su recorrido en la esquina (1,1) y el robot 2 en la esquina (31,1).
Al finalizar la carrera, un robot jefe (inicializado en la esquina (15,1)) debe informar qué robot llegó primero a la esquina central de su recorrido.



3- Realice un programa donde 4 robots colaboren para recoger todas las flores de una esquina indicada por un robot jefe, seleccionada de manera aleatoria dentro del cuadrante (2,2) y (10,10). Para ello el jefe determina inicialmente una esquina y los robots deben accederla, tomar **de a una** las flores y volver a su posición inicial. Cuando los robots terminan el jefe deberá informar cuál de ellos logró recoger más flores.
Las esquinas de inicio de los robots deberán ser jefe (1,1) y robots (2,1), (3, 1), (4,1) y (5,1).

4- Realice un programa en el que 4 robots mueven **de a una** todas las flores de la esquina (10,10) a la esquina (11,11). Para ello, cada robot que toma una flor de la esquina (10,10) la deposita en la esquina (11,11) y luego retorna a su esquina inicial. Cada robot que finaliza (o, sea, que detecta que la esquina (10,10) se ha vaciado) deberá avisar al robot coordinador que ha finalizado. Cuando todos los robots finalizaron, el robot coordinador deberá informar qué robot finalizó último y a continuación deberá recolectar todas las flores de la esquina (11,11).

El robot coordinador inicia en la esquina (1,1).
Los robots inician en las esquinas (9,9) (9,10) (9,11) y (9,12).

b- Implemente una variante en la cual los robots luego de tomar cada flor de la esquina (10,10) vuelvan a su esquina inicial, pasen a la esquina (11,11) a depositarla y finalmente vuelvan a la esquina inicial.

c- Analizar:
- ¿Cuál de las 2 soluciones maximiza la concurrencia?
- ¿Se podría resolver este problema sin que los robots deban regresar a su esquina inicial?

5- Realice un programa en el que 4 robots juegan una carrera por avenidas diferentes: 4, 6, 8 y 10 respectivamente. Todos los robots inician en la calle 1.
Para poder avanzar, cada robot debe juntar un papel de una fuente de papeles localizada en la esquina (11,11), colocarlo en la esquina actual de su avenida y avanzar un paso. Cuando la esquina fuente ya no tiene más papeles, o cuando se haya completado la avenida, deberán avisar al robot coordinador y este determinará el robot que llegó más lejos.

6.a- Tres robots deben recorrer el perímetro de su cuadrante, como se indica a continuación:
- El robot 1 comienza la esquina (2,2) y debe realizar un cuadrante de 6x6 juntando todas las flores que encuentre
- El robot 2 comienza en la esquina (5,5) y debe realizar un cuadrante de 10x10 juntando todas las flores y los papeles que encuentre
- El robot 3 comienza en la esquina (9,9) y debe realizar un cuadrante de 7x7 juntando todos los papeles que encuentre

Cada robot que finalice su cuadrante deberá avisar al robot fiscalizador. Al recibir el aviso, el robot fiscalizador indicará inmediatamente una calle a la que deberá dirigirse el robot recolector, considerando que el robot que finalizó primero irá a la calle 20, el segundo a la 21 y el tercero a la 22.
Cuando los robots recolectores reciben un número de calle, deberán posicionarse en la avenida 1 de dicha calle, y avanzar a lo largo de la calle depositando en cada esquina un papel, una flor o ambos, según lo que cada robot haya juntado. El recorrido finalizará al completar la calle o vaciarse las bolsas.

6.b Analizar (no es necesario implementar): ¿cómo debería modificarse el ejercicio anterior si los robots recolectores no conocen de antemano el tamaño de su cuadrante (por ejemplo, porque lo calcula el fiscalizador de manera aleatoria)?

6.c. Modifique el ejercicio anterior (**6.a**) para que ahora el robot fiscalizador espere a que todos los robots recolectores hayan completado sus cuadrantes antes de indicarles la calle que deberán recorrer.

Práctica 4 Concurrente

Objetivo:
Realizar programas en R-info con distintos tipos de robots. Utilizar memoria compartida para la comunicación y sincronización entre robots. Combinar problemas con memoria compartida y pasaje de mensajes. Distinguir modelos de algoritmos a desarrollar de acuerdo al problema planteado.

1- Clientes y Servidores
Existe un robot que sirve de flores a tres robots clientes. Cada cliente solicita al servidor que le deposite en su esquina siguiente una cantidad de flores aleatoria (entre 1 y 4). Por ejemplo, si el cliente se encuentra en la esquina (2,1) le solicitará que coloque x cantidad de flores en la esquina (2,2).
Cuando el robot servidor deposita las flores en la esquina solicitada, el cliente las junta y las deposita una a una a lo largo de la avenida en la que se encuentra.
El programa finaliza cuando todos los robos clientes completan su avenida. Asuma que el robot servidor tiene flores suficientes en su bolsa.

El robot servidor se inicia en la esquina (100,100)
Los robots clientes inician en las esquinas (1,1) , (2,1) y (3,1) respectivamente

Protocolo Cliente/Servidor

Cliente:	**Servidor**

INICIO: calcularRandom flores
Enviar ID al servidor
Enviar cantFlores al servidor
Enviar mi Avenida actua
Enviar Calle siguiente
Esperar ACK del servidor
Ir a la esquina Avenida,Calle
JuntarFlores
Volver a la esquina
Avanzar dejando flores
Si llegué a la avenida 100
  enviar 0 al servidor
sino
  Volver a *INICIO*
	INICIO: Recibir ID
	Recibir N Flores de ID
	si (flores <> 0)
	  recibir avenida de ID
	  recibir calle de ID
	  pos(avenida,calle)
	  depositar N flores
	  volver a (100,100)
	  enviar ACK a robot ID
	  volver a *INICIO*
	sino
	  contar un robot terminado
	  si terminaron los 3 robots
	  terminar

2. Productores y consumidores

Existen dos robots productores que recorren las avenidas 5 y 10 respectivamente, juntando todos los papeles de su avenida. A lo largo del recorrido, cada vez que juntan 5 papeles, los depositan en la esquina (50,50).

Además existen dos robots consumidores que intentan tomar una cantidad aleatoria de papeles (entre 2 y 5) de la esquina (50,50) para depositarla en su esquina de origen. Si la esquina (50,50) no posee la cantidad de papeles requerida, vuelven a su esquina de origen sin tomar ningún papel. Si luego de 8 intentos **seguidos** un consumidor detecta que la esquina (50,50) no tiene papeles suficientes para juntar, entonces asumirá que los productores ya han completado su trabajo y por lo tanto terminará su tarea también.

Los consumidores inician en las esquinas (11,1) y (12,1) respectivamente.

3. Sincronización barrera

Tres robots deben vaciar de papeles su avenida, comenzando por la calle 1 y terminando en la calle 100. El trabajo lo deben realizar todos juntos y en etapas: los robots inician juntos y cuando todos completan una etapa del trabajo pueden avanzar a la siguiente, lo que significa que para poder pasar de etapa los robots deben esperar que todos hayan completado la etapa en curso. Se proponen dos posibles soluciones a este problema: etapas homogéneas o etapas heterogéneas:

a) Implemente el programa considerando que cada robot completa una etapa cada 5 esquinas
b) Implemente el programa considerando que cada robot completa una etapa luego de juntar N papeles. El valor de N (entre 1 y 5) lo calcula cada robot antes de iniciar cada etapa.

En cada solución, analice cómo debería finalizar el programa.
Los robots inician en las esquinas (1,1), (2,1) y (3,1) respectivamente. Existe un robot coordinador, cuya única tarea es asignar identificadores a cada robot.

4. Jefe y trabajadores - Master/Slave

Un robot jefe asigna tareas a los trabajadores. Las tareas consisten en 1. recoger flores, 2. recoger papeles, 3. vaciar bolsa, 4. finalizar .
Existen 2 robots trabajadores que reciben solicitudes de tareas del robot jefe. Para cada solicitud, reciben la tarea y la esquina donde deben realizarla (salvo cuando la tarea es 4 que no deben acceder a una esquina). Luego de recibir la tarea, los robots van a la esquina indicada, realizan la tarea, avisan al jefe que ya la completaron y quedan a la espera de una nueva tarea.

El robot jefe inicia en la esquina (1,1) y los robots trabajadores inician en las esquinas (2,1) y (3,1). Las tareas se asignan aleatoriamente a cualquier esquina dentro del cuadrante comprendido entre las esquinas (2,2) y (100,100). El robot jefe envía 10 tareas aleatorias (entre 1 y 3) a trabajadores aleatorios y termina. Al finalizar el jefe envía la tarea 4.

Analice: existe el riesgo de que el programa quede bloqueado, y que ningún robot trabajador pueda realizar su tarea. ¿en qué caso puede suceder esto? ¿qué resulta necesario considerar para evitar esta situación?

Práctica 5 - Programación Concurrente

Objetivo: Repaso

1- Se organizó una competencia entre el equipo rojo y el equipo azul. Cada equipo consta de dos robots, y debe realizar una tarea:

- Los robots R1 y R2 del equipo rojo debe juntar todas las flores de las avenidas 2 y 3 respectivamente
- Los robots A1 y A2 del equipo azul debe juntar todos los papeles de las calles 98 y 99 respectivamente

Al finalizar la competencia, un robot fiscalizador deberá informar el equipo que juntó más objetos.

2- Tres robots recolectores deben avanzar por su calle vaciando las esquinas. El avance debe realizarse en conjunto en etapas, siguiendo el modelo de sincronización barrera, en el cual los robots deben esperar que todos terminen su tarea antes de avanzar a la siguiente etapa. Cada etapa consiste en recorrer 10 esquinas y luego depositar todas las flores recolectadas en la esquina (50,50). Una vez que los robots recolectores completaron toda su calle, un robot fiscalizador deberá juntar todas las flores de la esquina (50,50) e informar la cantidad total de flores juntadas. Los robots recolectores inician en las esquinas (1,1), (1,2) y (1,3) respectivamente. El robot fiscalizador inicia en la esquina (1,4).

3- Dos robots recolectores avanzan por las calles 3 y 4 respectivamente juntando todas las flores a su paso. Cada esquina tiene a lo sumo una flor. Cada vez que juntan 10 flores o que avanzan 15 esquinas, deberán vaciar de flores su bolsa en el depósito localizado en la esquina (10,10).
Cada vez que se depositan flores en el depósito, un robot cosechador deberá juntar dichas flores.
Cuando ambos recolectores hayan completado sus calles, el robot cosechador deberá informar la cantidad de flores recolectadas.
Los recolectores inician en la esquina (1,3) y (1,4), y el cosechador en la esquina (1,5)

4- Tres robots floreros tienen 8 intentos en total para juntar todas las flores dentro del cuadrante comprendido entre las esquinas (40,40) y (60,60). Para ello, en cada intento un robot fiscalizador indicará a un robot aleatorio la esquina a la que debe dirigirse. El fiscalizador calculará esta esquina de manera aleatoria. Al completarse los 8 intentos, los robots floreros deberán depositar todas las flores juntadas en la esquina (10,10), y el robot fiscalizador deberá informar la cantidad total de flores juntadas por los robots.
Los robots floreros inician en las esquinas (1,1), (2,1) y (3,1) respectivamente, y el fiscalizador en la (4,1).

5- Existe un robot servidor que tiene su bolsa con papeles. Tres robots clientes tienen 4 intentos cada uno para solicitar al servidor que les entregue papeles. Cada vez que el servidor recibe un pedido de papeles de un cliente, se ubicará en la esquina (100,1), colocará allí una cantidad aleatoria de papeles (entre 1 y 5) y avisará al cliente correspondiente la cantidad de papeles que le depositó.
Una vez que un cliente recibe un aviso, deberá recolectar uno a uno los papeles que le corresponden y depositarlos en su esquina inicial.
El programa finalizará cuando todos los clientes hayan completado todos sus intentos. Asuma que el servidor tiene los papeles suficientes para cubrir todas las solicitudes.
Los robots clientes inician en las esquinas (10,1), (11,1) y (12,1), y el robot servidor inicia en la esquina (13,1).