



División Académica de Ingeniería
Departamento Académico de Computación

Estructuras de Datos

Cuaderno de Ejercicios

- Conceptos de la POO: herencia, interfaces, clases abstractas, polimorfismo, sobrecarga y sobre escritura de métodos.
- Pila
- Cola
- Conjunto
- Recursión
- Estructuras enlazadas
- Listas

Silvia Guardati

Elaborado: enero 2008

Modificado: agosto 2008, enero 2009, junio 2009, junio 2011, agosto 2012, agosto 2013, mayo 2014, enero 2016

Introducción

En este cuaderno de ejercicios se presentan varios problemas donde se aplican los conceptos vistos en clase. Es muy importante resolver cada uno de los problemas para asegurar que los conceptos se entendieron y, por lo tanto, pudieron aplicarse en la solución de un problema.

Antes de resolver cada problema, ten en cuenta que:

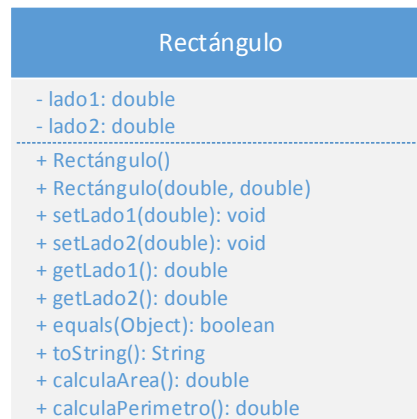
- 1) Es muy importante que leas cuidadosamente todo el enunciado del problema.
- 2) Debes determinar todas las clases involucradas en el problema.
- 3) Para cada clase, debes identificar sus miembros: atributos y métodos.
- 4) Antes de codificar, asegúrate que tu solución (algoritmo) cubre todos los casos que puedan presentarse.
- 5) Revisa tu solución y asegúrate que es eficiente y legible. Además, debes documentarla.
- 6) Debes probar tu solución para garantizar que la misma resuelve el problema planteado de manera **correcta y completa**.

Conceptos básicos de la POO: herencia, polimorfismo, clases abstractas, interfaz, sobrescritura y sobrecarga de métodos.

PROBLEMA 1

Escribe las clases que se mencionan más abajo. Luego compila y ejecuta tu programa. **Asegúrate que genera resultados correctos.**

Clase “Rectángulo” tal como se indica en el diagrama UML:



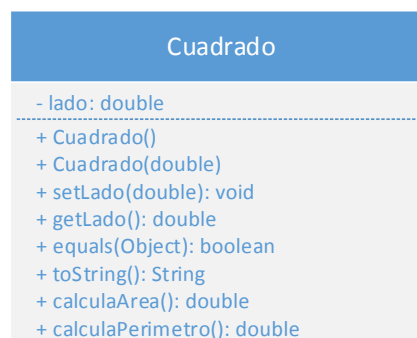
Clase “ComplejoVacacional” que tiene como atributos el nombre del complejo vacacional y los datos de sus *numAlb* albercas ($1 \leq \text{numAlb} \leq 10$). Todas las albercas tienen forma rectangular. Decide qué métodos incluir en esta clase, si el comportamiento esperado de la misma es:

- 1) Calcular y regresar el total de metros de lona que se necesita para cubrir la superficie de todas ellas.
- 2) Calcular y regresar el total de metros lineales requeridos para cercar cada una de las albercas.

PROBLEMA 2

Escribe las clases que se mencionan más abajo. Luego compila y ejecuta tu programa. **Asegúrate que genera resultados correctos.**

Clase “Cuadrado” tal como se indica en el diagrama UML:



Clase “Restaurante” que tiene como atributos el nombre del restaurante y el tamaño de las *numMesas* mesas ($1 \leq \text{numMesas} \leq 25$) que tiene disponible. Todas las mesas son cuadradas. Decide qué métodos incluir en esta clase, si el comportamiento esperado es el siguiente:

- 1) Calcular y regresar el total de metros de tela que se necesita para fabricar manteles para todas ellas. Suponga que a cada lado se le agregan 30 cms.
- 2) Calcular y regresar el total de metros lineales requeridos de puntilla para poner en todos los extremos de los lados de los manteles.

PROBLEMA 3

Escribe las clases que se mencionan más abajo. Luego compila y ejecuta tu programa. **Asegúrate que genera resultados correctos.**

Clase “Alumno” tal como se indica en el diagrama UML. Si lo necesitas puedes agregar otros atributos y/o métodos.



Clase “Escuela” que tiene un nombre, una dirección y *numAlumnos* alumnos ($1 \leq \text{numAlumnos} \leq 50$). La escuela debe tener los métodos necesarios para que pueda mostrar el siguiente comportamiento:

- 1) Dada una cierta clave regresar en forma de cadena todos los datos del alumno correspondiente. Considera todos los casos que puedan presentarse.
- 2) Regresar el nombre y promedio de todos los alumnos.
- 3) Regresar el nombre del alumno que tenga el mayor número de materias aprobadas. Considera que sólo hay un alumno que cumple con esta condición.

PROBLEMA 4

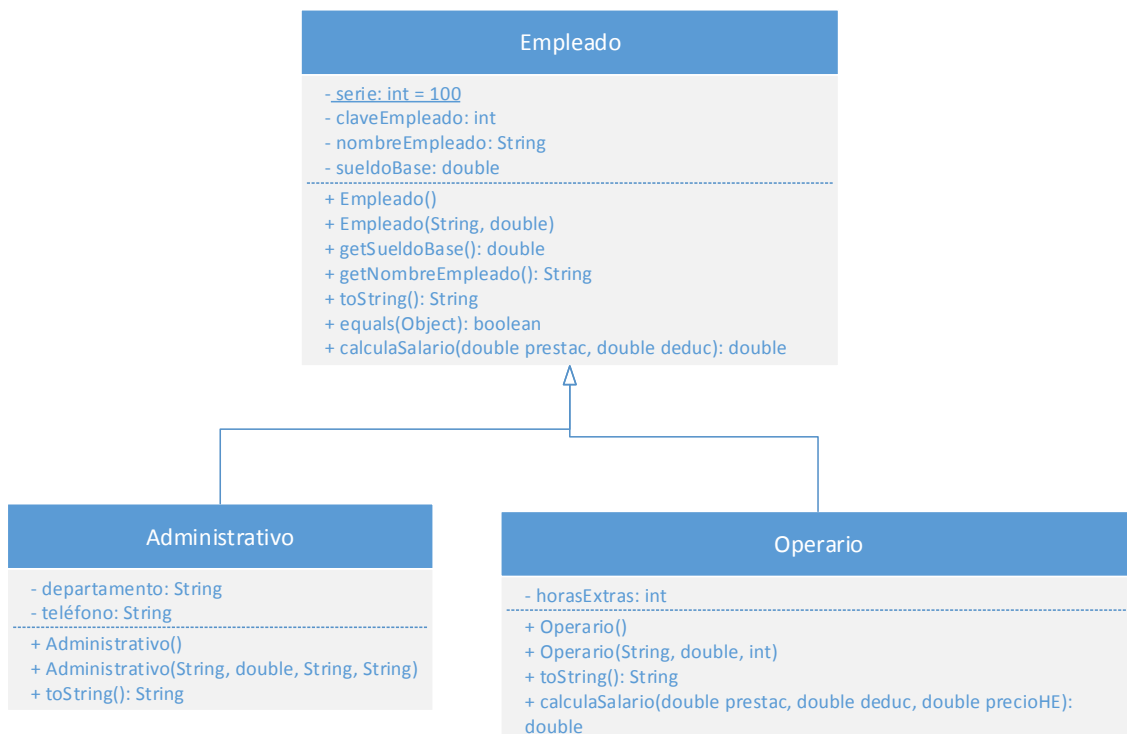
Considera el diagrama de clases que aparece más abajo (puedes agregar atributos y/o métodos si lo crees conveniente). Lee cuidadosamente toda la información que se proporciona. Analiza el problema y plantea la mejor solución. **Prueba tu programa.**

- El salario de un empleado, en general, se calcula como sueldo base + un % de prestaciones (prestac) – un % deducciones (deduc).
- El salario de un operario se calcula como sueldo base + un % de prestaciones (prestac) + las horas extras trabajadas * el precio de cada hora – un % deducciones (deduc).

Se tiene además la clase “Empresa” que tiene como atributos el nombre de la empresa, la dirección, el nombre del dueño, los datos de todos sus empleados administrativos y los datos de

todos sus empleados operarios. De acuerdo a lo que se describe a continuación, decide qué métodos incluir en esta clase. El método main es el encargado de crear el (los) objeto(s) e invocar el(los) método(s) necesario(s) para:

- 1) Dar de alta tanto operarios como administrativos.
- 2) Generar un reporte con los nombres y los sueldos base de cada empleado administrativo.
- 3) Dada la clave de un empleado administrativo y un porcentaje de aumento, si está registrado, actualizar su sueldo. En caso contrario se deberá imprimir un mensaje adecuado.
- 4) Dada la clave de un empleado administrativo y un nombre de departamento, si está, registrar el cambio de departamento. En caso contrario se deberá imprimir un mensaje adecuado.
- 5) Dada la clave de un operario y los datos necesarios, si está, imprimir cuanto cobrará ese operario este mes. En caso contrario se deberá imprimir un mensaje adecuado.
- 6) Generar un reporte con los nombres de todos los operarios que tienen un sueldo base menor a cierta cantidad dada como parámetro. Además, debe incluir el total de empleados que cumplen con esa condición.



PROBLEMA 5

Lee cuidadosamente la información que se proporciona. Analiza el problema y plantea la mejor solución. En el Colegio “*Los libros al poder*” existen cuatro secciones, según el nivel de enseñanza que se imparte:

- ✓ Kinder: comprende maternal, Kinder y preescolar.
- ✓ Primaria: comprende los 6 años de primaria.
- ✓ Secundaria: comprende los 3 años de secundaria.
- ✓ Preparatoria: comprende los 3 años de preparatoria.

De cada alumno se conoce:

1. Kinder:

- ✗ Nombre completo
- ✗ CURP
- ✗ Fecha de nacimiento: mes/día/año
- ✗ Nombre completo de la madre
- ✗ Nombre completo del padre
- ✗ Nombre completo del tutor (este dato puede estar vacío si el alumno tiene a uno o a ambos padres).
- ✗ Nivel: maternal, kinder I, kinder II o preescolar

2. Primaria:

- ✗ Nombre completo
- ✗ CURP
- ✗ Fecha de nacimiento: mes/día/año
- ✗ Nombre completo de la madre
- ✗ Nombre completo del padre
- ✗ Nombre completo del tutor (este dato puede estar vacío si el alumno tiene a uno o a ambos padres).
- ✗ Escuela de procedencia
- ✗ Promedio año anterior
- ✗ Grado que cursa: primero, segundo, ..., sexto
- ✗ Calificaciones de los 5 bimestres del año que cursa (puede que no esté completo, dependiendo de la fecha)

3. Secundaria:

- ✗ Nombre completo
- ✗ CURP
- ✗ Fecha de nacimiento: mes/día/año
- ✗ Nombre completo de la madre
- ✗ Nombre completo del padre
- ✗ Nombre completo del tutor (este dato puede estar vacío si el alumno tiene a uno o a ambos padres).
- ✗ Escuela de procedencia
- ✗ Promedio año anterior
- ✗ Grado que cursa: primero, segundo, tercero
- ✗ Calificaciones de los 5 bimestres del año que cursa (puede que no esté completo, dependiendo de la fecha)
- ✗ Actividad artística/deportiva que practica como parte de su carga escolar

4. Preparatoria:

- ✗ Nombre completo
- ✗ CURP
- ✗ Fecha de nacimiento: mes/día/año
- ✗ Nombre completo de la madre
- ✗ Nombre completo del padre
- ✗ Nombre completo del tutor (este dato puede estar vacío si el alumno tiene a uno o a ambos padres).
- ✗ Escuela de procedencia
- ✗ Promedio año anterior
- ✗ Grado que cursa: primero, segundo, tercero
- ✗ Calificaciones de los 5 bimestres del año que cursa (puede que no esté completo, dependiendo de la fecha)
- ✗ Área de especialización (asumiendo que se elige al empezar el primer año de preparatoria)

Actividades:

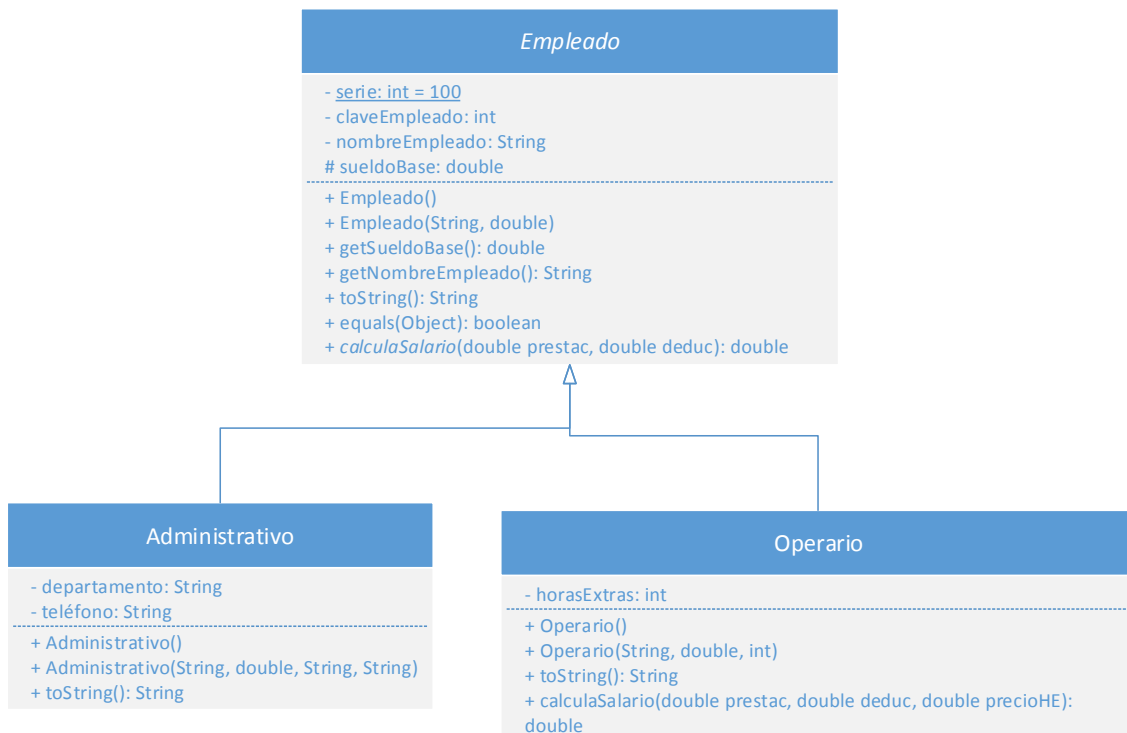
- 1) Define las clases necesarias para representar el conocimiento sobre los alumnos, disponible en la escuela.
- 2) Establece relaciones entre ellas de tal manera que representen la información de manera eficiente **PERO** sin perder la claridad en los conceptos que representan.
- 3) Diseña y desarrolla una aplicación que pueda:
 - a) Almacenar los datos de los alumnos de cada una de las secciones de la escuela.
 - b) Generar un reporte de todos los alumnos –por sección- de la escuela ordenados alfabéticamente.
 - c) Obtener e imprimir el total de alumnos que estén cursando primaria y que tengan como escuela de procedencia la misma escuela.
 - d) Imprimir todos los datos de los alumnos de secundaria que tengan un promedio (hasta el momento) mayor a 9 y que estén a cargo de un tutor.
 - e) Imprimir todos los datos de los alumnos de preparatoria que hayan obtenido un promedio menor a 9 en el año escolar anterior y que estén en el área 1.
 - f) Dado el nombre de un alumno, si está en el colegio, imprimir todos sus datos.

PRUEBA tu solución con pocos datos.

PARA la aplicación puedes simplificar las clases.

PROBLEMA 6

Lee cuidadosamente la información que se proporciona. Analiza el problema y plantea la mejor solución. **Prueba tu programa.**



Nota: El nombre de la clase y del método en *italica* indica que son **abstractos**.

- El salario de un administrativo se calcula como sueldo base + un % de prestaciones (prestac) – un % deducciones (deduc).
- El salario de un operario se calcula como sueldo base + un % de prestaciones (prestac) + las horas extras trabajadas * el precio de cada hora – un % deducciones (deduc).

Se tiene además la clase “Empresa” que tiene como atributos el nombre de la empresa, la dirección, el nombre del dueño, los datos de todos sus empleados administrativos y los datos de todos sus empleados operarios. De acuerdo a lo que se describe a continuación, decide qué métodos incluir en esta clase. El método main es el encargado de crear el(los) objeto(s) e invocar el(los) método(s) necesario(s) para:

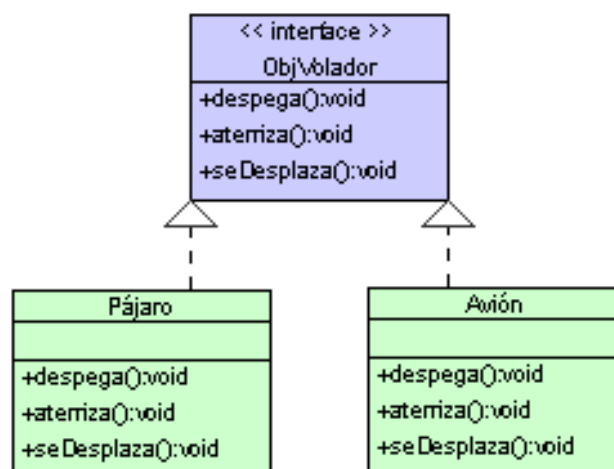
- 1) Dar de alta empleados, tanto operarios como administrativos.
- 2) Generar un reporte con los nombres y los sueldos base de cada empleado administrativo.
- 3) Generar un reporte con los nombres y los sueldos base de cada operario.
- 4) Dada la clave de un empleado administrativo y un porcentaje de aumento, si está registrado, actualizar su sueldo.
- 5) Dada la clave de un empleado administrativo y un nombre de departamento, si está, registrar el cambio de departamento.
- 6) Generar un reporte con los nombres de todos los operarios que tienen un sueldo base menor a cierta cantidad dada como parámetro. Además, debe incluir el total de empleados que cumplen con esa condición.
- 7) Suponiendo que el porcentaje de deducciones y aportaciones es el mismo para todos los administradores, calcular e imprimir el salario de cada uno de los administradores. Además imprimir el total de la nómina de la empresa (sólo con administradores).

(Se pueden omitir algunos de los incisos para simplificar el problema.)

¿Qué tanto cambia esta solución con respecto a la del ejercicio 4? Indica ventajas/desventajas de la nueva solución. ¿Tuviste que usar alguno de los otros conceptos vistos?

PROBLEMA 7

Lee cuidadosamente la información que se presenta más abajo. Analiza el problema y plantea la mejor solución. **Prueba tu programa.** En el diagrama de clases que aparece más abajo, las líneas punteadas indican interface. Agrega los atributos y métodos que creas conveniente.

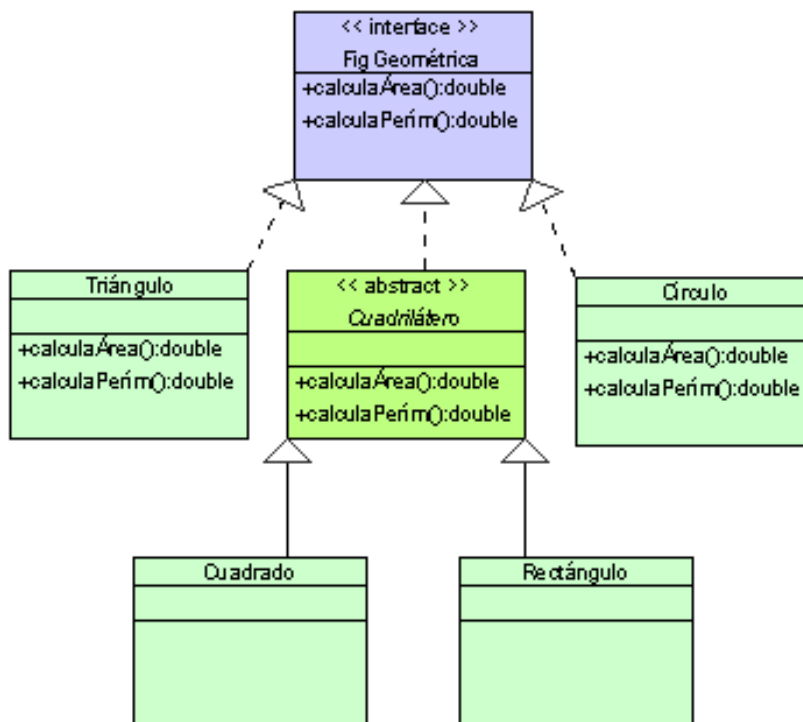


- 1) Define la interface y las clases que aparecen en el diagrama.
- 2) Decide los atributos y métodos a incluir.
- 3) Escribe una clase aplicación que pueda:
 - a) Declarar un arreglo de pájaros y otro de aviones.
 - b) Permita conocer, dada la clave de un pájaro y si fue almacenado, todos los datos de dicho pájaro. En caso contrario desplegar un mensaje adecuado.
 - c) Dada la clave de un avión, imprimir cuántos pasajeros puede transportar, siempre que el avión sea de pasajeros y se encuentre almacenado en el arreglo. En otro caso dar un mensaje.
 - d) Actualizar el hábitat de un pájaro. Decide qué datos necesitas.

PROBLEMA 8

Analiza el siguiente diagrama de clases. Posteriormente:

- 1) Define todas las clases/interfaces representadas en el esquema que aparece más abajo.
- 2) Decide qué atributos y métodos deben incluirse para lograr una definición adecuada de los correspondientes conceptos.



PROBLEMA 9

Retoma el problema 8. Desarrolla una aplicación que pueda, por medio de un menú, realizar las actividades que se describen a continuación.

- 1) Definir un arreglo polimórfico de FigGeométrica.
- 2) Dar de alta figuras.
- 3) Calcular e imprimir el área de todas las figuras almacenadas.
- 4) Encontrar e imprimir los datos del círculo más grande.
- 5) Calcular e imprimir el total de cuadrados.
- 6) Eliminar todos los triángulos equiláteros.

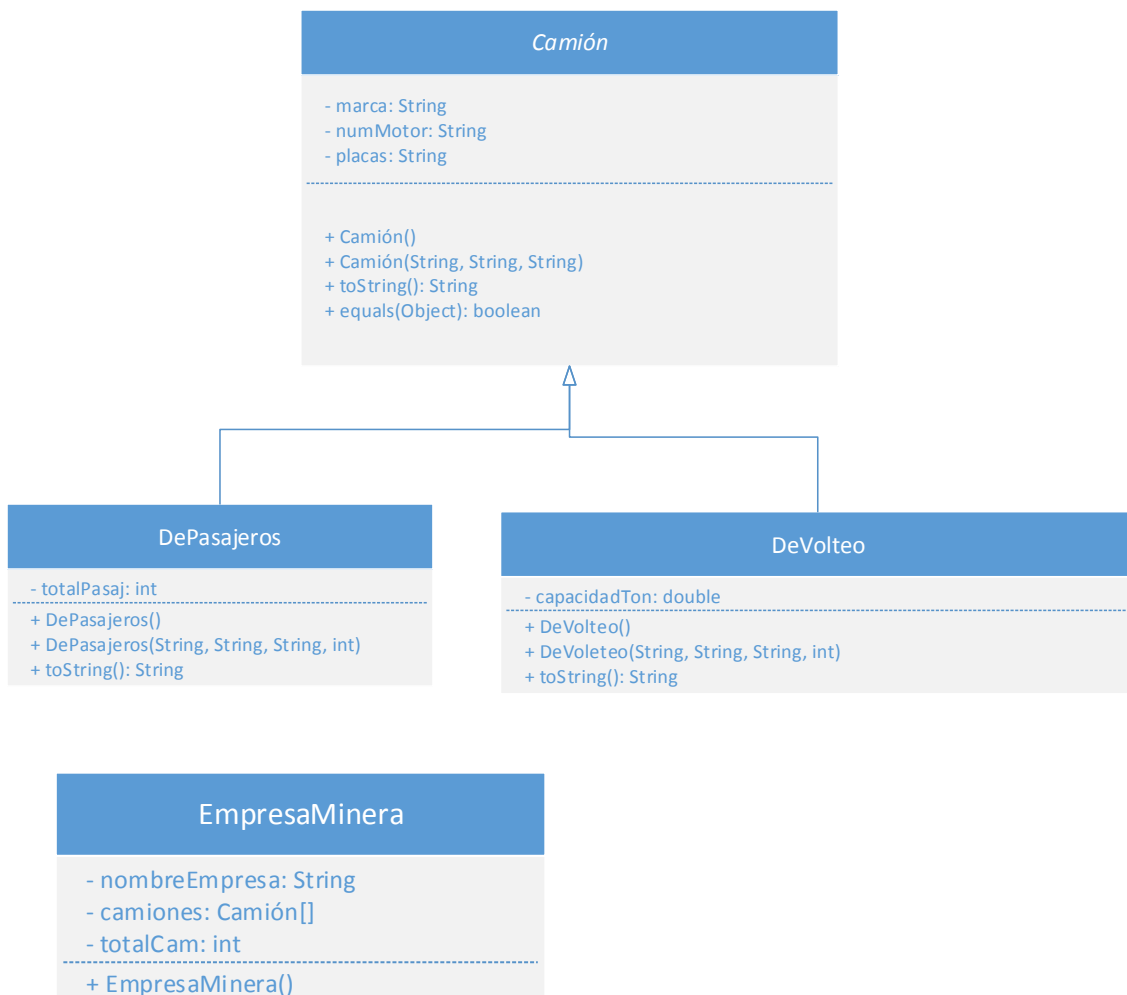
PROBLEMA 10

Una empresa minera tiene camiones de volteo, que utiliza para el movimiento y traslado de tierra, y camiones de pasajeros para transportar a su personal. El esquema de clases que aparece más abajo corresponde a como se han modelado los datos. Puedes agregar atributos y/o métodos si lo crees conveniente.

En la clase EmpresaMinera puedes agregar otros atributos y debes definir los métodos necesarios para que se pueda realizar lo que se describe a continuación:

- 1) Regresar en forma de cadena todos los datos de los camiones de pasajeros.
- 2) Dada la placa de un camión de volteo, actualizar su capacidad de transporte (la nueva capacidad se recibe como parámetro).
- 3) Calcular y regresar el total de camiones de pasajeros que sean de una cierta marca (la marca se recibe como parámetro).
- 4) Calcular y regresar el total de toneladas de tierra que puede ser transportado simultáneamente.

Hacer un main (o usar pruebas unitarias) para probar tu solución.



PROBLEMA 11

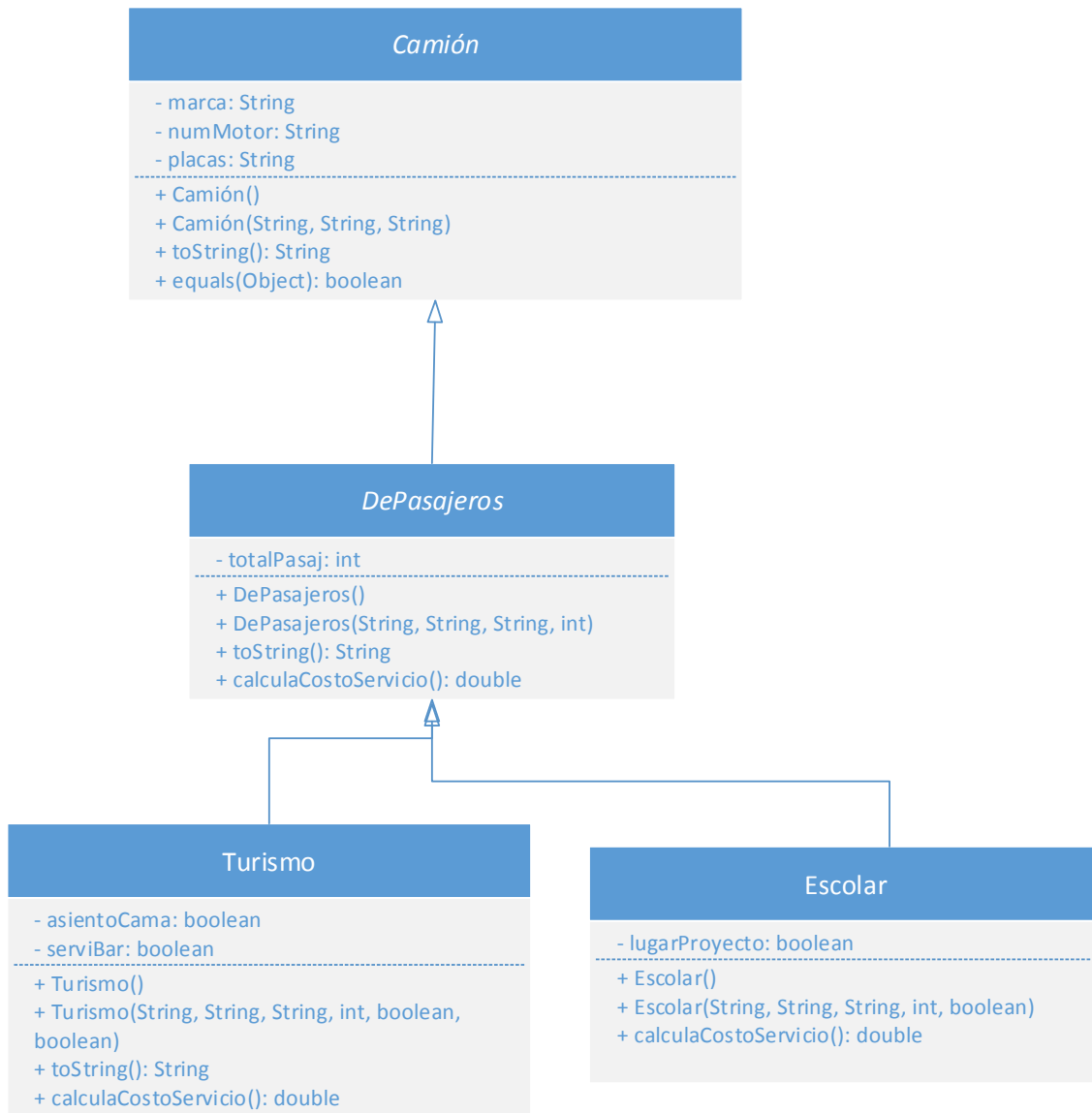
Observa el esquema de clases que aparece más abajo. Reúsa de problemas anteriores todo lo que puedas. Asimismo, considera:

- El costo del servicio para los camiones de pasajeros se calcula como el 0.01% del costo del camión dividido entre el número de pasajeros y multiplicado por la cantidad de kilómetros que recorrerá.
- El costo del servicio para los camiones de pasajeros, dedicados al transporte de escolares se calcula como el 0.01% del costo del camión dividido entre el número de pasajeros, multiplicado por \$250 y eso da el importe mensual que deberá abonar cada niño.
- El costo del servicio para los camiones de pasajeros, dedicados al transporte de turistas se calcula como el 0.01% del costo del camión dividido el número de pasajeros y multiplicado por la cantidad de kilómetros que recorrerá. Además, si el camión cuenta con servicio de bar, el costo se incrementa en un 5%, y si tiene asientos que pueden convertirse en camas, el costo se incrementa en otro 5%.

A su vez, se tiene información de una empresa de transporte, llamada “El rápido” que tiene tanto camiones escolares como de turismo. Define la clase “EmpTransp” que tiene como atributos el nombre de la empresa, el total de camiones y los datos de todos sus camiones. Si quieres puedes agregar otros atributos y debes definir los métodos necesarios para que la clase tenga el siguiente comportamiento:

- 1) Llega un cliente para rentar el servicio de un camión para turistas. Dada la cantidad de pasajeros que se quiere transportar, regresar en forma de cadena las características de todos los camiones disponibles y el costo del servicio de cada uno de ellos.
- 2) Llega el director de una escuela para rentar el servicio de varios camiones escolares. Dada la cantidad máxima de niños que se desea transportar por camión y la cantidad de camiones requeridos, indicar si es posible satisfacer la demanda.
- 3) Dado el número de placas de un camión, indicar si está disponible. Si lo está, regresar qué tipo de unidad es.
- 4) Obtener y regresar en forma de cadena los números de placas de todos los camiones escolares que estén disponibles, que puedan llevar más de 20 alumnos y que cuenten con un espacio para que los estudiantes lleven sus proyectos.
- 5) Obtener y regresar el total de camiones de una cierta marca (la marca se recibe como parámetro), destinados al transporte de turistas que cuenten con servicio de bar y que el costo del servicio sea inferior a una cantidad recibida como parámetro.

Hacer un main (o usar pruebas unitarias) para probar tu solución.



Estructura de datos Pila: definición de la clase Pila y su uso en la solución de problemas.

PROBLEMA 12

Lee cuidadosamente la información que se presenta más abajo. Analiza el problema y plantea la mejor solución. **Prueba tu solución.**

Se requiere de un programa que sea capaz de analizar una cadena (un dato tipo String) para determinar si en la misma hay igual número de paréntesis izquierdo que derechos. En tu solución debes tener:

- Interface Pila: según lo estudiado en clase.
- Clase Pila: según lo estudiado en clase.
- Clase RevisorParentesis: cuyo atributo es la cadena que se quiere analizar. Además debe contar con un método que realice el análisis de la misma (con la ayuda de un objeto tipo Pila) y regrese verdadero si se cumple la condición establecida o falso en caso contrario.

PROBLEMA 13

Lee cuidadosamente la información que se presenta más abajo. Analiza el problema y plantea la mejor solución. **Prueba tu solución.**

Se requiere de un programa que sea capaz de invertir el orden de los caracteres que forman una cadena (un dato tipo String). En tu solución debes tener:

- Interface Pila: según lo estudiado en clase.
- Clase Pila: según lo estudiado en clase.
- Clase InvierteCad: cuyos atributos son la cadenaInicial (la cual se quiere invertir) y la cadenaInvertida (resultado de la inversión). Además debe contar con un método que realice la inversión (con la ayuda de un objeto tipo Pila) de la cadenaInicial y la deje en cadenaInvertida.

PROBLEMA 14

Escribe un método estático que reciba un objeto tipo pila genérica y que regrese el número de elementos que tiene almacenados. INDICA claramente de qué tipo son los datos que utilizas en tu solución. La pila NO puede modificarse, es decir, terminado el método la misma debe quedar como se recibió. **Prueba tu solución.**

PROBLEMA 15

Escribe un método estático que reciba 2 pilas genéricas y regrese true si la primera de ellas “contiene” a la segunda. Una pila p1 contiene a una pila p2 si todos los elementos de p2 están también en p1. Las pilas NO pueden modificarse, es decir, terminado el método las mismas deben quedar como se recibieron. **Prueba tu solución.**

PROBLEMA 16

Escribe un método estático que invierta el orden de los elementos de una estructura tipo Pila genérica. Para ello puedes usar cualquier estructura de datos como auxiliar. **Prueba tu solución.**

PROBLEMA 17

Escribe un método estático que quite todos los elementos repetidos de una estructura tipo pila (si la pila tiene elementos repetidos, estos se encuentran en posiciones consecutivas). **Prueba tu solución.**

Estructura de datos Cola: definición de la clase Cola y su uso en la solución de problemas.

PROBLEMA 18

Se requiere de un método estático que invierta el orden de los elementos de una estructura tipo cola genérica. Para ello puedes usar cualquier estructura de datos como auxiliar. **Prueba tu solución.**

PROBLEMA 19

Escribe un método estático que quite todos los elementos repetidos de una estructura tipo cola (si la cola tiene elementos repetidos, estos se encuentran en posiciones consecutivas). Diseña e implementa un algoritmo iterativo y uno recursivo. Posteriormente analiza ventajas y desventajas de cada uno. **Prueba tu solución.**

PROBLEMA 20

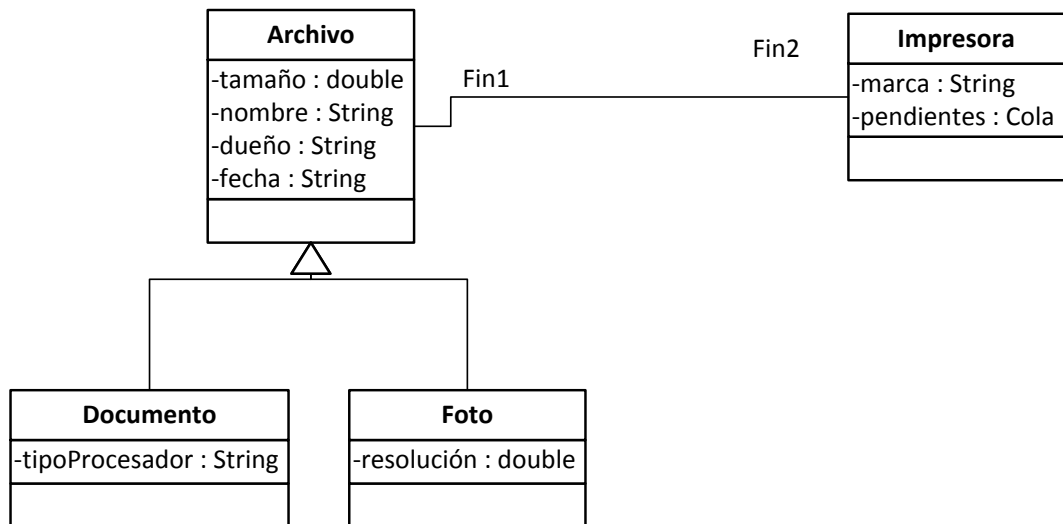
Escribe un método estático que quite todas las ocurrencias de un cierto objeto de una estructura tipo cola. Es decir, dado un objeto, si se encuentra en la cola, se deben eliminar todas sus ocurrencias. Diseña e implementa un algoritmo iterativo y uno recursivo. Posteriormente analiza ventajas y desventajas de cada uno. **Prueba tu solución.**

PROBLEMA 21

Observa el diagrama de clases que aparece más abajo. Debes hacer una aplicación que permita administrar una *impresora*, según lo especificado en el diagrama. La impresora tiene, entre sus atributos, los archivos que debe imprimir, los mismos están encolados. Es decir, se deben imprimir en el orden en el cual llegaron. Si requieres atributos adicionales, puedes agregarlos. En **Impresora** debes agregar los métodos necesarios para poder tener la siguiente funcionalidad. Decide si se necesitan parámetros y el tipo de resultado de los métodos.

- 1) Llega un nuevo archivo para ser impreso.
- 2) La impresora se desocupó, debe imprimir el siguiente archivo.
- 3) Eliminar todas las fotos de la cola de impresión.
- 4) Eliminar todos los documentos cuyo tamaño sea mayor a 500 (KB).

Prueba tus clases desarrollando una pequeña aplicación.



Estructura de datos Conjunto: definición de la clase Conjunto y su uso en la solución de problemas.

PROBLEMA 22

Según lo visto en clase, define una interface ConjuntoADT<T> en la que establezcas el comportamiento esperado de un conjunto. Posteriormente la clase Conjunto con los atributos que lo caracterizan y los métodos que implementen las operaciones establecidas en la interface (agregar un elemento, quitar un elemento, conjunto vacío, cardinalidad, etc.). Deben definirse dos clases, una para implementar el conjunto con un arreglo genérico, y otra usando una estructura enlazada.

Prueba tus clases desarrollando una pequeña aplicación o usando pruebas unitarias.

PROBLEMA 23

En las dos clases hechas en el ejercicio anterior agrega métodos para implementar las siguientes operaciones:

- 1) Unión de conjuntos.
- 2) Intersección de conjuntos.
- 3) Diferencia de conjuntos.

PROBLEMA 24

Define la clase Escuela con los atributos: nombre, dirección, y dos conjuntos de alumnos: uno formado por los alumnos que estudian alguna ingeniería y otro con los alumnos que estudian alguna licenciatura. Debes definir la clase Alumno con los atributos y métodos que creas conveniente. A la clase Escuela le puedes agregar otros atributos si lo consideras necesario. Además, le debes agregar los métodos requeridos para la siguiente funcionalidad:

- 1) Agregar alumnos a cualquiera de los conjuntos.
- 2) Quitar un alumno de cualquiera de los conjuntos.
- 3) Generar y regresar un conjunto formado con todos los alumnos de la escuela. El resultado debe darse en forma de cadena con todos los datos de los alumnos.
- 4) Generar y regresar un conjunto formado con todos los alumnos que están cursando una ingeniería y una licenciatura. El resultado debe darse en forma de cadena con todos los datos de los alumnos.
- 5) Generar y regresar un conjunto formado con todos los alumnos que están cursando solamente una ingeniería o una licenciatura, pero no ambas. El resultado debe darse en forma de cadena con todos los datos de los alumnos.
- 6) Calcular y regresar el promedio de todos los alumnos de Ingeniería.
- 7) Obtener y regresar el total de alumnos de licenciatura que son mayores que una edad (la cual se recibe como parámetro).

NOTA: Para este problema supón que no se pueden cursar dos ingenierías o dos licenciaturas.

Recursión: desarrollo de algoritmos recursivos. Recuerda que algunos de los problemas que debes resolver usando recursión pueden ser resueltos iterativamente de manera más eficiente, considerando el costo de recursos computacionales y la complejidad del código. Sin embargo, por su simplicidad son útiles para reforzar este nuevo concepto.

PROBLEMA 25

Escribe un método estático que reciba un arreglo de enteros y regrese la suma de sus componentes. Debes usar recursión. **Prueba tu solución.**

PROBLEMA 26

Escribe un método estático que reciba un arreglo de enteros e imprima cada uno de sus componentes: a) de izquierda a derecha y b) de derecha a izquierda. Debes usar recursión. **Prueba tu solución.**

PROBLEMA 27

Define la clase ArregloGenérico en la cual escribas los métodos que se mencionan más abajo de manera recursiva. Analiza tus soluciones con respecto a las correspondientes implementaciones iterativas.

- 1) Búsqueda secuencial: recibe un dato tipo T y regresa la posición en la que está o un -1 si no está.
- 2) Búsqueda binaria: recibe un dato tipo T y regresa la posición en la que está o el negativo de la posición + 1 en la que debería estar.
- 3) toString(): regresa el contenido del arreglo en forma de cadena.
- 4) Método que encuentre y regrese la posición del mayor de los elementos del arreglo.
- 5) Método que elimine todas las ocurrencias de un cierto objeto en el arreglo. Decide qué parámetros se necesitan y qué tipo de resultado da el método.
- 6) Escribe una pequeña aplicación para probar tus métodos.

PROBLEMA 28

Define la clase OperacionesArregloBidimensional en la cual escribas los métodos que se mencionan más abajo de manera recursiva. Analiza tus soluciones con respecto a las correspondientes implementaciones iterativas.

- 1) sumaPorRenglón: método que suma por renglón todos los elementos del arreglo bidimensional, regresando la suma obtenida.
- 2) sumaPorColumna: método que suma por columna todos los elementos del arreglo bidimensional, regresando la suma obtenida.
- 3) toString():regresa el contenido del arreglo en forma de cadena.

Estructuras enlazadas: definición de estructuras de datos enlazadas y su uso en la solución de problemas.

PROBLEMA 29

Según lo visto en clase, define las clases que representen a un nodo (Nodo) y a una estructura enlazada (EE). Agrega todos los métodos necesarios para implementar el comportamiento esperado.

PROBLEMA 30

En la clase EE escribe el método: **public int eliminaTodosRepetidosOrdenado()** que elimine todos los elementos repetidos de una EE, dejando sólo una ocurrencia de cada dato. Además, regresa como resultado el total de elementos eliminados. Supón que los elementos de la EE están ordenados y por lo tanto los repetidos ocupan posiciones consecutivas. **Prueba tu solución.**

PROBLEMA 31

En la clase EE escribe el método: **public int eliminaTodosRepetidosDesordenado()** que elimine todos los elementos repetidos de una EE, dejando sólo una ocurrencia de cada dato. Además, regresa como resultado el total de elementos eliminados. Supón que los elementos de la EE están desordenados y por lo tanto los repetidos pueden estar en cualquier posición de la EE. **Prueba tu solución.**

PROBLEMA 32

En la clase EE escribe el método: **public boolean eliminaAnteriorA(T info)** que elimine el nodo anterior al que ocupa la “info”. Regresa true si lo pudo eliminar y false en caso contrario. CONSIDERA TODOS LOS CASOS QUE PUEDEN PRESENTARSE. **Prueba tu solución.**

PROBLEMA 33

En la clase EE escribe el método: **public boolean eliminaSiguienteDe(T info)** que elimine el nodo siguiente al que ocupa la “info”. Regresa true si lo pudo eliminar y false en caso contrario. CONSIDERA TODOS LOS CASOS QUE PUEDEN PRESENTARSE. **Prueba tu solución.**

PROBLEMA 34

En la clase EE escribe el método: **public boolean insertaAntesQue(T refer, T nuevo)** que inserta un nodo (con nuevo como información) antes que el nodo dado como referencia (refer). Regresa true si lo pudo insertar y false en caso contrario. CONSIDERA TODOS LOS CASOS QUE PUEDEN PRESENTARSE **Prueba tu solución.**

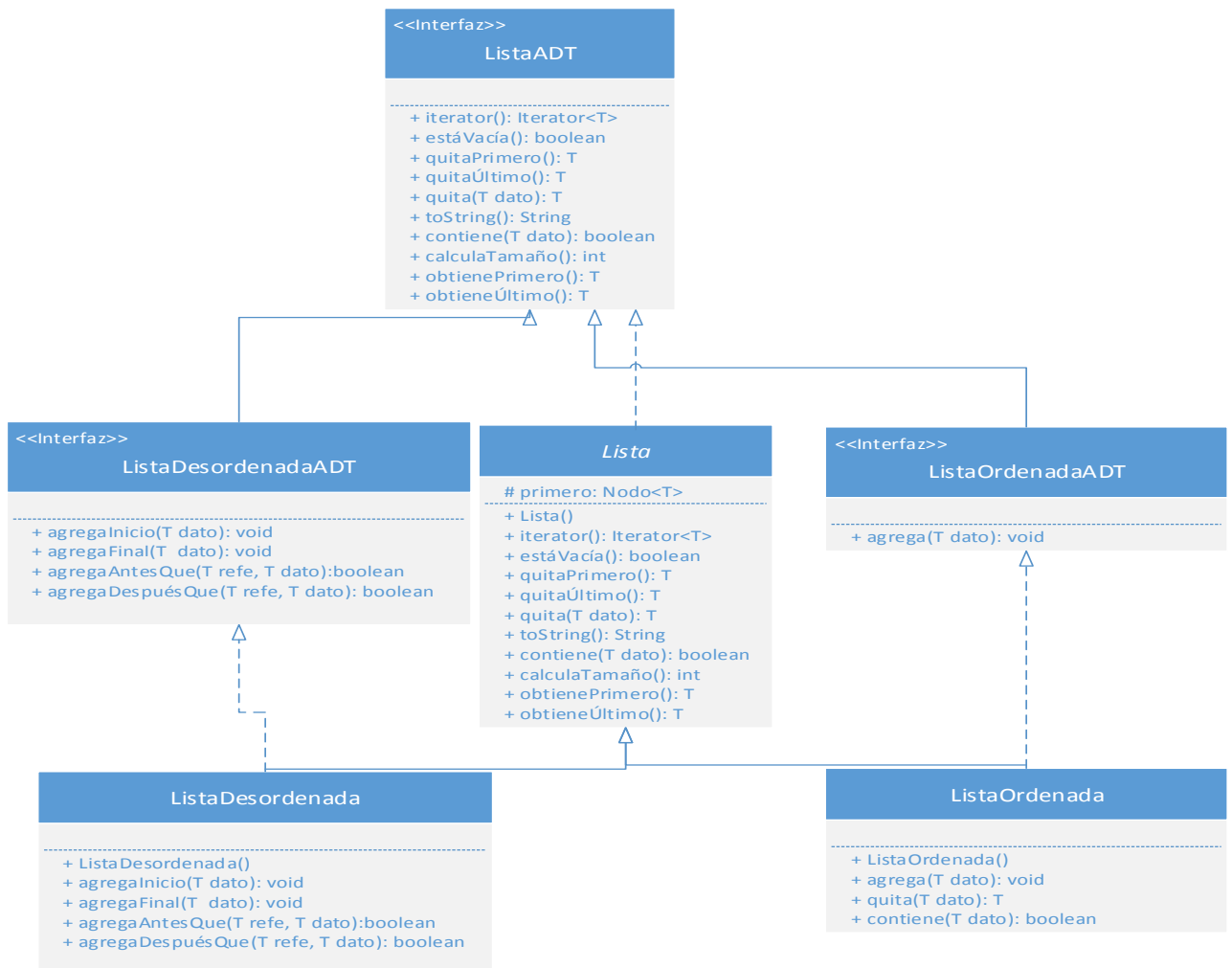
PROBLEMA 35

Escribe un método estático que reciba dos $EE\langle T \rangle$ como parámetros y regrese como resultado una $EE\langle T \rangle$ formada a partir de la mezcla de los nodos de las EE dadas. Es decir, la nueva EE tendrá el primer nodo de la primer EE , el segundo nodo de la segunda EE , el tercer nodo de la primer EE , el cuarto nodo de la segunda EE , etc. SE DEBEN USAR LOS NODOS DE LAS EE PARA FORMAR LA NUEVA, NO SÓLO LAS INFORMACIONES QUE ALMACENAN.
Prueba tu solución.

PROBLEMA 36

Rescribe las clases Pila y Cola usando una EE como base para su implementación (en vez de un arreglo genérico).

Estructura de datos Lista: definición de la clase Lista (y sus variantes) y su uso en la solución de problemas.



PROBLEMA 37

En la clase `ListaDesordenada<T>` vista en clase rescribe de manera recursiva los métodos: `toString()` y el de búsqueda considerando que los elementos están desordenados. **Prueba tu solución.**

PROBLEMA 38

Debes usar las clases `NodoSimple<T>` y `ListaOrdenada<T>` para almacenar los datos de un grupo de personas. En la clase `Persona` incluye los atributos y métodos que creas conveniente. Define la clase `Club` cuyos atributos son *el nombre del club* y *una lista de socios* (personas) ordenados por nombre. Debe tener métodos que permitan dar de alta un socio, quitar un socio y consultar los datos de un socio.

Escribe una pequeña aplicación para probar tu solución.

PROBLEMA 39

Agrega a la clase ListaOrdenada<T> un método equals(ListaOrdenada<T> otra): boolean que permita comprobar si dos listas ordenadas son iguales. Recibe como parámetro una lista y regresa un valor booleano (true si son iguales, false en caso contrario). Dos listas son iguales si tienen **exactamente el mismo** contenido y en el mismo orden. Asegúrate que tu clase ListaOrdenada<T> sigue siendo general. No se deben alterar los contenidos de las listas dadas. Implementa el método solicitado de manera iterativa y de manera recursiva. **Prueba tu solución.**

PROBLEMA 40

Retoma la solución desarrollada en el ejercicio 39. Escribe una aplicación que dadas dos listas de alumnos, si son iguales destruya una de las dos. En caso contrario forme una sola lista con la unión de ambas y la imprima.

PROBLEMA 41

Escribe un programa que, usando las clases NodoSimple<T> y ListaOrdenada<T> sea capaz de almacenar (ordenados alfabéticamente) sólo los nombres de los países de América, según la ubicación: Norte, Centro y Sur, en tres listas distintas. Posteriormente, deberá formar una lista que contenga los nombres de todos los países del continente, también ordenados alfabéticamente. Las 3 listas iniciales no deben alterarse. Para formar la nueva lista se deben ir “mezclando” el contenido de las otras tres manteniendo el orden. Modulariza y **prueba tu solución.**

PROBLEMA 42

Define la clase MateriaSemestre con los siguientes atributos: nombre de la materia, nombre del salón en el cual se imparte, nombre del semestre en el cual está siendo impartida, nombre del maestro que la imparte, lista de alumnos inscritos y lista de libros usados como fuente para dicha materia. Esta clase tiene, además de los métodos esperados (por ejemplo constructores, toString(), ...) los que se describen más abajo. Define las clases Alumno y Libro con los atributos y métodos que creas conveniente. En el main debes incluir las instrucciones necesarias para probar tu solución.

- 1) Un método que permita dar de baja un alumno de dicha materia.
- 2) Un método que permita dar de alta un alumno de dicha materia.
- 3) Un método que permita cambiar el salón en el cual se imparte la materia.
- 4) Un método que permita agregar un libro como referencia de dicha materia.
- 5) Un método que permite quitar uno de los libros dados como referencia de dicha materia.
- 6) Un método que calcula y regresa el promedio de todos los alumnos inscritos en la materia.

- ✓ **En cada caso decide qué dato (o datos) debe suministrar el usuario.**
- ✓ **Decide qué tipo de listas resulta más adecuada para almacenar los alumnos y los libros.**
- ✓ **Considera todos los posibles casos de errores.**
- ✓ **Trata de generalizar tu solución.**
- ✓ **Cuida la eficiencia de tu solución.**

PROBLEMA 43

Define una clase (NodoDoble) para representar un nodo con doble dirección, una de ellas para guardar la dirección del nodo que lo precede y la otra para guardar la dirección del nodo que le sigue. Luego escribe la clase ListaDoble<T> de tal manera que utilice objetos de tipo NodoDoble para formar una lista. Incluye los métodos necesarios para:

- 1) Buscar un elemento en una lista desordenada. Escribe el método de manera recursiva.
- 2) Generar una cadena (toString(): String) con toda la información de la lista. Escribe el método de manera recursiva.
- 3) Determinar si una lista está contenida en otra (estáContenida(ListaDoble<T> otra): boolean). Implementa el método de manera iterativa y recursiva. El método recibe como parámetro una lista y regresa un valor booleano igual a true si la lista recibida como parámetro está contenida en la lista que invoca al método, y false en caso contrario. **Una lista está contenida en otra si todos sus elementos (sin importar el orden) están en la segunda.**
- 4) Determinar si una lista tiene sus elementos ordenados de manera creciente (estáOrdenadaCreciente(): boolean). Realiza una solución iterativa y una recursiva.
- 5) Determinar si una lista NO tiene sus elementos ordenados de manera creciente (noEstáOrdenadaCreciente(): boolean).
- 6) Escribe una aplicación (Empresa) que tiene como atributos dos listas, una de **clientes** y otra de **deudores** y usando el método del inciso 3 determine si todos los deudores de la empresa son sus clientes. En cada caso contrario debe imprimir un mensaje adecuado. Puedes usar una interfaz gráfica para hacer más amigable tu sistema. A la clase Empresa puedes agregarle los atributos y métodos que creas necesarios. Para definir a los clientes y deudores puedes usar la clase Persona de ejercicios anteriores.

PROBLEMA 44

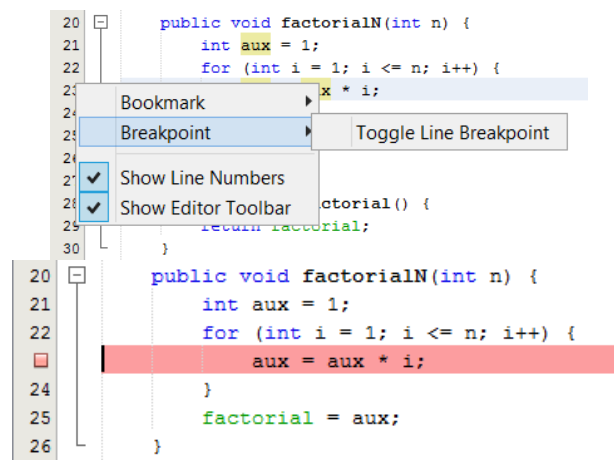
Escribe un método estático que reciba 2 objetos tipo ListaDoble, ambas ordenadas de manera creciente, y que regrese como resultado una lista que sea la mezcla de las 2 listas recibidas. La lista resultado también debe ser de tipo ListaDoble pero debe estar ordenada de manera decreciente.

Uso del debugger en NetBeans

Depurar (o *debugging*) un programa permite analizar su código y sus variables interactivamente durante la ejecución del mismo. Generalmente se usa para identificar fallas o faltas en el código. Se utilizan **breakpoints** en el código para indicar los puntos en los cuales se quiere revisar, ya que estos provocan la detención de la ejecución. Una vez que el programa es detenido se pueden analizar las variables, cambiar su contenido, etc.

¿Cómo colocar los breakpoints?

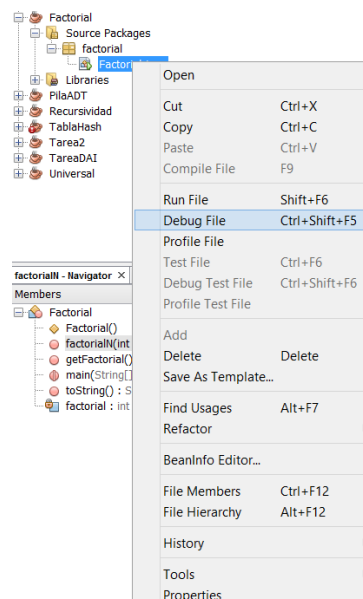
Para colocar un breakpoint en la línea de código elegida, se debe dar clic derecho en el pequeño margen izquierdo y seleccionar **Breakpoint** → **Toggle Line Breakpoint** en el menú desplegable. Otra manera es dar un clic sobre el número de la línea.



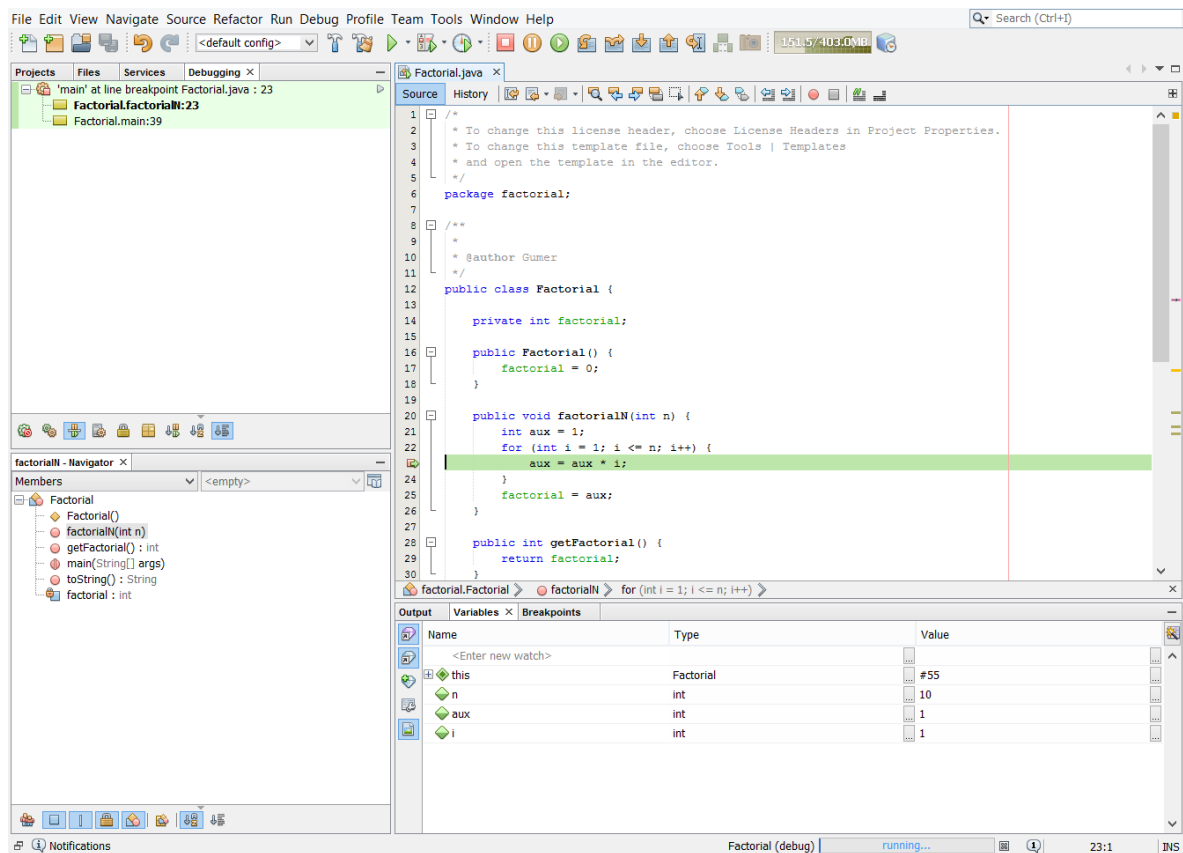
De inmediato aparecerá un cuadro rojo y la línea seleccionada será resaltada en el mismo color.

¿Cómo iniciar el Debugger?

Para iniciar el Debugger se debe seleccionar el archivo de Java que será ejecutado, dar clic derecho en él y seleccionar **Debug File**. Otra manera es presionando las teclas Ctrl + Shift + F5 simultáneamente.



Si no se ha definido ningún breakpoint, el programa correrá normal. Para utilizar el depurador se requiere definir primero los breakpoints. Una vez iniciado el Debugger, NetBeans abrirá una vista similar a la siguiente:



El programa se ejecuta inmediatamente y el debugger está listo para usarse. Automáticamente se detiene en el primer breakpoint para que el usuario pueda empezar su análisis. La línea de código con el breakpoint en turno será marcada de color verde.

¿Cómo se analiza el programa?

Junto a la consola (Output) aparecerán las ventanas *Variables* y *Breakpoints*, en cada una se podrán visualizar sus correspondientes elementos.

La pestaña *Breakpoints* permite eliminar o desactivar breakpoints.









La pestaña *Variables* muestra tipos y contenidos de las variables a medida que las mismas se van usando durante la ejecución. En esta ventana también se puede cambiar el valor asignado a una variable en tiempo de ejecución.

En la parte superior derecha de la ventana de NetBeans, se muestra el uso de memoria actual de tu programa.

89,9/223,1MB

¿Cómo controlar la ejecución del programa mientras se usa el depurador?

En la barra de herramientas aparecen botones para controlar la ejecución del programa, aunque también se pueden usar las teclas correspondientes.

Tecla	Descripción
F5 	Ejecuta la línea actualmente seleccionada y avanza hasta el siguiente breakpoint o hasta el final del programa según sea el caso.
Shift + F5 	Finaliza el debugger y con él la ejecución del programa.
F7 	Hace que el debugger "entre en" (step into) el código. Muestra el código de métodos invocados y profundizará en él cuando se realizan muchas llamadas a métodos anidados.
Ctrl + F7 	Hace que el debugger salga de la función en la que se encuentra y avanza a la siguiente línea del programa.
F8 	Provoca que el código "pase a lo siguiente (step over)". Ignora el funcionamiento interno de las llamadas a funciones y sólo se enfoca en el valor que retorna.
Shift + F8 	Evalúa la expresión que consiste en llamadas a múltiples métodos, si no hay llamadas a métodos actúa como un F8 (step over).
F4 	Continúa la ejecución del programa hasta que se tope con el cursor.
	Pausa la ejecución.

Ejemplo

```
public class Factorial {  
  
    private int factorial;  
  
    public Factorial() {  
        factorial = 0;  
    }  
  
    public void factorialN(int n) {  
        int aux = 1;  
        for (int i = 1; i <= n; i++) {  
            aux = aux * i;  
        }  
        factorial = aux;  
    }  
  
    public int getFactorial() {  
        return factorial;  
    }  
  
    public static void main(String[] args) {  
        Factorial prueba = new Factorial();  
        int n = 10;  
        prueba.factorialN(n);  
        System.out.println("El factorial de " + n + " es igual a " + prueba.getFactorial());  
    }  
}
```

1. Captura el código anterior.
2. Coloca un breakpoint dentro del método factorialN () de la clase Factorial. Utiliza el debug para correr el programa y sigue la ejecución del método.
3. Podrás analizar cómo va aumentando la variable i y como va cambiando la variable aux. Cuando acabe de ejecutarse el método, el Debugger regresará al main para seguir con la ejecución e imprimir en la consola.