Objetos

[C.01] - Creo que necesito un préstamo

Crear una aplicación de consola que contenga la clase Cuenta.

Deberá tener los atributos:

- titular que contendrá la razón social del titular de la cuenta.
- cantidad que será un número decimal que representa al monto actual de dinero en la cuenta.

Construir los siguientes métodos para la clase:

- Un constructor que permita inicializar todos los atributos.
- Un método *getter* para cada atributo.
- mostrar retornará una cadena de texto con todos los datos de la cuenta.
- ingresar recibirá un monto para acreditar a la cuenta. Si el monto ingresado es negativo, no se hará nada.
- retirar recibirá un monto para debitar de la cuenta. La cuenta puede quedar en negativo.

En el método main, simular depósitos y extracciones de dinero de la cuenta, e ir mostrando como va variando el saldo.

[C.02] - Adivinate el numerín, crack!

Crear un juego simple de adivinanza de números. El jugador debe adivinar un número secreto generado aleatoriamente. La aplicación dará pistas sobre si el número ingresado es mayor o menor que el número a ser adivinado, mientras muestra la cantidad de intentos realizados por el jugador.

Crear el proyecto de Java *AdivinarNumeros* que contenga la clase Adivinadora, la cuál tendrá los siguientes miembros:

Atributos:

- numeroSecreto el cuál contiene un valor aleatorio entero.
- intentos indicará la cantidad de veces que el jugador ha intentado adivinar el número secreto.

Métodos:

- Constructor que inicialice el atributo numeroSecreto con un valor aleatorio entre uno y cien (utilizar el método de instancia 'nextInt' de la clase Random) y en cero al atributo intentos.
- adivinar(int) compara el número recibido cómo parámetro con el atributo numeroSecreto. Retornará la cadena de texto *"Mayor"*, *"Menor"* o *"Igual"*, según corresponda. Cada vez que es invocado, incrementará en una unidad el valor del atributo intentos.

En el método main, crear una instancia de la clase Adivinadora y permitir que el jugador ingrese un número mientras no lo haya adivinado. La aplicación le indicará si el número ingresado es menor o mayor al número secreto y en todo momento sabrá la cantidad de intentos realizados.
Cuando el jugador haya ganado, mostrar la cantidad de intentos realizados y en número secreto con el siguiente formato:
"Ganaste!!! El número 31 lo adivinaste en 912 intentos!!!"
Siendo 31 el número secreto y 912 la cantidad de intentos realizados.

Nota: generar una versión que tenga en cuenta distintos niveles de dificultad. Nivel ***fácil***, cantidad ilimitada de intentos fallidos. Nivel ***medio***, hasta 50 intentos. Nivel ***difícil***, hasta 10 intentos.

[C.03] - ¿Vos cuántas primaveras tenés?

Crear una aplicación de consola que contenga la clase Persona.

Deberá tener los atributos:

- nombre
- fechaDeNacimiento
- dni

Deberá tener un constructor que inicialice todos los atributos.

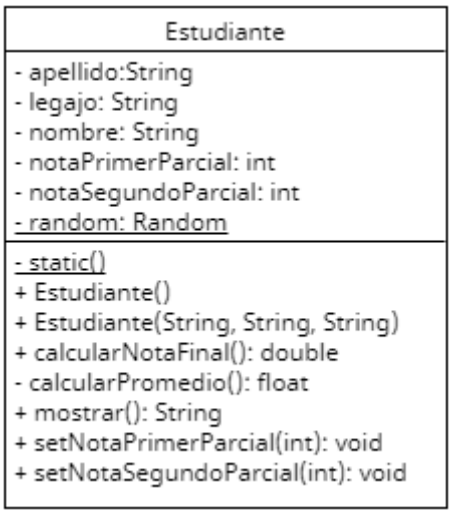
Construir los siguientes métodos para la clase:

- *setter* y *getter* para cada uno de los atributos.
- calcularEdad será privado y retornará la edad de la persona calculada a partir de la fecha de nacimiento.
- mostrar retornará una cadena de texto con todos los datos de la persona, incluyendo la edad actual.
- esMayorDeEdad si es mayor de edad devuelve el valor "*Es mayor de edad*", sino devuelve "*es menor*".

1. Instanciar 3 objetos de tipo Persona en el método main.
2. Mostrar quiénes son mayores de edad y quiénes no.

[C.04] - El ejemplo universal

Crear una aplicación de consola que contenga la clase del siguiente diagrama:



La clase Estudiante:

- Tendrá un bloque estático que inicializará el atributo estático random.
- Tendrá un constructor de instancia que inicializará los atributos nombre, apellido y legajo.
- El método setter setNotaPrimerParcial permitirá cambiar el valor del atributo notaPrimerParcial.
- El método setter setNotaSegundoParcial permitirá cambiar el valor del atributo notaSegundoParcial.
- El método privado calcularPromedio retornará el promedio de las dos notas.
- El método calcularNotaFinal deberá retornar la nota del final con un número aleatorio entre 6 y 10 incluidos siempre y cuando las notas del primer y segundo parcial sean mayores o iguales a 4, caso contrario la inicializará con el valor -1.
- El método mostrar utilizará StringBuilder para armar una cadena de texto con todos los datos de los alumnos:
    - Nombre, apellido y legajo.
    - Nota del primer y segundo parcial.
    - Promedio.
    - Nota final. Se mostrará sólo si el valor es distinto de -1, caso contrario se mostrará la leyenda "*Alumno desaprobado*".

En el método main:

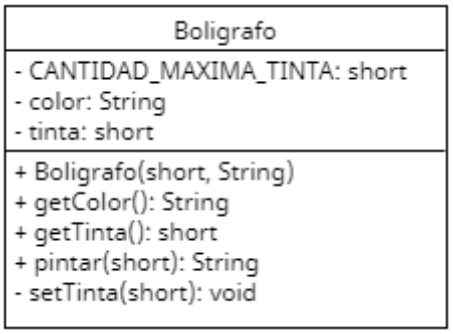
1. Crear tres instancias de la clase Estudiante (tres objetos).
2. Cargar las notas del primer y segundo parcial a todos los alumnos. Dos deberán estar aprobados y uno desaprobado.
3. Mostrar los datos de todos los alumnos.

IMPORTANTE

Para darle un valor aleatorio a la nota final utilice el método de instancia nextInt de la clase Random.

[C.05] - Invento argentino

En un proyecto de biblioteca de clases, crear la clase Boligrafo a partir del siguiente diagrama:

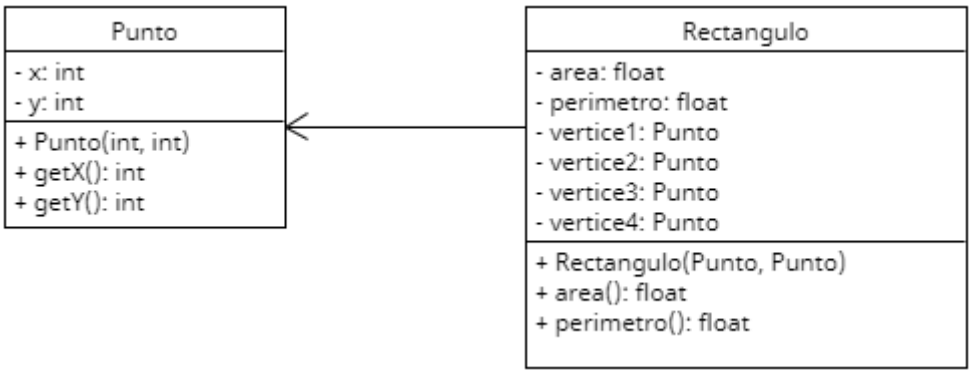


- La cantidad máxima de tinta para todos los bolígrafos será de 100. Generar una constante CANTIDAD_MAXIMA_TINTA en Boligrafo donde se guardará dicho valor.
- Generar los métodos getter getColor y getTinta para los correspondientes atributos (sólo retornarán el valor de los mismos).
- Generar un método setter privado setTinta que valide el nivel de tinta y, si es válido, modifique el valor del atributo tinta.
    - El argumento de tipo short contendrá la cantidad de tinta a agregar o quitar. Podrá ser positivo (cargar tinta) o negativo (gastar tinta).
    - Se deberá validar que el nivel de tinta resultante sea mayor o igual a cero y menor o igual a CANTIDAD_MAXIMA_TINTA. Si no es válido, no se deberá modificar el atributo ni realizar ninguna acción.
- El método recargar colocará la tinta en su nivel máximo. *Reutilizar código*.
- El método pintar restará la tinta gastada (*reutilizar código*). El parámetro gasto representará la cantidad de unidades de tinta a utilizar y utilizará tanta tinta como tenga disponible sin quedar en negativo. El método retornará cómo resultado una cadena con tantos * como unidades de tinta haya gastado, por ejemplo:
    - Si no había nada de tinta retornará una cadena de texto vacía.
    - Si el nivel de tinta era 10 y la cantidad a gastar 2, entonces retornará **.
    - Si el nivel de tinta era 3 y la cantidad a gastar 10, entonces retornará ***.

1. Agregar el **.jar** en un nuevo proyecto de consola.
2. En el método main, crear un bolígrafo de tinta azul y una cantidad inicial de tinta de 100 unidades y otro de tinta roja y 50 unidades de tinta.
3. Utilizar todos los métodos y mostrar los resultados por consola.
4. Al utilizar el método pintar, si corresponde, se deberá dibujar por pantalla con el color de dicho bolígrafo.

[C.06] - Prueba de geometría

En un proyecto de biblioteca de clases, crear las clases modeladas en el siguiente diagrama:



Ambas clases deberán encontrarse dentro del paquete Geometria.

La clase Punto debe tener:

- Dos atributos *privados* con acceso de sólo lectura (sólo con getters), que serán las coordenadas del punto.
- Un constructor que reciba los parámetros x e y.

La clase Rectangulo:

- Tiene los atributos de tipo Punto: vertice1, vertice2, vertice3 y vertice4 (que corresponden a los cuatro vértices del rectángulo).
- La base de todos los rectángulos de esta clase será siempre horizontal. El constructor calculará los vértices 2 y 4 del rectángulo a partir de los vértices 1 y 3. Utilizar el método abs de la clase Math que retorna el valor absoluto de un número y será necesario para obtener la distancia entre puntos.
- Realizar los métodos getters para los atributos privados area y perimetro.
- El área (base * altura) y el perímetro(base + altura) / 2) se deberán calcular sólo una vez cuando se llame por primera vez a su correspondiente método getter. En las siguientes invocaciones de dichos métodos se deberá retornar siempre el valor calculado anteriormente.

1. Agregar el **.jar** en un nuevo proyecto de consola.
2. En la clase que contiene al método main, desarrollar un método de clase (estático) que muestre todos los datos de una instancia de Rectangulo que reciba como parámetro.
3. En el método main probamos las funcionalidades de las clases Punto y Rectángulo.
    1. Instanciar un nuevo Rectangulo.
    2. Imprimir por pantalla los valores de área y perímetro.

[C.07] - Administrar libros

Crear una aplicación de consola que contenga la clase Libro.

Deberá tener los atributos:

- titulo representa el título del libro en formato de cadena de texto.
- autor se guarda el nombre del autor del libro.
- disponible indica mediante un valor booleano si el libro está o no disponible.

Deberá tener un constructor que inicialice todos los atributos.

Construir los siguientes métodos:

- *setter* y *getter* para cada uno de los atributos.
- mostrarInformacion imprime por consola la información completa del libro.
- prestar marca el libro cómo no disponible.
- devolver marca el libro cómo disponible.

Crear la clase Lector.

Deberá tener los atributos:

- nombre representa el nombre del lector, en formato de cadena de texto.
- libroPrestado almacena al objeto de tipo Libro que el lector está leyendo.

Deberá tener un constructor que reciba el nombre del lector.

Construir los siguientes métodos:

- tomarPrestado si el libro que recibe cómo parámetro está disponible, lo asigna a libroPrestado e invoca al método prestar del libro.
- devolverLibro marca el libro cómo disponible y asigna ***null*** al atributo libroPrestado.
- mostrarInformacion imprime por consola la información completa del lector.

En el método main, crear instancias de las clases Libro y Lector. Probar el correcto funcionamiento de los métodos en ambas clases.