Ejercicios

Ejercicio1. Cafetera

Desarrolla una clase **Cafetera** con atributos ***capacidadMaxima*** (la cantidad máxima de café que puede contener la cafetera) y ***cantidadActual*** (la cantidad actual de café que hay en la cafetera).

Implementa, al menos, los siguientes métodos:

- Constructor predeterminado: establece la capacidad máxima en 1000 (ml) y la actual en cero(cafetera vacía).
- Constructor con la capacidad máxima de la cafetera; inicializa la cantidad actual de café igual a la capacidad máxima.
- Constructor con la capacidad máxima y la cantidad actual. Si la cantidad actual es mayor que lacapacidad máxima de la cafetera, la ajustará al máximo.
- Getters y Setters.
- **llenarCafetera()**: hace que la cantidad actual sea igual a la capacidad.
- **servirTaza(int)**: simula la acción de servir una taza con la capacidad indicada.Si la cantidad actual de café "no alcanza" para llenar la taza, se sirve lo que quede.
- **vaciarCafetera()**: pone la cantidad de café actual en cero.
- **agregarCafe(int)**: añade a la cafetera la cantidad de café indicada.

Ejercicio2. Fracción

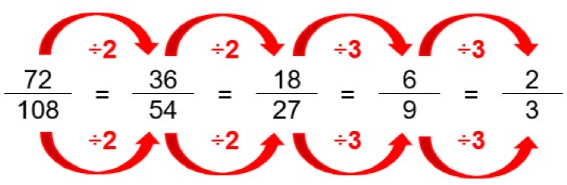
Crea la clase **Fraccion**. Los atributos serán ***numerador*** y ***denominador***.

El método Constructor creará la fracción 1/1.

Implementa los métodos:

- Invertir
- Simplificar
- Multiplicar
- Dividir

Realizar un programa para probar todas las funcionalidades de la clase Fracción



Ejercicio3. Tiempo

Crea la clase **Tiempo** con los métodos **suma** y **resta**.

Los objetos de la clase Tiempo son intervalos de tiempo y se crean de la forma

```
Tiempo t = new Tiempo(1,20,30)
```

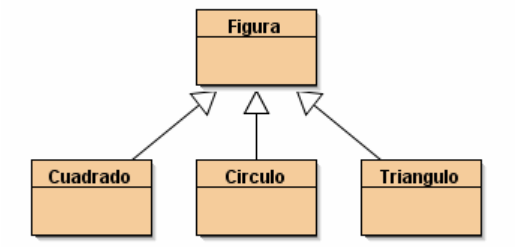
donde los parámetros que se le pasan al constructor son las horas, los minutos y los segundos respectivamente.

- Crea el método **toString** para ver los intervalos de tiempo de la forma 10h 35m 5s.
- Si se suman por ejemplo 30m 40s y 35m 20s el resultado debería ser 1h 6m 0s.
- Si se restan y el primer tiempo es menor que el segundo se escribirá 0s.
- Crea el método **compareTo** para poder comparar tiempos y así poder ordenar.

Realiza un programa de prueba para comprobar que la clase funciona bien. Dentro de ese programa haz que permita introducir 10 tiempos y los ordene de menor a mayor.

Ejercicio4. Figuras

Escribe un programa que implemente la siguiente jerarquía de clases:



Implementar los atributos y métodos necesarios para ejecutar el siguiente programa:

```
public class CreaFiguras {
	public static void main(String[] args) {
		Vector<Figura> figuras = new Vector<Figura>();
		figuras.add(new Circulo(10)); // Radio=10
		figuras.add(new Cuadrado(10)); // Lado=10
		figuras.add(new Triangulo(10,5)); // Base=10, Altura=5;
		for (Figura f: figuras){
			System.out.println("Área: "+f.area());
			System.out.println("Perímetro: "+f.perimetro());
		}
	}
}
```

Al ejecutar el programa, deberá aparecer por pantalla el área y el perímetro de cada una de las figuras creadas.

Ayuda

Circulo: área (PI * radio * radio) - perímetro (2 * PI * radio)
Cuadrado: área (lado * lado) - perímetro del Cuadrado (4 * lado)
Triángulo: área((base * altura) / 2) - perímetro (suponemos isósceles – usar Pitágoras)

Ejercicio5. Vehículos

Crea la clase **Vehiculo**. A partir de esta crea las subclases **Bicicleta** y **Coche**. Para la clase Vehiculo, crea los atributos de clase **vehiculosCreados** y **kilometrosTotales**, así como el atributo de instancia **kilometrosRecorridos,** por cada vehículo.

Crea los métodos necesarios para gestionar las clases.

Crea un programa con un menú como el que se muestra:

VEHÍCULOS
=========
1. Anda en bicicleta
2. Anda en coche
3. Ver kilometraje de la bicicleta
4. Ver kilometraje del coche
5. Ver kilometraje total
6. Ver vehículos totales
7. Salir
Elige una opción (1-7):

Ejercicio6. Universidad

Crear una pequeña base de datos de personas del instituto. Implementa las siguientes clases:

- **Direccion**:
  - Atributos: calle, ciudad, código postal, pais
  - Constructores predeterminado y parametrizado.
- **Persona**:
  - Atributos: nombre, apellidos, NIF y Direccion
  - Constructores parametrizado con o sin dirección
- **Estudiante**: Subclase de Persona.
  - Atributos: IDestudiante
  - Constructores : predeterminado y constructor parametrizado que admita el ID.
  - Métodos getters, setters y toString().
- **Profesor**: Subclase de Persona.
  - Atributos : ndespacho
  - Constructores: predeterminado y constructor parametrizado que admita el despacho.
  - Métodos getters, setters y toString().

Crea una lista de personas y prueba a añadir varios alumnos y varios profesores a la lista y sus operaciones.