Ejercicios POO+

Ejercicio1. Tarjeta SIM

Implementa la clase TarjetaSIM. Una Tarjeta tiene asociado un número.

Se pueden comunicar unas con otras y el tiempo de conversación se añade para ambas tarjetas.

A continuación, se proporciona el contenido del main y el resultado que debe aparecer por pantalla.

```
TarjetaSIM t1 = new TarjetaSIM("678112233");
TarjetaSIM t2 = new TarjetaSIM("644744469");
TarjetaSIM t3 = new TarjetaSIM("622328909");
TarjetaSIM t4 = new TarjetaSIM("664739818");
    System.out.println(t1);
    System.out.println(t2);
    t1.llama(t2, 320);
    t1.llama(t3, 200);
    System.out.println(t1);
    System.out.println(t2);
    System.out.println(t3);
    System.out.println(t4);
```

Salida del programa:

```
Num. 678112233 – 0s de conexión
Num. 644744469 – 0s de conexión
Num. 678112233 – 520s de conexión
Num. 644744469 – 320s de conexión
Num. 622328909 – 200s de conexión
Num. 664739818 – 0s de conexión
```

Ejercicio2. Movil

Implementa la clase Movil como subclase de TarjetaSIM (no hace falta modificar).
Cada móvil lleva asociada una tarifa que puede ser "plata", "oro" o "platinum". El coste por minuto es de 10, 6 y 3 céntimos respectivamente. Se tarifican los segundos exactos. Obviamente, cuando un móvil llama a otro, se le cobra al que llama, no al que recibe la llamada.
A continuación, se proporciona el contenido del main y el resultado que debe aparecer por pantalla. Para que el total tarificado aparezca con dos decimales, puedes utilizar `DecimalFormat`.

Programa Principal:

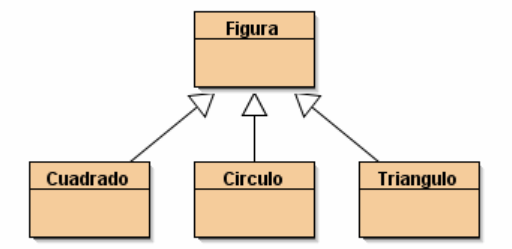
```
Movil m1 = new Movil("678112233", "plata");
Movil m2 = new Movil("644744469", "oro");
Movil m3 = new Movil("622328909", "platinum");
System.out.println(m1);
System.out.println(m2);
m1.llama(m2, 320);
m1.llama(m3, 200);
m2.llama(m3, 550);
System.out.println(m1);
System.out.println(m2);
System.out.println(m3);
```

Salida del programa:

```
Num. 678112233 – 0s de conexión - 0,00 euros de gasto
Num. 644744469 – 0s de conexión - 0,00 euros de gasto
Num. 678112233 – 520s de conexión - 0,86 euros de gasto
Num. 644744469 – 870s de conexión – 0,87 euros de gasto
Num. 622328909 – 750s de conexión - 0,00 euros de gasto
```

Ejercicio3. 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 (4 * lado)

Triángulo: área((base * altura) / 2) - perímetro (suponemos isósceles – usar Pitágoras)

Ejercicio4. 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):

Ejercicio5. 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.