Ejercicios

Utilizaremos el archivo **cliente.php** del campus.

1. Crear una clase Persona, una clase PYME y una clase Multinacional. Las 3 clases deben heredar de la clase Cliente.
2. Mover los atributos y métodos correspondientes (que sólo correspondan a personas físicas), de la clase Cliente a la clase Persona.
3. Agregar a las clases PYME y Multinacional los atributos y métodos que correspondan. (ej: CUIT, Razón social, etc). Agregar constructor y métodos getters y setters.
4. Realizar los cambios necesarios de modo tal que la clase Cliente pase a ser una clase abstracta.
5. Crear una clase abstracta Cuenta con atributos: CBU, Balance, Fecha de último movimiento. Agregar setters y getters para obtener los datos de la cuenta. Agregar un constructor que permite inicializar el CBU.
6. Agregar un método abstracto para **debitar** cierto monto que reciba como parámetro el monto y desde donde se está haciendo la transacción (cajero Banelco, cajero Link, caja). Agregar otro método (no abstracto) que permita **acreditar** cierto monto (y programar su comportamiento). (Tener en cuenta que cada método debe, además, modificar la fecha de último movimiento).
7. Crear 4 clases llamadas Classic, Gold, Platinum y Black. Todas “hijas” de la clase Cuenta.
8. Implementar los métodos debitar y acreditar en cada clase, con las siguientes reglas de negocio:
    a. Debitar:
        i. Classic: Si la transacción es desde un cajero Banelco, debe debitar un 5% mas. De Link un 10% mas. Desde caja no se agrega importe extra.
        ii. Gold: Desde Banelco y Caja no se agrega importe extra. Desde Link un 5% mas.
        iii. Platinum: 5% desde cualquier medio, a menos que la cuenta tenga un balance de 5.000 o más.
        iv. Black: No se cobra importe extra.
    b. Acreditar: Utiliza la cuenta que realiza su padre, pero debe modificar el importe a acreditar según el tipo de cuenta.
        i. El banco se queda con un 5% si es Classic.
        ii. Se queda con un 3% si es Gold, salvo que se esté acreditando 25.000 o más.
        iii. Si es Platinum el banco no retiene nada.
        iv. 0% para Black o 4% para montos superiores a 1.000.000.
9. Hacer los cambios necesarios en el Cliente, para que pueda tener una cuenta asociada. (Concepto de composición). Agregar la asignación de la cuenta en el constructor.
10. Una vez por mes, el banco debe cobrarle a cada cliente por los servicios prestados. Cada Cliente deberá tener un método que permita cobrar los servicios:

   a. Si el cliente tiene cuenta Classic, el banco cobra un monto fijo de $100.
   b. Si el cliente tiene cuenta Gold, el banco cobra $10 por cada letra del apellido del cliente o $5 por cada letra de la razón social.
   c. Si el cliente tiene una cuenta Platinum el banco cobra un 10% del total del balance de la cuenta.
   d. Si el cliente tiene una cuenta Black, el banco cobra $500, más $100 multiplicado por el número de día de la semana en la que se hizo la última transacción.

```
$date = '2016/09/28';
$weekday = date('l', strtotime($date));
```

11. Generar una interfaz “Liquidable” que declare un método **liquidarHaberes**. Tanto PYME como Multinacional deben implementar “Liquidable”. El método **liquidarHaberes** recibe una Persona y un monto en donde se le paga a la Persona dicho monto de la cuenta de la empresa.
    a. Dentro de PYME este método cobra un 1% de cada liquidación que desaparecen de la cuenta de la PYME.
    b. Dentro de Multinacional este método cobra un valor fijo de $500 que desaparecen de la cuenta de la Multinacional.
12. Crear una interfaz “Imprimible” que declare un método **mostrar**. Este método deberá retornar lo que corresponda, según la clase que lo implemente:
    a. Para Persona, el nombre y apellido.
    b. Para PYME, la razón social
    c. Para Multinacional, la razón social.
    d. Para las cuentas, el balance.
13. Crear un archivo “banco.php”, que instancie un Banco y lo llene con Clientes (que ya deben tener cuentas a la hora de ser creados). Luego de eso, realizar las operaciones disponibles como prueba, imprimiendo para cada una sobre qué Cliente se hizo y como queda el balance de su Cuenta luego de la operación.

Ejercicios Complementarios (Figuras)

1. En un archivo llamado **figura.php.** Crear una clase abstracta llamada **Figura.** La cual tenga los métodos **getPerimetro** y **getArea.**
2. Crear una nueva clase llamada **Rectangulo**, la cual extiende a forma e implemente sus métodos correctamente. (Agregar los atributos y métodos necesarios)
3. Extender nuevamente a **Figura** a una nueva clase llamada **Circulo** que implemente sus métodos correctamente. (Agregar los atributos y métodos necesarios)
4. Extender nuevamente a **Figura** a una nueva clase llamada **Triangulo** que implemente sus métodos correctamente. (Agregar los atributos y métodos necesarios)
5. Realizar la prueba de generar distintos circulos, triangulos y rectangulos probando los métodos **getPerimetro** y **getArea .**
6. Aprovechando el código que ya tenemos crear la clase **RectanguloConRelleno** que permite elegir el color de fondo del rectángulo.

Definiciones: Consideraremos para el ejercicio que dos rectángulos son iguales si su base y altura son las mismas (sin importar color de relleno). Para el círculo, basta con que su radio se igual. Para el triángulo les dejamos averiguar a ustedes como compararlos pero debería seguir un criterio similar a nivel geométrico.

7. Crear la interfaz **Comparable** que contenga el método **equals** el que cual debería poder comparar dos cosas con un criterio elegido (ver el criterio para las figuras).
8. Hacer que **Rectangulo**, **Triangulo**, y **Circulo** implementen **Comparable**.
9. Generar distintos **Rectangulos** y compararlos:
    a. Utilizando ==
    b. Utilizando ===
    c. Utilizando **equals**.

    ¿Qué sucede?
10. Crear la interface **Dibujable** que contenga el método **dibujar** el cual debería generar un <svg> de lo que sea que se vaya a dibujar.
11. Hacer que **Rectangulo**, **Triangulo**, **Circulo** y **RectanguloConRelleno** implementen **Dibujable**.

Ejercicios Complementarios (Calculadora)

1. Crear un interfaz llamada **Operable ,** que tenga como método **operar($num1, $num2)**. Crear las siguientes clases, que implementan la interfaz **Operable**:
    a. **Suma,** donde la implementación del método **operacion** retorna la suma de los dos parametros que recibe.
    b. **Resta,** donde la implementación del método **operacion** retorna la resta de los dos parametros que recibe.
    c. **Division,** donde la implementación del método **operacion** retorna la división de los dos parametros que recibe.
2. Extender la clase **Suma** a una nueva clase llamada **Multiplicacion.** Que **operacion** retorne la suma de **$num1** sumada a si misma la cantidad de veces que indique **$num2**. Utilizar el método **operacion** de la clase padre.
3. Crear una nueva clase llamada **Calculadora** que contará con:
    a. Atributo **operacion** (de tipo **Operacion**)
    b. Atributo **num1**
    c. Atributo **num2**
    d. Método **setNum1**
    e. Método **setNum2**
    f. 3 constantes que representarán las 4 operaciones (‘suma’, ‘resta’, ‘multiplicacion’, ‘division’).
    g. Por construcción, se recibe el identificador de la operación y los números a operar.
    h. Método **makeOperacion** que tomará el identificador de la operación y construirá el objeto Operación correspondiente.
    i. Método **operar** que devuelve el resultado.
4. Crear un archivo main.php que incluya la clase calculadora. Dado esto lograr que devuelva:
    a. El resultado de 1 + 2
    b. El resultado de 10 - 5
    c. El resultado de 100 * 7
    d. El resultado de 500 / 20
5. Crear una clase **OperacionFactory** que tenga un método **make**, el cual recibe un identificador de operacion y construya el objecto **Operacion** correspondiente. Reemplazar el código del método **makeOperacion** por el uso de esta clase.
6. Aprovechando el código ya generado, crear la clase **CalculadoraEnteros** que siempre y sin importar la operación devuelva en su método **operacion** el número redondeado en caso de ser decimal.