PRÁCTICA 1. Fundamentos de POO

Objetivo:

El alumno aplicará los principios del *Paradigma Orientada a Objetos* en el lenguaje de programación Java.

Los conceptos específicos que aplicará son:

- Declaración de clases con una correcta abstracción de atributos y métodos de acuerdo al ámbito del problema.
- Uso de modificadores de acceso
- Uso de constructores
    - Por omisión
    - Por parámetros
- Creación de instancias de clase
    - Invocación a miembros funcionales de clase

DESARROLLO

1. *Sobrecarga de Constructores "Figuras geométricas"*

Defina las clases Figura y Punto, y realice la sobrecarga de constructores como se muestra a continuación.

Punto
- coordX: double - coordY: double
+ Punto( ) + Punto(x:double, y:double) + getX(): double + setX(x:double): void + getY(): double + setY(y:double): void

Figura
- nombre: String - area : double - perimetro : double
+ Figura (p1:Punto, radio:double) + Figura(p1:Punto, p2:Punto) + Figura(p1:Punto, p2:Punto, p3:Punto) + imprimirInformacion():void

Implementación clase Principal

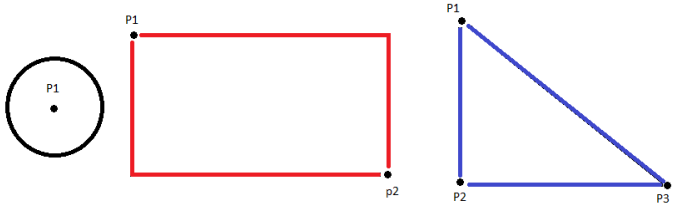
```
public class Principal {
    public static void main(String args[])
    {
        Punto p1, p2, p3, p4, p5;

        p1 = new Punto();
        p2 = new Punto(5.6, 1.6);
        p3 = new Punto(8.9, 7.2);

        Figura figs[] = new Figura[4];
        //Creación de diferentes instancias de figuras
        // mediante la sobrecarga de constructores
        figs[0] = new Figura();
        figs[1] = new Figura(p1, 18.5);
        figs[2] = new Figura(p1, p2);
        figs[3] = new Figura(p1, p2, p3);

        for(int i = 0; i < figs.length; i++)
        {
            figs[i].imprimirInformacion();
        }
    }
}
```

Implementación clase Punto

```
public class Punto {
    //ATRIBUTOS
    private double coordX, coordY;
    //CONSTRUCTOR POR OMISION
    public Punto()
    {
        coordX = 10.0;
        coordY = 10.0;
    }
    //CONSTRUCTOR POR PARAMETROS
    public Punto(double x, double y)
    {
        coordX = x;
        coordY = y;
    }
    //SETTERS Y GETTERS
    public void setX(double x)
    {
        coordX = x;
    }

    public double getX()
    {
        return coordX;
    }

    public void setY(double y)
    {
        coordY = y;
    }

    public double getY()
    {
        return coordY;
    }
}
```

La clase Principal crea un arreglo de 4 diferentes figuras, las cuales son construidas utilizando los diferentes constructores sobrecargados.

La siguiente figura muestra la construcción de las diferentes figuras



Implementación clase Figura

```
public class Figura {
    //ATRIBUTOS
    private String nombre;
    private double area, perimetro;
    //SOBRECARGA DE CONSTRUCTORES
    public Figura()
    {
        nombre = "Cuadrado";
        area = 100;
        perimetro = 40;
    }
    //Contructor para la figura circulo
    public Figura(Punto p1, double radio)
    {
        nombre = "Círculo";
        area =  Math.PI * radio;
        perimetro = Math.PI*2.0*radio;
    }
```

```
    //Contructor para la figura rectangulo
    public Figura(Punto p1, Punto p2)
    {
        nombre = "Rectangulo";
        /*En esta parte del código deberá
         realizar el calculo de el área y perímetro
         para un rectangulo a partir de 2 puntos*/
        area =  200;
        perimetro = 20;
    }
      //Contructor para la figura triángulo
    public Figura(Punto p1, Punto p2, Punto p3)
    {
        nombre = "Triángulo";
        /*En esta parte del código deberá
         realizar el calculo de el área y perímetro
         para un triángulo a partir de 3 puntos*/
        area =  300;
        perimetro = 30;
    }
    public void imprimirInformacion()
    {
        System.out.println("--------Datos de la figura geométrica-------");
        System.out.println("nombre = " + nombre);
        System.out.println("área = " + area);
        System.out.println("perímetro = " + perimetro);
    }
}
```

En la clase Figura deberá implementar el cálculo del área y perímetro dependiendo del número de puntos que requiera. Para ello debe considerar la fórmula para calcular la distancia entre puntos.

Sea los puntos P1 con coordenadas x1, y1 y P2 con coordenadas x2, y2

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

Para calcular el área de cualquier triángulo considerando que solo conoce la medida de sus lados.

El área de un triángulo ABC, cuyos lados miden ***a***, ***b*** y ***c*** unidades es:

$$Área = \sqrt{p(p-a)(p-b)(p-c)}$$

siendo

$$p = semiperímetro = \frac{a+b+c}{2}$$

2. SIMULACIÓN DE CAJERO AUTOMÁTICO

Defina la clase Cuenta como sigue:

Cuenta
- numero : long - nip : String - saldo : double - interesAnual: double - titular : String
+ Cuenta() + Cuenta(num:int, ni:String, sal:double, interes:double, tit:String) + getNumero() : int + setNumero(num:int) : void + getNip() : String + setNip(ni:String) : void + getSaldo() : double + setSaldo(sal : double) : void + getInteres() : double + setInteres(interes : double) : void + getCliente() : String + setCliente(cli: String) : void + retirarSaldo(double cant) : boolean + consultarSaldo( ) : double + cambiarNip(String nuevoNip) : boolean + transferenciaCuentas(Cuenta destino) : boolean + imprimirInformacion( ) : void

Implementación de funciones:

public boolean **retirarSaldo**(double cant). La función retirar saldo debe validar que la cantidad ingresada sea un monto correcto, es decir que no sea un número negativo y que la cantidad sea menor al saldo de la cuenta.

public double **consultarSaldo**(). La función consultar saldo imprime en consola el saldo de la cuenta

public boolean **cambiarNip**(String nuevoNip). La función cambiar nip debe verificar que la nueva cadena contenga 4 caracteres numéricos y no repetidos entre sí.

public boolean **transferenciaCuentas**(Cuenta destino). La función transferencia entre cuentas deberá afectar el saldo tanto de la cuenta origen como la cuenta destino

public void **imprimirInformacion**( ). Imprime en consola los datos de la cuenta

Requerimientos:

- Cree un arreglo de 10 objetos del tipo cuenta con diferentes valores de atributos.
- Imprima en consola los datos de cada cuenta
- Realice ejemplos que muestren el funcionamiento de los 4 métodos.

Conteste ampliamente las siguientes preguntas:

1. ¿Cuál es la diferencia entre el constructor por omisión y el constructor por parámetros?
2. ¿Qué ocurre si cambia el modificador de acceso de un constructor de public a private?
3. Cambie el modificador de acceso del método consultarSaldo a protected y private indique qué ocurre en cada caso.
4. Describa los pasos para crear un arreglo de objetos
5. Investigue 5 métodos de la clase String en Java e indique para que funcionan con ejemplos.

Entregar individualmente un reporte y el código fuente de las clases implementadas.

Requerimientos del Reporte.

El reporte deberá contemplar los siguientes aspectos:

- Objetivos
- Marco teórico
- Diagrama de clases (debe usar un programa especial para generar el diagrama de clases, por ejemplo: yEd)
- Implementación de las funciones principales (**No** debe copiar todo el código que implementó)
- Pruebas. Pantallas que prueben el funcionamiento de su problema
- Respuestas al cuestionario
- Conclusiones

El reporte deberá ser enviado en formato PDF (no debe exceder 5 cuartillas) y las clases implementadas (únicamente los archivos .java).

La calificación de la práctica consta de:

- **Revisión previa en laboratorio**
- **Reporte e implementación final**

Las prácticas solo podrán ser enviadas 3 días después a la fecha señalada. Cada día de retardo será penalizado con un punto menos sobre la calificación obtenida.

Fecha de entrega. Lunes 17 de febrero de 2020 a las 10 de la noche