	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Guadalupe Lizeth Parrales Romay

Asignatura: Programación Orientada a Objetos

Grupo: 4

No. De práctica(s): 4

Integrante(s): Miguel Adán Vargas López

No. de Equipo de cómputo empleado: Ninguno
Semestre:2021-2
Fecha de entrega:10/10/21

Observaciones: _____

CALIFICACIÓN: _____

Introducción

Para esta practica se busca empezar a trabajar con el modelado de objetos y desarrollar la habilidad de abstracción, uno de los pilares de la programación orientada a objetos, ya que para desarrollar los ejercicios se necesita empezar con el modelaje de una clase que debe tener ciertos parámetros y métodos definidos.

Desarrollo de práctica

Ejercicio 1

Código fuente comentado:

El programa comienza en la clase publica con el nombre del archivo, en ella se tienen dos variables para que el usuario introduzca el punto base para el que la recta se va a definir, en base a esto más adelante le pedirá al usuario un segundo punto para obtener la pendiente y así la ordenada al origen. Al tener los valores x y y del primer punto(x,y), se le pasa al constructor de la clase con la que vamos a estar trabajando.

```java
/*
 * @author Adan Vargas
*/
import java.util.Scanner;

//clase para probar clase RectXY
public class ej1{
    public static void main(String[] s_2){
        int xInicial, yInicial;

        //lectura punto base
        Scanner valor = new Scanner(System.in);
        System.out.println("Ingrese el valor de x de un punto base para la recta.\n");
        xInicial = valor.nextInt();
        System.out.println("Ingrese el valor de y de un punto base para la recta.\n");
        yInicial = valor.nextInt();
        RectXY recta = new RectXY(xInicial,yInicial);
    }
}

```

Para moldear la clase RectXY, vamos a definir dos variables de tipo flotante, que representaran la pendiente(m) y la ordenada al origen(b) de una recta, pero dichos valores deberán ser calculados en razón a lo que el usuario nos indique, por ello en el constructor, al tener los valores del punto inicial(x,y) introducido en el main, empezamos por obtener la pendiente. Al tener la pendiente por una función definida después, podemos imprimir la ecuación de la recta en base a dos puntos (el segundo es introducido en el método obtenerPendiente()), después de esto podemos preguntarle al usuario si desea obtener los puntos de la recta en base a un rango de x1 a x2, si el usuario contesta ‘s’ se le pide que introduzca los valores extremos del rango(x1 y x2) y se llama a la función ImprimirValoresRectaEnIntervalo();.

```
21  class RectXY{
22      float m, b;
23
24      public RectXY(int x, int y){
25          System.out.printf("Recta creada con el punto [%d,%d]\n",x,y);
26          m = ObtenerPendiente(x,y);
27          ImprimirEcuacion(m,b);
28
29          char workOnProcess;
30          Scanner valor = new Scanner(System.in);
31          System.out.println("¿Desea obtener algunos puntos de la recta en cierto rango?[s/n]\n");
32          workOnProcess = valor.next().charAt(0);
33
34          while(workOnProcess == 's'){
35              float x1, x2;
36              System.out.println("Ingrese el valor de x1 del rango x1 a x2\n");
37              x1 = valor.nextFloat();
38              System.out.println("Ingrese el valor de x2 del rango x1 a x2\n");
39              x2 = valor.nextFloat();
40              ImprimirValoresRectaEnIntervalo(x1,x2,b,m);
41              workOnProcess = 'n';
42          }
43          System.out.println("\nFin del programa\n");
44      }
45
```

Estas dos funciones meramente buscan imprimir en consola, sin embargo también tienen ciertos comportamientos, por ejemplo, la función ImprimirEcuacion, que es para imprimir la consola en términos de y = mx + b, hace una comparación, si b es mayor a 0, debe imprimir consigo un signo de más, ya que al imprimir el valor de b, no se reflejara el signo en consola, pero si es negativo, si se reflejara en consola el signo negativo.

```
46      public void ImprimirEcuacion(float m, float b){
47          System.out.printf("y = %fx ",m);
48          if(b > 0)
49              System.out.printf("+ %f \n",b);
50          else
51              System.out.printf("%f \n",b);
52      }
53
54      public void ImprimirValoresRectaEnIntervalo(float x1, float x2, float b, float m){
55          System.out.printf("Los puntos entre los intervalos de x: %f y %f\n",x1,x2);
56          for(float i = x1; i <= x2; i++){
57              System.out.printf("[%f,%f]\n",i,((m*i)+b));
58          }
59      }
60
```

Para obtener la pendiente, se toma el punto inicial y se le pide al usuario que introduzca otro, para poder realizar m = y2 – y1 / x2 – x1, después de esto podemos sustituir la ordenada al origen de y = mx + b en b = y – mx, y así ya tenemos como representar la ecuación.

```
61      public float ObtenerPendiente(float _x1, float _y1){
62          float pendiente;
63          float x1,x2,y1,y2;
64          Scanner valor = new Scanner(System.in);
65          //lectura primer punto
66          x1 = _x1;
67          y1 = _y1;
68          System.out.println("En base al primer punto con el que se creo la recta\n Favor de introducir otro punt
69          //lectura segundo punto
70          System.out.println("Introduzca el valor de x del segundo punto: \n");
71          x2 = valor.nextFloat();
72          System.out.println("Introduzca el valor de y del segundo punto: \n");
73          y2 = valor.nextFloat();
74          pendiente = (y2 - y1)/(x2 - x1);
75          b = ObtenerOrdenada(x1,y1,pendiente);
76          return pendiente;
77      }
78
79      public float ObtenerOrdenada(float x, float y, float m){
80          float ordenada;
81          ordenada = y - (m*x);
82          return ordenada;
83      }
84  }
85
```

Programa ejecutado en consola:

```
C:\Users\adan_\Desktop\Tester\Java\Practica4>java ej1.java
Ingrese el valor de x de un punto base para la recta.

2
Ingrese el valor de y de un punto base para la recta.

4
Recta creada con el punto [2,4]
En base al primer punto con el que se creo la recta
 Favor de introducir otro punto.
Introduzca el valor de x del segundo punto:

1
Introduzca el valor de y del segundo punto:

7
y = 10.000000x -3.000000
Â¿Desea obtener algunos puntos de la recta en cierto rango?[s/n]

n

Fin del programa
```

```
C:\Users\adan_\Desktop\Tester\Java\Practica4>java ej1.java
Ingrese el valor de x de un punto base para la recta.

1
Ingrese el valor de y de un punto base para la recta.

1
Recta creada con el punto [1,1]
En base al primer punto con el que se creo la recta
 Favor de introducir otro punto.
Introduzca el valor de x del segundo punto:

3
Introduzca el valor de y del segundo punto:

9
y = 4.000000x -3.000000
Â¿Desea obtener algunos puntos de la recta en cierto rango?[s/n]

s
Ingrese el valor de x1 del rango x1 a x2

1
Ingrese el valor de x2 del rango x1 a x2

5
Los puntos entre los intervalos de x: 1.000000 y 5.000000
[1.000000,1.000000]
[2.000000,5.000000]
[3.000000,9.000000]
[4.000000,13.000000]
[5.000000,17.000000]
```

Ejercicio 2

Código fuente comentado:

Para este programa primero se le pregunta al usuario si busca instanciar un objeto de la clase Empleado y si contesta ‘s’ entra en un ciclo while donde se le pedirán unos datos definidos en la clase y por ultimo se llama a una función de dicha clase para imprimir en consola los datos.

```java
/*
 * @author Adan Vargas
*/
import java.util.Scanner;

//clase para probar clase Empleado
public class ej2{

    public static void main(String[] s_2){
        char workOnProcess;
        float horasTrabajo;
        float sueldoPorHora;
        String nombre;
        Scanner valor = new Scanner(System.in);
        System.out.println("¿Desea instanciar un empleado?[s/n]\n");
        workOnProcess = valor.next().charAt(0);

        while(workOnProcess == 's'){
            System.out.println("Ingrese el nombre del empleado: \n");
            nombre = System.console().readLine();
            System.out.println("Ingrese el sueldo por hora del empleado \n");
            sueldoPorHora = valor.nextInt();
            System.out.println("Ingrese las horas de trabajo a la semana del empleado \n");
            horasTrabajo = valor.nextFloat();
            Empleado empleado = new Empleado(horasTrabajo,sueldoPorHora,nombre);
            empleado.ImprimirInformacion();
            workOnProcess = 'n';
        }
        System.out.println("\nFin del programa\n");
    }
}
```

La clase empleado tiene contemplados algunos valores generales de un empleado, como su salario, su nombre, sus horas de trabajo y cuanto cobra por la hora, estos son definidos en el main por el usuario del programa y son relacionados en el constructor con los parámetros de la clase a excepción del salario, que es un valor definido por horasTrabajo multiplicadas por el sueldoPorHora y estos son impresos en consolas por el método de la clase ImprimirInformacion().

```
33  class Empleado{
34      private float salario;
35      private float horasTrabajo;
36      private float sueldoPorHora;
37      private String nombre;
38
39      public Empleado(float _horasTrabajo, float _sueldoPorHora, String _nombre){
40          horasTrabajo = _horasTrabajo;
41          sueldoPorHora = _sueldoPorHora;
42          nombre = _nombre;
43          salario = horasTrabajo*sueldoPorHora;
44      }
45      public void ImprimirInformacion(){
46          System.out.printf("Horas de trabajo de %s: %f\n",nombre, horasTrabajo);
47          System.out.printf("Sueldo por hora de trabajo de %s: %f\n",nombre, sueldoPorHora);
48          System.out.printf("Salario de %s: %f",nombre, salario);
49      }
50  }
```

Programa ejecutado en consola:

```
C:\Users\adan_\Desktop\Tester\Java\Practica4>java ej2.java
Â¿Desea instanciar un empleado?[s/n]

s
Ingrese el nombre del empleado:

adan
Ingrese el sueldo por hora del empleado

100
Ingrese las horas de trabajo a la semana del empleado

50
Horas de trabajo de adan: 50.000000
Sueldo por hora de trabajo de adan: 100.000000
Salario de adan: 5000.000000
Fin del programa
```

Ejercicio 3

Código fuente comentado:

Para este programa primero se le pregunta al usuario si busca instanciar un objeto de la clase Vehiculo y si contesta ‘s’ entra en un ciclo While donde se le pedirán unos datos definidos en la clase como, por ejemplo, la marca, el tipo de vehículo, color, etc. y por último se llama a una función de dicha clase para imprimir en consola los datos.

```java
/*
 * @author Adan Vargas
*/
import java.util.Scanner;

//clase para probar clase Vehiculo
public class ej3{

    public static void main(String[] s_2){
        char workOnProcess;
        String tipoVehiculo;
        String marca;
        int anoModelo;
        float precio;
        String color;
        Scanner valor = new Scanner(System.in);
        System.out.println("¿Desea instanciar un vehiculo?[s/n]\n");
        workOnProcess = valor.next().charAt(0);

        while(workOnProcess == 's'){

            System.out.println("Ingrese el tipo de vehiculo[auto, bicicleta, motocicleta...]\n");
            tipoVehiculo = System.console().readLine();
            System.out.println("Ingrese la marca del vehiculo\n");
            marca = System.console().readLine();
            System.out.println("Ingrese el año del modelo del vehiculo\n");
            anoModelo = valor.nextInt();
            System.out.println("Ingrese el precio que tiene el vehiculo en mercado\n");
            precio = valor.nextFloat();
            System.out.println("Ingrese el color del vehiculo\n");
            color = System.console().readLine();
            Vehiculo carro = new Vehiculo(marca,tipoVehiculo,color,anoModelo,precio);
            carro.ImprimirInformacion();
            workOnProcess = 'n';
        }
```

La clase Vehiculo tiene contemplados algunos valores generales de un vehículo terrestre, como su marca, el tipo de automóvil que es, su color y cuánto cuesta actualmente adquirirlo, estos son definidos en el Main por el usuario del programa y son relacionados en el constructor con los parámetros de la clase y estos son impresos en consolas por el método de la clase ImprimirInformacion().

```
                color = System.console().readLine();
                Vehiculo carro = new Vehiculo(marca,tipoVehiculo,color,anoModelo,precio);
                carro.ImprimirInformacion();
                workOnProcess = 'n';
            }
            System.out.println("\nFin del programa\n");
        }

    }

    class Vehiculo{
        private String tipoVehiculo;
        private String marca;
        private int anoModelo;
        private float precio;
        private String color;

        public Vehiculo(String _marca, String _tipoVehiculo, String _color, int _anoModelo, float _precio){
            marca = _marca;
            anoModelo = _anoModelo;
            precio = _precio;
            color = _color;
            tipoVehiculo = _tipoVehiculo;
        }

        public void ImprimirInformacion(){
            System.out.printf("\nEl vehiculo es : %s \n",tipoVehiculo);
            System.out.printf("La marca del vehiculo es : %s \n",marca);
            System.out.printf("La vehiculo es un modelo de: %d \n",anoModelo);
            System.out.printf("El precio actual en mercado del vehiculo : %f \n",precio);
            System.out.printf("El color del vehiculo es : %s \n",color);
        }
    }

```

Programa ejecutado en consola:

```
C:\Users\adan_\Desktop\Tester\Java\Practica4>java ej3.java
Â¿Desea instanciar un vehiculo?[s/n]

s
Ingrese el tipo de vehiculo[auto, bicicleta, motocicleta...]

auto
Ingrese la marca del vehiculo

ford
Ingrese el aÃ±o del modelo del vehiculo

2018
Ingrese el precio que tiene el vehiculo en mercado

80000
Ingrese el color del vehiculo

negro

El vehiculo es : auto
La marca del vehiculo es : ford
La vehiculo es un modelo de: 2018
El precio actual en mercado del vehiculo : 80000.000000
El color del vehiculo es : negro

Fin del programa
```

Conclusiones

La realización de esta practica deja demostrada la importancia de la planeación que debe haber antes de realizar un programa, ya que en este caso que trabajamos con programación orientada a objetos tenemos que tomar en cuenta a los 4 pilares y al inicio es clave la abstracción y tener en cuenta como trabajaremos la encapsulación y esto debe quedar bien definido antes de empezar a escribir código como buena práctica, debido a que puede resultar en confusiones, problemas que pudieron ser evitados y sobre todo para tener bien claro con que métodos y variables vamos a contar y cual será el flujo de nuestro programa para que sea lo más eficiente posible.