TEORÍA DE REPASO UD 3 ORIENTACIÓN A OBJETOS I

ORIENTACIÓN A OBJETOS

Veamos un resumen de todo lo estudiado en clase con un ejemplo:

Se desea llevar un control del estado de una cuenta corriente; la cuenta corriente está caracterizada por su saldo y sobre ella se pueden realizar tres tipos de operaciones:

- saldo: devuelve el saldo de la cuenta (puede ser negativo, hasta -1000 €).
- Imposición (cantidad): ingresa en la cuenta una cantidad de dinero dada por el usuario.
- Reintegro (cantidad): saca de la cuenta una determinada cantidad de dinero especificada por el usuario.

Supón que la cuenta inicialmente tiene un saldo de cero. Escribe una clase CuentaCorriente que implemente la funcionalidad descrita.

NOTA: NO COPIES NI PEGUES EL CÓDIGO, INTENTA HACERLO TÚ ANTES MIENTRAS LO VAS LEYENDO DETENIDAMENTE

1) De momento, lo primero que tenemos que plantearnos es ¿cuántas clases me hacen falta? En este caso serán solo dos, una clase CuentaCorriente y otra principal a la que llamaremos Prueba para probar las funcionalidades del programa.

2) Crearemos la clase CuentaCorriente (no principal, luego no lleva main) donde escribiremos todo lo relacionado con una cuenta, es decir, los atributos que la caracterizan, los constructores para los objetos de este tipo y los métodos para realizar las operaciones en dicha cuenta. Todo esto, no es más que una especie de manual sobre cómo se hacen las cosas. Por este orden escribimos, los atributos, constructores, getters y setters (métodos para obtener o acceder a los atributos) y métodos:

```
public class CuentaCorriente {

        //Atributos

        private double saldo;

        private String nombreTitular;

        //Constructor con parámetros

        public CuentaCorriente (double saldo, String nombre){

                this.saldo = saldo;

                this.nombreTitular = nombre;

        }
```

```
//Constructor vacío
public CuentaCorriente (){
}
//Getters & Setters. Se pueden autogenerar con eclipse
public double getSaldo() {
        return saldo;
}
public void setSaldo(double saldo) {
        this.saldo = saldo;
}
public String getNombreTitular() {
        return nombreTitular;
}
public void setNombreTitular(String nombreTitular) {
        this.nombreTitular = nombreTitular;
}
//Métodos propios, los que hace el programador
//Un método para mostrar el menú del programa
public void mostrarMenu (){
        System.out.println("1.-  Ver saldo");
        System.out.println("2.-  Ingresar dinero");
        System.out.println("3.-  Retirar dinero");
}
//Un método para mostrar los datos de un objeto cuentaCorriente, Titular y saldo
public void mostarSaldo(){
        System.out.println (this.nombreTitular+", tu saldo es "+this.saldo+"€.");
}
```

```
        /*Método para ingresar dinero en una cuenta, se le debe pasar el dinero que queremos ingresar y devuelve el nuevo saldo ya actualizado con esa cantidad sumada*/

        public double ingresarSaldo(double ingreso){

                this.saldo = this.saldo + ingreso;

                return this.saldo;

        }

/*Método para retirar dinero, comprueba que la cantidad que se quiere retirar no es mayor que el saldo con el if, pues no se puede retirar más de lo que se tiene. Se le pasa lo que se quiere retirar y devuelve el saldo con la cantidad que se lleva el usuario ya restada*/

        public double retirarSaldo (double retirar){

                if (retirar > this.saldo) {

                        System.out.println("No puedes retirar más dinero del "

                        + "que posees en la cuenta. Lo sentimos
                        "+this.nombreTitular+".");

                        return this.saldo;

                }else{

                        this.saldo = this.saldo - retirar;

                        return this.saldo;

                }

        }

}
```

3) Todo esto es lo que se puede hacer y cómo se hace con nuestra cuenta corriente. Todavía no hemos hecho nada. Para hacer objetos y realizar acciones con ellos, debemos crear la clase principal, crear los objetos correspondientes y hacer llamadas a estos métodos que acabamos de definir. Veamos la clase principal:

```
import utilidades.Leer;

public class principal {

        public static void main(String[] args) {
                /*Declaramos las variables que nos van a hacer falta para lecturas por teclado
                o guardar resultados devueltos por los métodos que llamemos*/
```

```
double saldo = 0.0, retirar = 0.0, ingresar = 0.0;
int opcion = 0;
String nombre = null;

/*Damos la bienvenida y pedimos los primeros datos para crear el objeto
"CuentaCorriente"*/

System.out.println("BIENVENIDOS");
System.out.println("GRACIAS POR CONFIAR EN NUESTRO BANCO");
System.out.println("¿CÚAL ES SU NOMBRE?");
nombre = Leer.dato();
System.out.println("¿CON QUÉ SALDO ABRE SU CUENTA?");
saldo = Leer.datoDouble();

/*Una vez que tenemos los datos leídos y guardados en variables (nombre y
saldo), creamos la cuentaCorriente. En este caso se llamará cuenta1*/

CuentaCorriente cuenta1 = new CuentaCorriente(saldo, nombre);

do {
        /*Llamamos al método definido en la clase CuentaCorriente
mostrarMenu para quemuestre las posibilidades que hay en nuestro programa (puedes ver la
clase cuentaCorriente para observar qué hace este método)*/

        cuenta1.mostrarMenu();
        opcion = Leer.datoInt();//Se lee la opción para entrar en el switch

/*En el switch, basta con ir llamando a los métodos adecuados en cada caso según la opción
que haya elegido el usuario. Recuerda que para llamar a un método basta con poner el
nombre del objeto, un punto y el nombre del método, dando los parámetros entre paréntesis
que este necesita según lo hayamos declarado en la clase cuentaCorriente, por ejemplo, en el
case 1, a mostrarSaldo no se le pasa nada ni devuelve nada*/
        switch (opcion) {
        case 1:
                cuenta1.mostarSaldo(); //Llamada a mostrarDatos
                break;
        case 2:
/*Para ingresar dinero primero debemos pedir cuánto se quiere ingresar, se lee esta cantidad
por el teclado, se guarda en una variable llamada ingresar y esta variable se le pasa como
parámetro (entre los paréntesis) al método que hace el ingreso, en nuestro caso llamado
ingresarSaldo */

//Pedimos la cantidad
                System.out.println(cuenta1.getNombreTitular()+",      ¿Cuánto
deseas ingresar?");
//Leemos la cantidad
                ingresar = Leer.datoDouble();
//Hacemos la llamada pasando la cantidad ymostramos por pantalla el nuevo saldo
                System.out.println("Tu nuevo saldo es: "
                        + cuenta1.ingresarSaldo(ingresar) + "€");

                break;
```

```
                        case 3:
//Análogo al case 2
                                System.out.println(cuenta1.getNombreTitular()+",      ¿Cuánto
deseas retirar?");
                                retirar = Leer.datoDouble();
                                System.out.println("Tu nuevo saldo es: "
                                                + cuenta1.retirarSaldo(retirar) + "€");
                                break;

                        default:
                                System.out.println("No  has  elegido  una  opción  correcta
"+cuenta1.getNombreTitular()+".");
                                break;
                        }
//Código para repetir el programa, do-while
                        System.out.println("¿Quieres realizar otra operación?  1.-SI   0.-NO");
                        opcion = Leer.datoInt();
                } while (opcion == 1);

                System.out.println("Hasta la próxima");

        }

}
```

ATRIBUTOS DE TIPO NO BÁSICO. COMPOSICIÓN DE CLASES

Hay dos formas de reutilizar el código, mediante la composición y mediante la herencia. La herencia la veremos más adelante. La composición significa utilizar objetos dentro de otros objetos, es decir, que los atributos de un objetos, no sean básicos (int, double, etc.) sino que sean del tipo de otra clase definida por el programador. En el siguiente ejemplo, uno de los atributos de la clase Rectángulo, es del tipo Punto, que es otra clase del programa.

Vamos a estudiar la clase *Rectangulo* definiendo el origen, no como un par de coordenadas *x* e *y* (números enteros) sino como objetos de una nueva clase denominada *Punto*.

La clase *Punto*

La clase *Punto* tiene dos miembros dato, la abscisa *x* y la ordenada *y* de un punto del plano. Definimos dos constructores uno por defecto que sitúa el punto en el origen, y otro constructor explícito que proporciona las coordenadas *x* e *y* de un punto concreto.

```
public class Punto {
   int x;
```

```
   int y;
//funciones miembro, aquí van los constructores y los métodos de esta clase, por ejemplo, mostrarDatos ().
}
```

El constructor explícito de la clase *Punto* podemos escribirlo de dos formas

```
public Punto(int x1, int y1) {
    x = x1;
            y = y1;
}
```

Cuando el nombre de los parámetros es el mismo que el nombre de los miembros datos escribimos

```
public Punto(int x, int y) {
    this.x = x;
            this.y = y;
}
```

this.x que está a la izquierda y que recibe el dato *x* que se le pasa al constructor se refiere al miembro dato, mientras que *x* que está a la derecha es el parámetro. **this** es una palabra reservada que guarda una referencia al objeto propio, u objeto actual. Tendremos ocasión a lo largo del curso de encontrar esta palabra en distintas situaciones.

La función miembro *desplazar* simplemente cambia la posición del punto desde *(x, y)* a *(x+dx, y+dy)*. La función *desplazar* cuando es llamada recibe en sus dos parámetros *dx* y *dy* el desplazamiento del punto y actualiza las coordenadas *x* e *y* del punto. La función no retorna ningún valor

```
public void desplazar(int dx, int dy){
            x=x+dx;
            y=y+dy;
}
```

Para crear un objeto de la clase *Punto* cuyas coordenadas *x* e *y* valgan respectivamente 10 y 23 escribimos, en el main:

```
import Leer;

public class principal {

        public static void main(String[] args) {

                /*Aquí se puede hacer la lectura por teclado de los dos números, 10 y 23 para que sea el usuario quien los facilite al programa. Se deben guardar en dos variables normales definidas aquí mismo en el main y luego pasarlas como parámetro al constructor del punto. Quedaría así:

int coorx, coordy;

System.out.println("Introduzca la posición x");
```

```
coordx=Leer.datoInt ( );

coordy=Leer.datoInt ( );

Punto p1= new Punto (coorx, coordy);

*/

                    Punto p=new Punto(10, 23);

                    p.desplazar (-10, 40);

          }
```

Para desplazar el punto *p* 10 unidades hacia la izquierda y 40 hacia abajo, llamamos desde el objeto *p* a la función *desplazar* y le pasamos el desplazamiento horizontal y vertical.

```
p.desplazar(-10, 40);
```

El código completo de la clase *Punto*, es el siguiente

```
public class Punto {
   int x = 0;
   int y = 0;
 public Punto(int x, int y) {
      this.x = x;
            this.y = y;
 }
//Esto es otro tipo de constructor vacío, pone la x e y del punto a cero
 public Punto() {
            x=0;
            y=0;
 }
 void desplazar(int dx, int dy){
            x=x+dx;
            y=y+dy;
 }
}
```

La clase *Rectángulo*

La clase *Rectangulo* tiene como miembros dato, el *origen* que es un objeto de la clase *Punto* y las dimensiones *ancho* y *alto*.

```
public class Rectangulo {
   Punto origen;
   int ancho ;
   int alto ;
//Constructores y métodos
}
```

El constructor por defecto, crea un rectángulo situado en el punto 0,0 y con dimensiones nulas:

```
public Rectangulo() {
        origen = new Punto(0, 0);
        ancho=0;
        alto=0;
}
```

El constructor explícito crea un rectángulo situado en un determinado punto *p* y con unas dimensiones que se le pasan en el constructor

```
public Rectangulo(Punto p, int w, int h) {
        origen = p;
        ancho = w;
        alto = h;
}
}
```

Para desplazar un rectángulo, trasladamos su origen (esquina superior izquierda) a otra posición, sin cambiar su anchura o altura. Desde el objeto *origen*, llamamos a la función *desplazar* miembro de la clase *Punto:*

Este método siguiente desplazar, está en la clase Rectángulo, y dentro de él, se llama al método desplazar de la clase Punto.

```
void desplazar(int dx, int dy) {
        origen.desplazar(dx, dy);
}
```

El código completo de la nueva clase *Rectangulo*, es el siguiente.

```
public class Rectangulo {
    Punto origen;
    int ancho ;
    int alto ;

public Rectangulo(int w, int h) {
        origen = new Punto(0, 0);
        ancho=w;
        alto=h;
}
public Rectangulo(Punto p, int w, int h) {
        origen = p;
        ancho = w;
        alto = h;
}
void desplazar(int dx, int dy) {
        origen.desplazar(dx, dy);
}
int calcularArea() {
        return ancho * alto;
}
```

```
}
```

Objetos de la clase *Rectangulo*

Para crear un rectángulo *rect1 en el main*, situado en el punto de coordenadas 44, 70 y de dimensiones 100 y 200 escribimos:

```
Punto p=new Punto(44, 70);
        Rectangulo rect1=new Rectangulo(p, 100, 200);
```

O bien, en una sola línea:

```
Rectangulo rect1=new Rectangulo(new Punto(44, 70), 100, 200);
```

Para desplazar el rectángulo *rect1* desde el punto (100, 200) a otro punto situado 40 unidades hacia la derecha y 20 hacia abajo, sin modificar sus dimensiones, escribimos

```
rect1.desplazar(40, 20);
```

Para hallar y mostrar el área del rectángulo *rect1* podemos escribir

```
System.out.println("el área es "+rect1.calcularArea());
```

```
public class RectanguloApp {

  public static void main(String[] args) {

    Punto p=new Punto(44, 70);
        Rectangulo rect3=new Rectangulo(p, 100, 200);
     //O con una rectángulo vacío
     Rectangulo rect4=new Rectangulo();
     rect3.desplazar(40, 20);
     System.out.println("el área es "+rect3.calcularArea());
     //O guardar el valor devuelto por calcularArea en una variable y mostrar la variable. Dicha variable debe estar declarada en el main
     int areaRect=rect3.calcularArea ( );
     System.out.println("el área es "+areaRect);
     }
  }
}
```