Boletín 7 – Herencia

Ejercicio 1:

Dentro de un sistema bancario que ya está funcionando disponemos de una clase CUENTA, los datos que guarda dicha clase son los siguientes:

- String Titular; Nombre del dueño de la cuenta
- double Saldo: Dinero disponible en la cuenta

Además una cuenta tiene los siguientes métodos:

- Los constructores correspondientes
- Los métodos de modificación y de devolución
- Un método *reintegro,* que recibe una cantidad y lo saca de la cuenta (resta del saldo),
- Un método *ingreso*, que recibe una cantidad y lo mete en la cuenta (suma al saldo).
- Un método toString, para poder escribir los datos de la cuenta en pantalla.

La nueva política del banco exige que tengamos otros tipos de cuentas con características diferentes. En particular debemos admitir los siguientes tipos:

- Cuenta de crédito: Permite obtener reintegros hasta un saldo negativo fijo para cada cuenta y, si el saldo es negativo el banco le cobra un 1% más.
- Cuenta de nómina: Permite obtener reintegros hasta un saldo negativo igual al importe mensual de la nómina que el cliente tiene domiciliada en dicha cuenta.
- Cuenta de ahorro: No admite saldo negativo. Cada vez que se ingresa una cantidad se suma el 0,1%.
- Cuenta joven: Exige que el cliente sea menor de 25 años.

Crea un programa principal donde definas una cuenta de cada clase, con un saldo inicial de 100 €, realiza una operación de ingreso de 50 € y un reintegro de 200 €, muestra los saldos de cada cuenta.

Ejercicio 2:

Tenemos la siguiente clase EMPLEADO definida del siguiente modo, de cada empleado guardamos la siguiente información:

- String nombre
- int edad
- string nif

Los métodos de la clase son los siguientes:

- Los constructores correspondientes
- Los métodos de modificación y de devolución
- Un método toString, para poder escribir los datos de la cuenta en pantalla.

Se nos pide que creemos nuevas clases partiendo de la anterior:

- Empleado temporal, del que nos interesa saber la fecha de alta y de baja en la empresa.
- Empleado por horas. Nos interesa el precio de la hora trabajada, y el número de horas que ha trabajado este mes.
- Empleado fijo. Debemos añadir a la información que almacenamos sobre él el año de alta en la empresa.

Además debemos añadir a todos los empleados la funcionalidad de cálculo del sueldo con las siguientes consideraciones:

- En los empleados temporales el sueldo mensual es fijo.
- En los empleados fijos el sueldo es el resultado de sumarle a la base un complemento anual fijo multiplicado por el número de años en la empresa.
- En los empleados por horas el sueldo se calcula multiplicando su sueldo por hora por el número de horas de este mes.

Crea un programa principal, donde definas un empleado de cada tipo y muestra el sueldo que recibe cada uno de ellos.

Ejercicio 3.1:

Desarrollar una clase Producto cuyos datos miembro sean una cadena de texto identificación y un valor real (double) precioBase. Ambos datos de tipo privado. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Producto que recibe como argumentos una cadena de texto identificación y un valor real (double) precioBase, y construye un nuevo objeto de la clase Producto cuya identificación y precio base vienen dados respectivamente por los argumentos identificación y precioBase.
- Los métodos de acceso getIdentificacion y getPrecioBase, sin argumentos, que respectivamente devuelven el identificador y el precioBase de un objeto Producto.
- El método modificador setIdentificacion, que recibe como argumento una cadena de texto identificación y cambia el campo identificación del objeto sobre el que se aplica, asignándole como nuevo valor el del argumento identificación.
- El método modificador setPrecioBase, que recibe como argumento un valor real (double) precioBase y cambia el campo precioBase del objeto sobre el que se aplica, asignándole como nuevo valor el del argumento precioBase.
- El método de impresión en pantalla toString, sin argumentos, tal que al ser evaluado sobre un objeto de tipo Producto cuya identificación es “RJ45” y cuyo precio base es 10.50, genere la siguiente cadena: RJ45 (10.5)

Ejercicio 3.2:

Desarrollar una clase ProductoInventariado derivada de la clase Producto anterior, con dos campo enteros adicionales cantidad y beneficio. Ambos de tipo privado. Para esta clase se piden los siguientes constructores y métodos:

- El constructor ProductoInventariado, que recibe como argumentos una cadena de texto identificación, un valor real (double) precioBase y dos datos enteros cantidad y beneficio, y construye un nuevo objeto de la clase Producto utilizando el constructor de la clase base (super(...)), cuya identificación, precio base, cantidad y beneficio vienen dados respectivamente por los argumentos identificación, precioBase, cantidad y beneficio.
- Los métodos de acceso getCantidad y getBeneficio, sin argumentos, que respectivamente devuelven la cantidad y el beneficio de un objeto ProductoInventariado.
- Los métodos modificadores setCantidad y setBeneficio, que reciben como argumento un valor entero y modifican, respectivamente, el valor del campo cantidad y beneficio del objeto sobre el que se aplican, asignándoles como nuevo valor el del argumento recibido.
- El método precioFinal, sin argumentos, que devuelve el precio final de un objeto de la clase ProductoInventariado determinado como el precio base al que se suma el porcentaje de beneficio.
- El método de impresión en pantalla toString sin argumentos, tal que al ser evaluado sobre un objeto de tipo ProductoInventariado cuya identificación es “RJ45”, cuyo precio base es 10.50, cuya cantidad es 10 y cuyo beneficio es 13, genera la siguiente cadena: 10 RJ45 (10.5) (+13%)

Ejercicio 3.3:

Desarrollar una clase Tienda cuyos datos miembro son una cadena de texto nombre, un vector de objetos del tipo ProductoInventariado donde se almacena información de los productos de la tienda y un valor real (double) caja que almacena el dinero del que dispone la tienda. Para esta clase se piden los siguientes constructores y métodos:

- El constructor Tienda, que recibe como argumentos una cadena de texto nombre y un valor real caja, y construye un nuevo objeto de la clase Tienda cuyo nombre y caja vienen dados respectivamente por los argumentos nombre y caja.
- El método buscaProducto, que recibe como argumento una cadena de texto id, y devuelve el rpoducto inventariado con ese id si lo encuentra.
- El método añadirProducto, que recibe como argumento un identificador de producto id, un precio base p (tipo double), una cantidad c (tipo int) y un beneficio b (tipo int) y lo añade al inventario. Si el producto ya estaba en el inventario entonces sólo hay que modificar los datos relativos al precio base, cantidad y beneficio. Si el producto no está en el inventario entonces hay que añadirlo. En cualquier caso, sólo se podría añadir un producto si el coste total (cantidad * precio base) es menor o igual que el dinero del que dispone la tienda, el cual ha de ser disminuido de manera adecuada. Si en el inventario no hay sitio para el producto o éste no puede ser adquirido por no disponer de suficiente dinero entonces se ha de indicar en pantalla un mensaje informativo.
- El método venderProducto, que recibe como argumento un identificador de producto id y una cantidad c y, si el producto existe en el inventario en una cantidad mayor o igual que c, entonces disminuye en c unidades la cantidad del producto id que hay en el inventario y modifica adecuadamente la caja. Si no hay unidades sufientes del producto id para vender, entonces se ha de indicar en pantalla un mensaje informativo.

Construye un programa principal, donde debes crear una tienda que se llame “Hipercor”, con un ménu que te de las opciones, de añadir producto, buscar producto y vender producto.

Ejercicio 4.1:

Se quiere guardar información de las temperaturas medidas un determinado día. Para ello, vamos a crear una clase TempDia que guarde la fecha de la medición, la temperatura máxima registrada y la temperatura mínima. Esta clase debe tener los siguientes métodos:

- Un constructor por defecto, que inicialice los atributos.
- Un constructor que tome una fecha, un valor de temperatura máxima y de temperatura mínima, y que inicialice los atributos con estos valores.
- Reescribir el método toString, que devuelva la fecha, la temperatura máxima, la mínima y la media (máx. +min. / 2).
- Y todos los métodos que estimes necesario.

Ejercico 4.2:

A partir de la clase anterior se quiere heredar una nueva clase TempHumDia que me permita, además almacenar la humedad máxima, la mínima. Cuando redefinas toString debes tener en cuenta que muestre también la humedad media.

Ejercicio 4.3:

Ahora, y partiendo de la clase anterior vamos a construir una clase EstacionClimatica, donde vamos a guardar el nombre de la estación, el año del que vamos a guardar datos y un vector de objetos de tipo TempHumDia, que corresponden a las mediciones realizadas en cada día del año. Esta clase tendrá los siguientes métodos:

- Un constructor por defecto.
- Un constructor que reciba el nombre y el año.
- Un método AddDia, que recibe una fecha, una temperatura máxima y otra mínima, humedad máxima, humedad mínima y que, si el año de la fecha corresponde con el año de la estación, introduce el nuevo objeto en el vector- Si el año no corresponde con el de la estación dar un mensaje de error.
- Un método DelDia, que reciba una fecha y borre ese día si existen datos.
- Un método ListarDias que te muestre en pantalla los datos de los días que hemos introducidos: fecha, temperatura máxima, mínima y media, humedad máxima, mínima y media.

En el programa principal crea una estación, introduce cinco días pedidas por teclado (si la fecha no es correcta te la tiene que volver a pedir) y lista esos días. Luego borra dos (pedida por teclado y validándola de nuevo) y los vuelve a listar.