Polimorfismo

Cadenas

Manuel Molino Milla

6 de febrero de 2024

Ejercicio 1

Construye una clase *ArrayReales* que declare un atributo de tipo *double[]* y que implemente una interfaz llamada *Estadisticas*. El contenido de esta interfaz es el siguiente:

```
public interface Estadisticas {
   int obtenerNumeroValores(); //número de elementos de una colección
   double obtenerValorMinimo();  //valor mínimo de una colección
   double obtenerValorMaximo(); //valor máximo de una colección
   double calcularSuma(); //suma de los valores de una colección
   double calcularValorMedio(); //valor medio de de una colección
   double calcularDesviacionTipica(); //desviación típica de de una colección.
}
```

Crea en la clase *ArrayReales* el correspondiente constructor y *getter* del atributo.
Crea una clase *Test* que genere un *array* de 10 valores doubles, puede hacerlo de la siguiente manera:

```
double[] numbers =  new Random().doubles(0,350).limit(10).toArray();
```

Con ese *array* puedes inicializar el atributo de la clase *ArrayReales*
Muestra en consola el número de valores, el valor máximo, el valor mínimo, la suma, el valor medio y la desviación típica de los valores de acuerdo a los métodos implementados de la interfaz. Usa *printf* para mostrarlos.
Crea posteriormente un método *default* en la *interfaz* que devuelva como *String* todos los valores anteriores que se muestran en consola.

Cuestiones:

- Esa interfaz ¿Puede ser implementada por una clase que en vez de tener como atributo un *array* de *doubles*, tuviera como atributo una lista (*List*) de *doubles*?
- ¿Y si el atributo fuera de otro tipo, como por ejemplo un *array* de *int*, *long* o *float*?
- ¿Qué ventajas le ves en este caso el uso de *interfaces*?

Ejercicio 2

- Crea un *enum* denominado *TipoTransaccion* con los siguientes valores: *INGRESO y RETIRADA*
- Crea una interface denominada *Banco* que declare (no implemente) los siguientes métodos:
  - Obtener saldo.
  - Ingresar dinero.
  - Sacar dinero.

  Además se implementarán los siguientes métodos privados:
  - Un método que dada una cantidad (será para retirar dinero) que nos diga si se puede realizar dicha transacción. Dicha operación será factible, siempre y cuando, después de retirar el dinero el saldo sea cero o positivo.
  - Un método que nos muestre en consola información sobre la transacción, se le pasará como parámetros el tipo de transación y la cantidad a operar. Ejemplo de salida:

    ```
    Transación: INGRESO
    Cantidad: 50€
    Saldo: 150 €
    ```

  Se implementará también el siguiente método *default*:
  - Método que ejecuta la transacción, como parámetros el tipo de transacción y la cantidad. Se analizará si es un deposito y por tanto se llamará al método que ingresa dinero. Y en el caso de retirar dinero, se comprobará previamente que hay saldo, y si es posible la transacción, se llamará al método que saca dinero. En ambos casos, en retirada y deposito, tras llevar acabo dicha acción, se llamará al método anterior que muestra la información de la transacción.
- Crea una clase denominada *CuentaCorriente* que implemente la interface anterior y que tenga como atributo el saldo inicial.
- Posteriormente, crea un método estático en la interface que se le pase un saldo inicial y retorne un objeto *Banco* inicializando una cuenta corriente.
- Crea una clase *Main* que realice las siguientes acciones:
  - Cree un objeto *Banco* usando el método estático de la interface.
  - Realice un deposito de dinero.
  - Realice un intento de retirada de dinero superior al saldo actual.
  - Realice una retirada de dinero inferior al saldo actual.