CURSO 2018-19.

Práctica 1 Pilas y Colas

OBJETIVOS:

- Utilización de las clases Pila y Cola
- Modificación de las clases Pila y Cola
- Creación de nuevos proyectos utilizando las clases Pila y Cola
- Utilizar las clases de la librería estándar

ENTREGA:

Se deberán entregar dos ficheros .zip cada uno de ellos con un proyecto Java.

Los proyectos Java que se piden son:

Practica1PilasyColas, que tendrá todo lo pedido en los apartados 1, 2, 3, 4 y 5.

Practica1Palindromo, que tendrá todo lo pedido en los apartados 6 y 7.

1. Construir un proyecto utilizando las clases Pila y Cola

Crear un proyecto Java, **Practica1PilasyColas,** introduciendo las clases Pila y Cola que se adjuntan. Añadir una clase Principal con un método *main*, que permita instanciar objetos de estas clases y mandarles mensaje

2. Añadir métodos a la clase Pila

a) Añadir un método a la clase Pila que devuelva la suma de elementos de la pila.

int sumarElementos()

Ejemplos:

p1 p1.sumarElementos() devolverá valor 30

2
1
8
9
10

b) Añadir un método a la clase Pila para llevar a la pila todos los elementos de otra pila, pasada como parámetro del método, en el mismo orden en el que están**.** Esta pila quedará vacía.

void unirPila(Pila p)

Ejemplo:

p1

3
2
1

p2

40
30
20
10

p1.unirPila(p2) dejará:

p1

40
30
20
10
3
2
1

p2 vacia

c) Instanciar dos objetos Pila en la clase Principal y probar los métodos anteriores.

sumarElementos debe probarse al menos con los siguientes valores, apilados en el orden que se indican:

pila1: 8, 1, 4, 2 pila1: 8 y pila1: pila vacía

unirPila deben probarse al menos con las siguientes combinaciones de pilas, donde se apilan los valores en el orden que se indica:

pila1: 2, 4, 8

pila2: 40, 20, 30, 10, pila2: 40 y pila2: pila vacía

3. Escribir funciones fuera de la clase Pila.

Supongamos ahora que no podemos acceder al código de la clase Pila, pero estamos realizando una aplicación que necesita realizar las mismas operaciones que en el caso anterior. Para solucionarlo, realizaremos funciones fuera de la clase Pila.

a) Escribir una función que reciba una pila como parámetro y devuelva un valor con la suma de sus elementos. La pila no debe quedar modificada.

int sumarElementos (Pila p)

b) Escribir en la clase principal una función que reciba dos pilas como parámetros y lleve a la primera de ellas la unión de ambas como en el apartado anterior. La segunda pila quedará vacía.

void unirPilas (Pila p1, Pila p2), dejando la unión en la pila p1.

c) Probar estas funciones en la clase principal con los datos indicados en el apartado anterior.

4. Añadir métodos a la clase Cola

a) Añadir un método a la clase Cola que deje en la cola solo los n primeros elementos, siendo n un valor entero pasado por parámetro. Si la cola tiene menos de n elementos dejará todos.

void dejarN(int n)

Ejemplos:

c1 n=3

8	1	9	4	2

Quedará c1

8	1	9

c1 n=8

8	4	2	9

Quedará c1

8	4	2	9

b) Instanciar un objeto Cola en la clase Principal y probar el método anterior al menos con los siguientes valores:

cola1: 20, 40, 10, 50 n=2, n=4 n=9 n=0

5. Escribir funciones fuera de la clase Cola

a) Supongamos ahora que no podemos acceder al código de la clase Cola, pero estamos realizando una aplicación que necesita dejar en una cola solo sus n primeros elementos. Para solucionarlo, realizaremos una función fuera de la clase Cola, que reciba como parámetros un número entero y una cola, y realice esta operación. Si la cola tiene menos de n elementos dejará todos.

void dejarN(int n, Cola c)

b) Probar esta función en la clase principal, al menos con los datos indicados en el apartado anterior.

6. Comprobador de Palíndromos

Una frase es un palíndromo si la sucesión de caracteres obtenida al recorrerla de izquierda a derecha (ignorando blancos) es la misma que si se recorre de derecha a izquierda.

Ejemplo: ”dabale arroz a la zorra el abad”

Para comprobar si una frase es un palíndromo se realizarán dos versiones, una utilizando dos pilas y otra utilizando una pila y una cola.

Algoritmo utilizando dos pilas:

Llevar todos los caracteres no blancos de la frase a la primera pila.

Desapilar la mitad de los elementos y llevarlos a la segunda pila. Si hay un número impar de elementos, se descarta el elemento central.

Comparar las dos pilas. Si son iguales, la frase es palíndromo.

Algoritmo utilizando una pila y una cola:

Llevar todos los caracteres no blancos de la frase a la pila y a la cola.

Comparar el contenido de la pila y de la cola. Si son iguales, la frase es palíndromo.

Escribir un proyecto Java comprobador de palíndromos. El proyecto se denominará ***Practica1Palindromo***

El proyecto constará de las siguientes clases:

- Clase NodoCaracteres
- Clase PilaCaracteres
- ClaseColaCaracteres
- Clase Comprobador1
- Clase Comprobador2
- Clase Pincipal

- Las clases NodoCaracteres, PilaCaracteres y Cola Caracteres serán similares a las de los apartados anteriores, considerando ahora que deben tratar caracteres en lugar de enteros.
- **Clase Comprobador1** tendrá las siguientes características:
  - **Atributos:**
    - frase, de tipo String
    - pila1 y pila2, de tipo PilaCaracteres

- **Métodos:**

· Constructor sin parámetros, la frase se inicializará con el String vacío y se instanciarán las Pilas con el constructor sin parámetros.

· Constructor con un parámetro de tipo String para inicializar la frase. Las pilas se instanciarán como en el constructor anterior.

· Métodos para comprobar si la frase es palíndromo o no.

· La clase deberá tener al menos los dos siguientes métodos:

public void esPalindromo(),

que comprueba si la frase es palíndromo o no.

Método para leer una frase del teclado y llevarlo al atributo frase.

```
public void leerFrase()  {
    Scanner sc = new Scanner(System.in);
    frase = sc.nextLine();
  }
```

Instanciamos en el método un objeto de la clase Scanner, para utilizarlo a continuación.

A estos métodos se pueden añadir los métodos privados que se consideren.

El método *charAt(i)* de *String* permite acceder al carácter i de un string.

- **Clase Principal (1ª Parte)**

· Instanciar tres objetos *Comprobador1* (cp1,cp2 y cp3) con el constructor con un parámetro, introduciendo en el parámetro los siguientes valores (escribiremos todos los caracteres en minúscula):

· aroma a mora

· camino a casa

· dabale arroz a la zorra el abad

· Comprobar si estas frases son palíndromo.

Ej.: cp1.esPalindromo();

· Instanciar un objeto cp4, con el constructor sin parámetros.

· Mandar a este objeto un mensaje *leerFrase* y a continuación comprobar si la frase leída es palíndromo o no. Se debe probar con las mismas frases del apartado anterior y otras que el alumno elija.

Ej.: cp4.leerFrase();

cp4.esPalindromo();

- **Clase Comprobador2** tendrá las siguientes características:
  - **Atributos:**
    - frase, de tipo String
    - pila1, de tipo PilaCaracteres
    - cola1, de tipo ColaCaracteres

- **Métodos:**

- Constructor sin parámetros, la frase se inicializará con el *String* vacío y se instanciarán la pila y la cola con los constructores sin parámetros.
- Constructor con un parámetro de tipo *String* para inicializar la frase. La pila y la cola se instanciarán como en el constructor anterior.
- Métodos para comprobar si la frase es palíndromo o no.
- La clase deberá tener al menos los dos siguientes métodos:

  public void esPalindromo(),

  que comprueba si la frase es palíndromo o no.

- Método para leer una frase del teclado y llevarlo al atributo frase.

```
public void leerFrase() {
    Scanner sc = new Scanner(System.in);
    frase = sc.nextLine();
}
```

- **Clase Principal (2ª Parte)**

- Instanciar otros tres objetos Comprobador2 (cp21, cp22 y cp23) con el constructor con un parámetro, introduciendo en el parámetro los mismos valores que en las pruebas del apartado anterior.
- Comprobar si estas frases son palíndromo.
- Instanciar un objeto cp24, con el constructor sin parámetros.
- Mandar a este objeto un mensaje *leerFrase* y a continuación comprobar si la frase leída es palíndromo o no.

7. Comprobador de Palíndromos con *Stack*

a) Añadir al proyecto anterior una nueva clase Comprobador3, similar a Comprobador1, utilizando ahora la clase *Stack* de la biblioteca Java pata instanciar las pilas.

b) Realizar las mismas pruebas con objetos de esta clase, que las realizadas con los objetos *Comprobador1*