UNIVERSITAT POLITÈCNICA DE VALÈNCIA

CAMPUS D'ALCOI

Programación

Práctica 3: Estructuras Datos Lineales

Josué Ferri Pascual

jferri@dsic.upv.es

Práctica 3

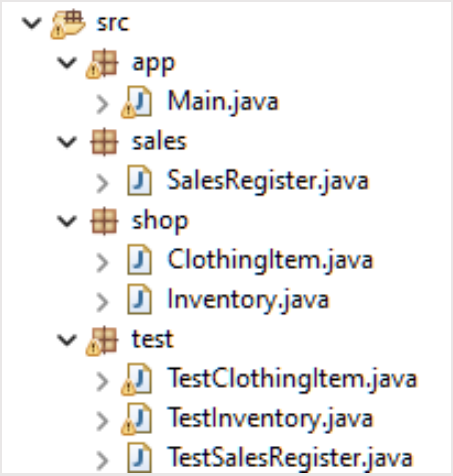
Implementación carrito de la compra

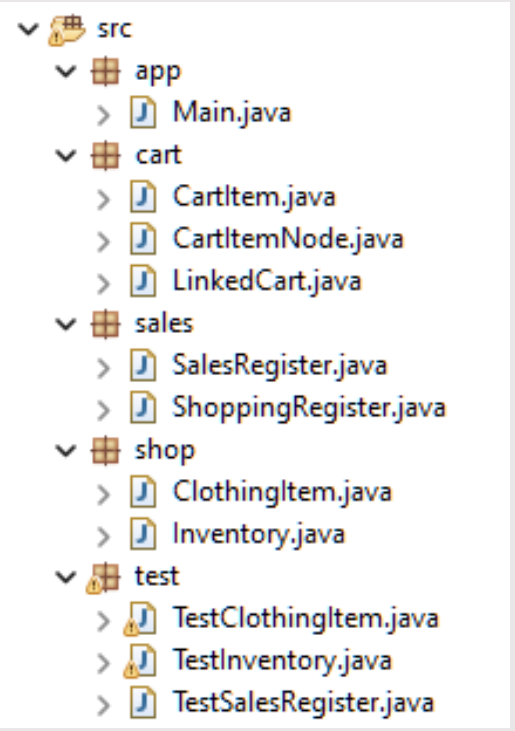
- ➔ Tarea 1 - CartItem
- ➔ Tarea 2 - CartItemNode
- ➔ Tarea 3 - LinkedCart
- ➔ Tarea 4 – ShoppingRegister
- ➔ Tarea 5 - Actualizar Main

Objetivos

➔ Implementar un carrito de la compra

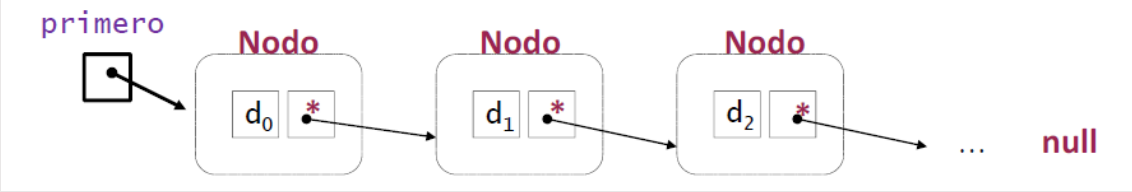
- Partimos de la práctica 1





➔ Problemática de los arrays

- Implementación lista enlazada para carrito



Tarea 1 - CartItem

➔ **Crea la clase CartItem en el paquete cart**

➔ **La clase contendrá las siguientes variables de instancia:**

- clothingItem de tipo ClothingItem
- units de tipo int (unidades que se tienen de él en el carro)

➔ **Declara constructores para instanciar objetos de la clase.**

➔ **Aplica el principio de ocultación**

- Evitar el acceso a las variables de instancia de manera directa
- Declara getters & setters necesarios.

➔ **Sobrescribir toString()**

Tarea 2 - CartItemNode

➔ **Creación de la clase CartItemNode en el paquete cart**

➔ **Variables de instancia (lista enlazada):**

- Variable de instancia data.
- Variable de instancia next.

➔ **Constructores**

- Creación de nodos sin referencia al siguiente nodo.
- Creación de nodos con referencia al siguiente nodo.

➔ **Principio de ocultación y getters/setters**

Tarea 3 - LinkedCart

➔ **Crea esta clase en el paquete cart**

➔ **Variables de instancia necesarias:**

- first, referencia al primer nodo de la lista enlazada.
- size, tipo int

➔ **Constructor de la lista que inicializa una lista enlazada vacía.**

➔ **Métodos de consulta:**

- Comprobar si la lista es vacía.
- Obtener el número de elementos de la lista.
- Buscar un elemento CartItem en la lista indicando nombre y talla, devolviendo su posición
- Buscar (sobrecarga) devuelve el CartItem que está en una posición dada

➔ **Métodos modificadores:**

- Insertar un elemento en una posición dada i (0<= i <=n)
- Eliminar un CartItem que se encuentra en la posición i (0<= i <=n-1), devolviendo el elemento.

Tarea 4 – ShoppingRegister

➔ Agregar Artículos al Carrito (addToCart)

- public void addToCart(Inventory inventory, String itemName, char size, int quantity) {...}
- Verifica si hay stock en el inventario
- Iterativamente extrae artículo del inventario -> inserta el CartItem en el carrito
- Si artículo en carrito, incrementa la cantidad

➔ Eliminar Artículos del Carrito

- public void removeFromCart(Inventory inventory, String itemName, char size, int quantity){...}
- Si artículo esta en carrito reduce la cantidad -> Devuelve unidades eliminadas al inventario
- Si cantidad en carrito es cero -> elimina el artículo del carrito.

➔ Confirmar Carrito y Procesar Venta:

- Muestra los detalles del artículo y calcula el costo total del pedido.
- Devuelve cada artículo al inventario -> registra la venta utilizando el método SalesRegister.processSale.
- Vacía el carrito

➔ Mostrar Carrito

Tarea 5 - Actualizar Main

➔ Añadir opciones gestión carrito de la compra

- 5. Agregar prenda al carrito
- 6. Eliminar prenda del carrito
- 7. Confirmar carrito de compra
- 8. Mostrar productos del carrito

```
*** Bienvenido a Strafalarius ***
Seleccione opción:
1. Agregar nueva prenda al inventario
2. Mostrar inventario
3. Procesar venta
4. Mostrar estadísticas de ventas
5. Agregar prenda al carrito
6. Eliminar prenda del carrito
7. Confirmar carrito de compra
8. Mostrar productos del carrito
9. Salir
Seleccione una opción (1-9):
```

➔ Verificar funcionamiento

- Primero manualmente
- Luego usar verificador

```
TestLinkedCart.checkLinkedCart();
TestShoppingRegister.checkShoppingRegister();
```

```
Productos en el carrito:
Producto: Shirt, Talla: M, Precio: 19.99, Cantidad: 1
Total Score TestShoppingRegister: 10.0 / 10.0
```

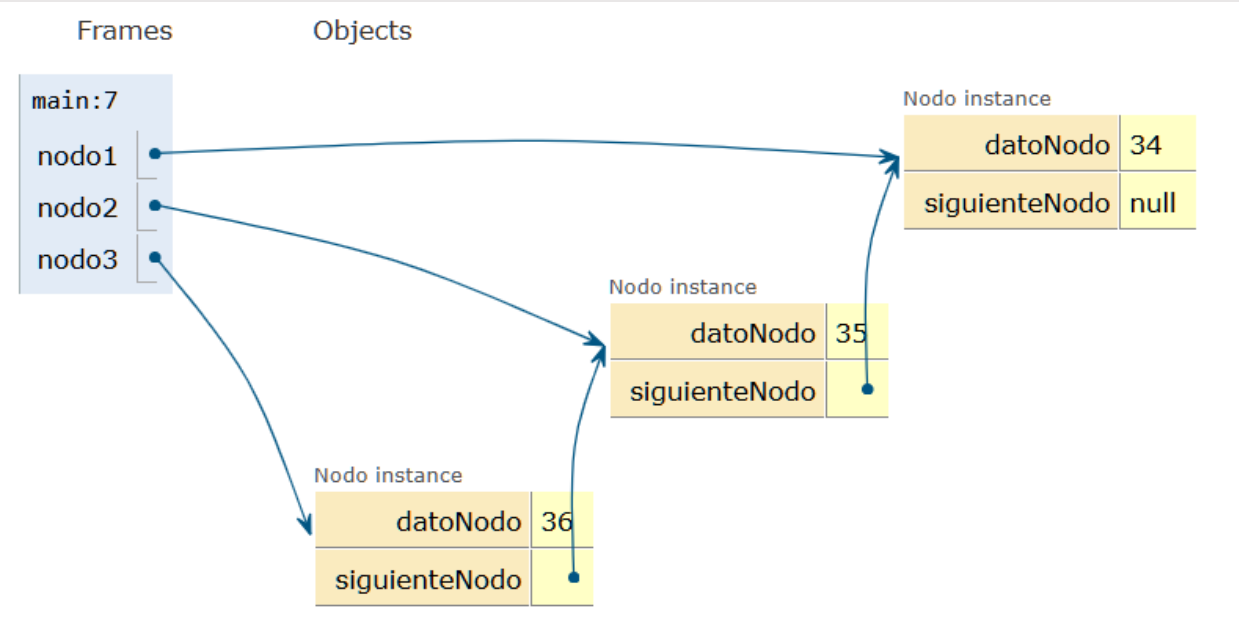
Nodo

➔ Nodo

- Nodo (int dato)
- Nodo (int dato, Nodo siguiente)

```
class Nodo {
    int datoNodo;
    Nodo siguienteNodo;

    Nodo(int dato) {
        datoNodo = dato;
        siguienteNodo = null;
    }

    Nodo(int dato, Nodo siguiente) {
        datoNodo = dato;
        siguienteNodo = siguiente;
    }
}
```

```
public class TestNodo {
    public static void main(String[] args) {
        Nodo nodo1 = new Nodo(34);
        Nodo nodo2 = new Nodo(35, nodo1);
        Nodo nodo3 = new Nodo(36, nodo2);
    }
}
```



link

Pilas

➔ Pila

- Pila(): ultimoNodo, talla
- apilar(int dato)
- int desapilar()
- boolean esVacia()
- int talla()
- datoUltimo()

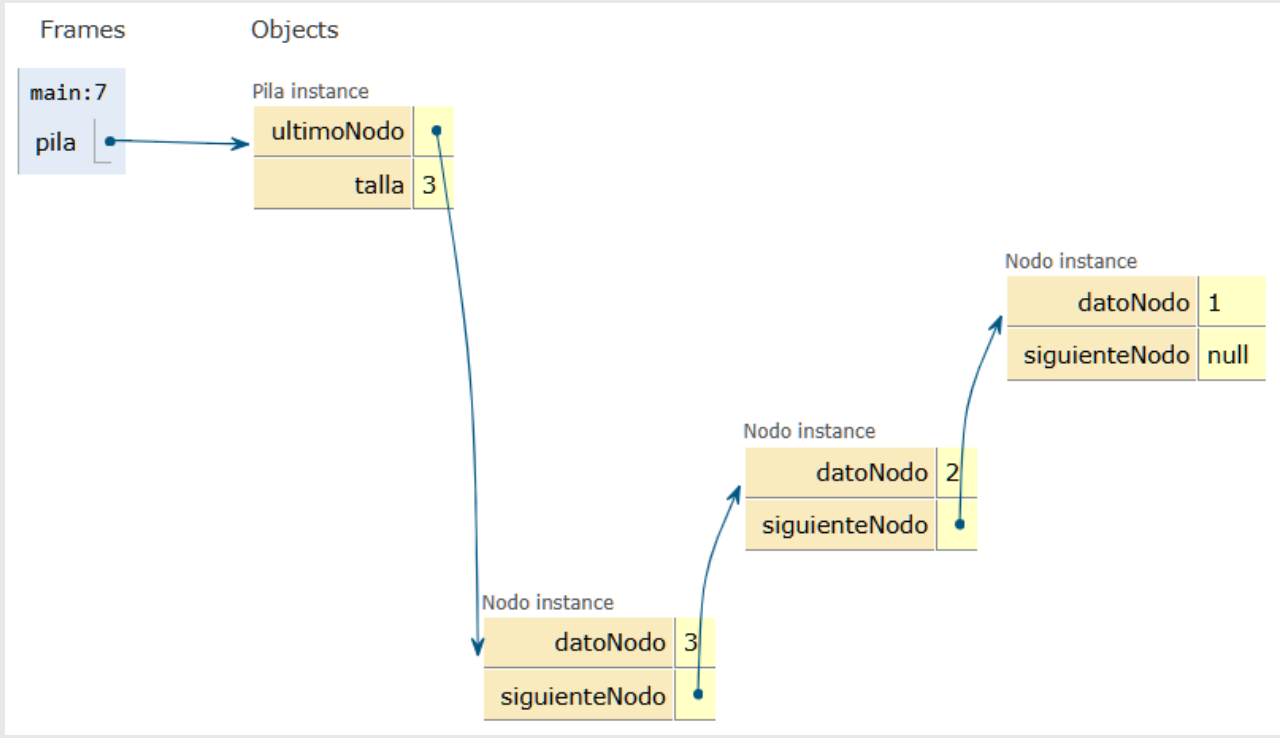
```
class Pila {
    private Nodo ultimo;
    private int talla;

    public Pila() {
        ultimo = null;
        talla = 0;
    }

    public void apilar(int dato) {
        ultimo = new Nodo(dato, ultimo);
        talla++;
    }

    public int desapilar() {
        if (esVacia()) {
            System.out.println("La pila está vacía");
        }
        int dato = ultimo.datoNodo;
        ultimo = ultimo.siguienteNodo;
        talla--;
        return dato;
    }
    ...
}
```

Pilas

➔ Memoria

- apilar(int dato)

```
public class TestPila {
    public static void main(String[] args) {
        Pila pila = new Pila();
        pila.apilar(1);
        pila.apilar(2); // Apilar elementos
        pila.apilar(3); // Apilar elementos
    }
}
```



Cola

➔ Cola

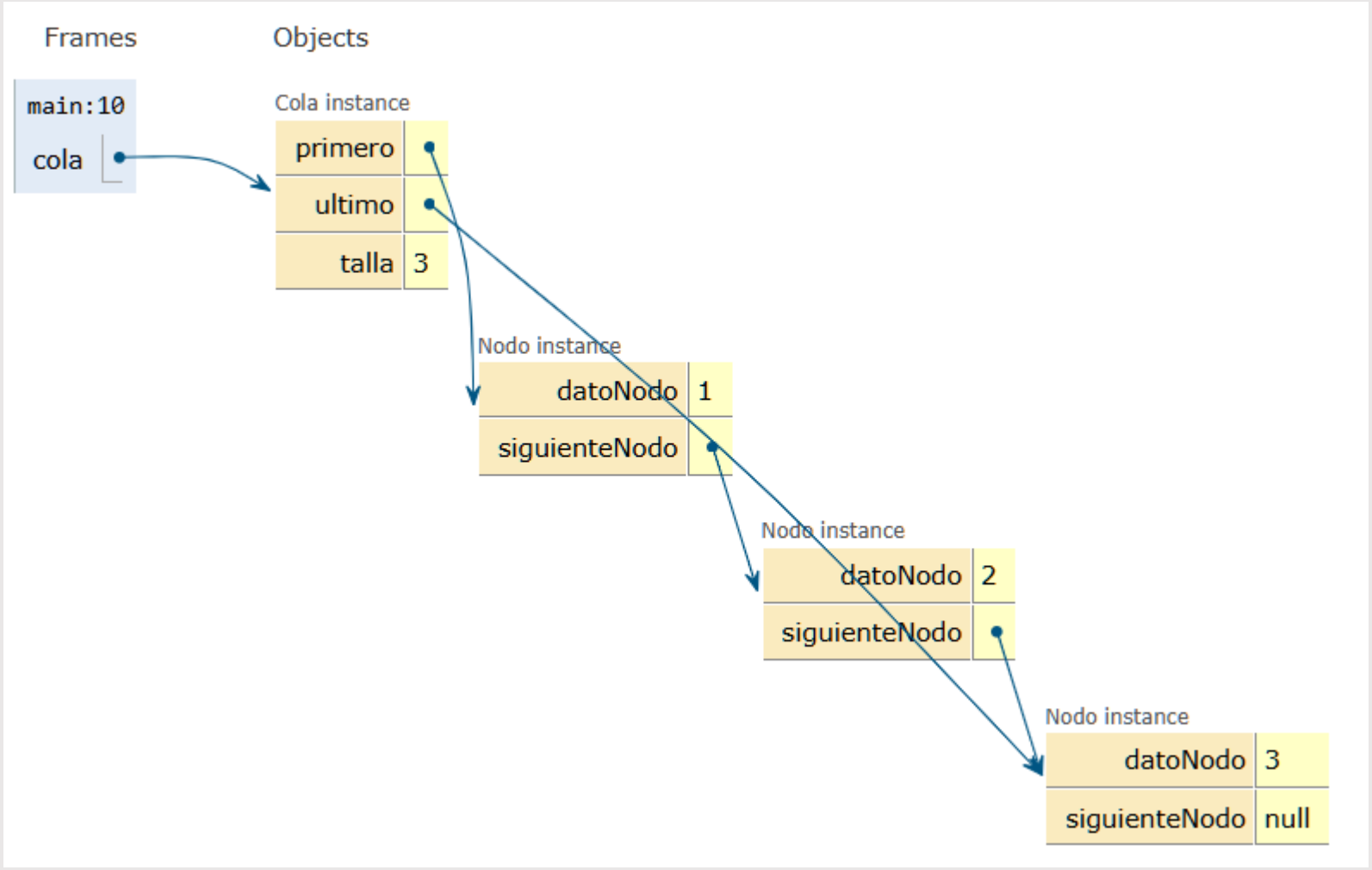
- Cola(), primero, último, talla
- encolar(int dato)
- int desencolar()
- boolean esVacia()
- int talla()
- datoPrimerNodo()

```
class Cola {
    private Nodo primero, ultimo;
    private int talla;

    public Cola() {
        primero = null;
        ultimo = null;
        talla = 0;
    }

    public void encolar(int x) {
        Nodo nuevo = new Nodo(x);
        if (ultimo != null) ultimo.siguienteNodo = nuevo;
        else primero = nuevo;
        ultimo = nuevo;
        talla++;
    }

    public int desencolar() {
        if (esVacia()) {
            System.out.println("La cola está vacía");
        }
        int x = primero.datoNodo;
        primero = primero.siguienteNodo;
        if (primero == null) ultimo = null;
        talla--;
        return x;
    }
...
}
```

Cola

➔ Memoria

- encolar(int dato)



Listas

➔ Lista

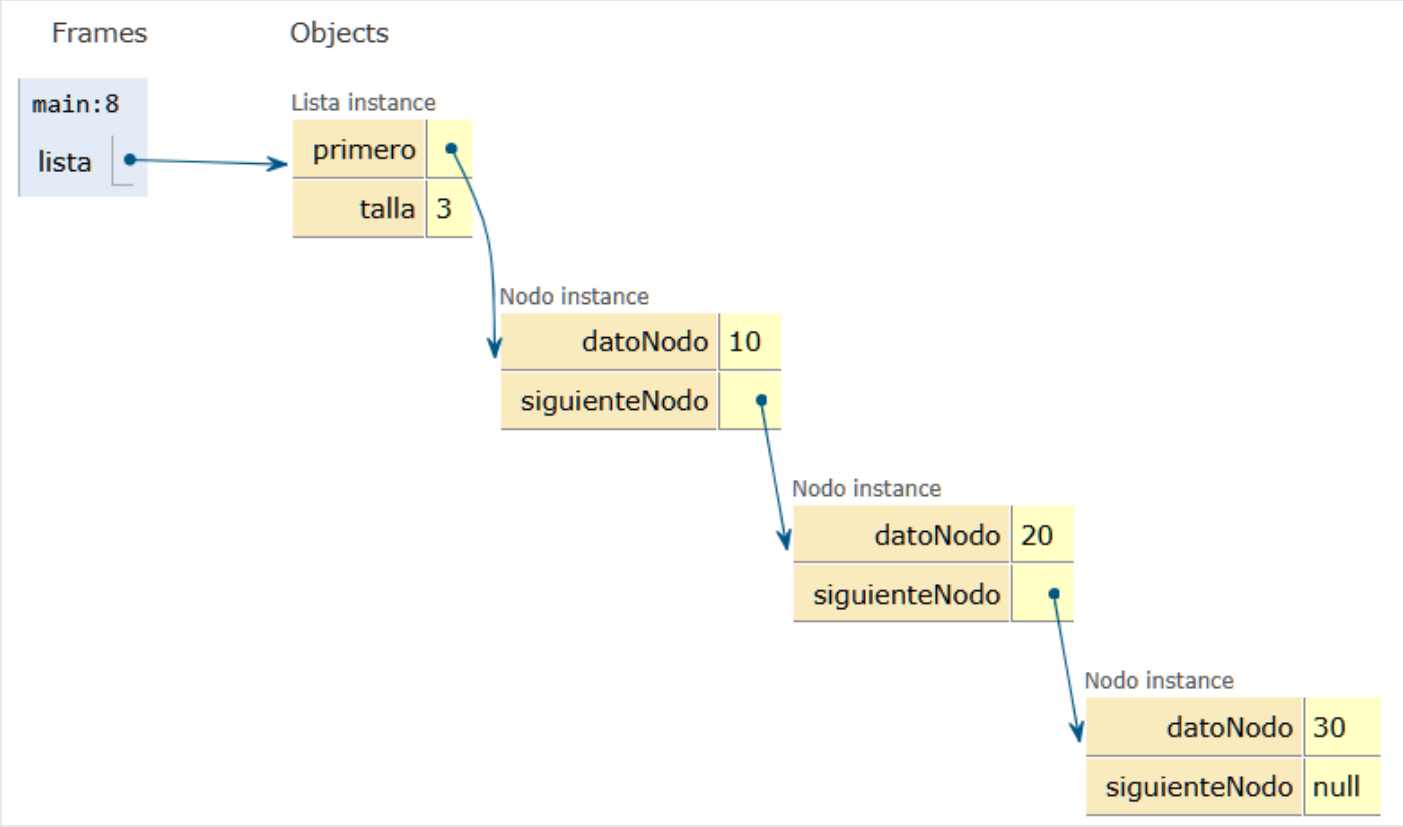
- Lista(), primero, talla
- insertar(int posicion, int dato)
- int eliminar (int posicion)
- int recuperar (int posicion)
- int buscar (int dato)
- boolean esVacia()
- int talla()

```
class Lista {
private Nodo primero;
private int talla;

     public Lista() {
          primero = null;
          talla = 0;
     }

     public void insertar(int posicion, int dato) {
          if (posicion < 0 || posicion > talla)
               System.out.println("Posición fuera de rango");
          if (posicion == 0)
               primero = new Nodo(dato);
          else {
               Nodo aux = primero;
               for (int k = 0; k < posicion - 1; k++)
                    aux = aux.siguienteNodo;
               aux.siguienteNodo = new Nodo(dato);
          }
          talla++;
     }
...
}
```

Lista

➔ Memoria

- insertar(int posicion, int dato)



Práctica 3: Estructuras Datos Lineales

Josué Ferri Pascual

jferri@dsic.upv.es