Ejercicios de clase
TEMA 1 – Estructuras de Datos (EDAs), en Java

Ejercicio 1

Amplia la funcionalidad de la EDA *Pila* vía herencia para añadir un nuevo método que devuelva el elemento situado en la base de la pila. Implementa este método:

a) Accediendo a los atributos de *LEGPila*.
b) Usando únicamente los métodos del modelo.

SOLUCIÓN:

```
public interface PilaExt<E> extends Pila<E> {
    E base(); // SII !esVacia()
}
```

a) Accediendo a los atributos de *LEGPila*:

```
public class LEGPilaExt<E> extends LEGPila<E> implements PilaExt<E> {

    public E base() {
        NodoLEG<E> aux = tope;
        while (aux.siguiente != null)
            aux = aux.siguiente;
        return aux.dato;
    }

}
```

b) Usando únicamente los métodos del modelo:

```
public class LEGPilaExt<E> extends LEGPila<E> implements PilaExt<E> {

    public E base() {
        E res, aux = desapilar();
        if (esVacia()) res = aux;
        else res = base();
        apilar(aux);
        return res;
    }

}
```

Ejercicio 2

Amplia la funcionalidad de la EDA *Cola* vía herencia para añadir un nuevo método que invierta el orden de los elementos la cola. Implementa este método:

a) Accediendo a los atributos de *ArrayCola*.
b) Usando únicamente los métodos del modelo.

SOLUCIÓN:

```
public interface ColaExt<E> extends Cola<E> {
    void invertir();
}
```

a) Accediendo a los atributos de *ArrayCola*:

```
public class ArrayColaExt<E> extends ArrayCola<E> implements ColaExt<E> {

  public void invertir() {
    int i = primero, j = fin;
    for (int cont = 0; cont < talla/2; cont++) {
      E aux = elArray[i];
      elArray[i] = elArray[j];
      elarray[j] = aux;
      if (++i == elArray.length) i = 0;
      if (--j == -1) j = elArray.length – 1;
    }
  }

}
```

b) Usando únicamente los métodos del modelo:

```
public class ArrayColaExt<E> extends ArrayCola<E> implements ColaExt<E> {

  public void invertir() {
    if (!esVacia()) {
      E tmp = desencolar();
      invertir();
      encolar(tmp);
    }
  }

}
```

Ejercicio 3

Implementa la interfaz *Cola* mediante una *ListaConPI* (suponer que tenemos la clase *LEGListaConPI* como implementación de esta interfaz)

SOLUCIÓN:

```
public class LPICola<E> extends LEGListaConPI<E> implements Cola<E> {

  public void encolar(E e) {
    fin();
    insertar(e);
  }

  public E desencolar() {    // SII !esVacia()
    inicio();
    E primero = recuperar();
    eliminar();
    return primero;
  }

  public E primero() {       // SII !esVacia()
    inicio();
    return recuperar();
  }

}
```

Ejercicio 4

Amplia la funcionalidad de la EDA *Lista con Punto de Interés* vía herencia con los siguientes métodos:

- void **buscar**(E x): sitúa el PI sobre *x*. Si el dato no se encuentra se colocará el PI al final de la lista
- void **vaciar**(): vacía la lista
- void **invertir**(): invierte el orden de los elementos de la lista
- void **eliminar**(E x): elimina de la lista todos los elementos iguales a *x*.

Utiliza para ello únicamente los métodos existentes en el modelo *ListaConPI*.

SOLUCIÓN:

```
public interface ListaConPIExt<E> extends ListaConPI<E> {
  void buscar(E x);
  void vaciar();
  void invertir();
  void eliminar(E x);
}


public class LEGListaConPIExt<E> extends LEGListaConPI<E> implements ListaConPIExt<E> {

  public void buscar(E x) {
    inicio();
    while (!esFin() && !recuperar().equals(x)) siguiente();
  }

  public void vaciar() {
    inicio();
    while (!esVacia()) eliminar();
  }

  public void invertir(){
    if (!esVacia()) {
      inicio();
      E dato = recuperar();
      eliminar();
      invertir();
      insertar(dato);
    }
  }

  public void eliminar(E x) {
    inicio();
    while (!esFin())
      if (recuperar().equals(x)) eliminar();
      else siguiente();
  }

}
```

Ejercicio 5

Amplía la funcionalidad de la EDA *Pila* vía herencia para añadir un nuevo método que devuelva el elemento más pequeño de la pila. Implementa este método:

a) Accediendo a los atributos de *LEGPila*.
b) Usando únicamente los métodos del modelo.

SOLUCIÓN:

```
public interface PilaExt<E extends Comparable<E>> extends Pila<E> {
  E minimo();
}
```

a) Accediendo a los atributos de *LEGPila*:

```
public class LEGPilaExt<E extends Comparable<E>> extends LEGPila<E> implements PilaExt<E> {

  public E minimo() {
    NodoLEG<E> aux = tope;
    E min = null;
    while (aux != null) {
      if (min == null || aux.dato.compareTo(min) < 0) min = aux.dato;
      aux = aux.siguiente;
    }
    return min;
  }

}
```

b) Usando únicamente los métodos del modelo:

```
public class LEGPilaExt<E extends Comparable<E>> extends LEGPila<E> implements PilaExt<E> {

  public E minimo() {
    if (esVacia()) return null;
    E dato = desapilar();
    E minResto = minimo();
    apilar(dato);
    if (minResto == null || dato.compareTo(minResto) < 0) return dato;
    return minResto;
  }

}
```