Este programa se desarrolló con el fin de organizar las actividades de la planta de Accudyn, por medio del uso de estructuras de datos. Estas estructuras permiten un control eficiente de los procesos.El programa atiende el departamento de producción, en donde se implementa una cola para manejar las tareas en orden de llegada, asegurando que las actividades se realicen de forma secuencial con la regla de primeras llegadas, primeras salidas.

Por otro lado, en el departamento de mantenimiento correctivo se utiliza una pila para gestionar las fallas reportadas, priorizando la atención a las más recientes, de esta manera, el programa simula el flujo real de trabajo, manteniendo el orden en producción y la inmediatez en mantenimiento. Aquí se implementa la regla de últimas llegadas, primeras salidas, para mantener un orden de corrección de errores lo más rápido posible.Todo está pensado para que el personal cuente con un sistema simple y funcional que optimiza el tiempo y reduce errores. Al momento de estar realizando las actividades pertenecientes al departamento de procesos se incluye la posibilidad de agregar alguna falla o error ocurridas durante el turno. Y esto se anida automáticamente a la lista de errores.

Main

Aquí se encuentra el menú del programa, en este se llama a todas las funciones del resto de clases para poder hacer que el programa funcione correctamente.

```
import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        boolean seguir = true;
        Queue cola = new Queue();
        Pila pila = new Pila();
        // Menú principal de la maquila ACCUDYN
        while (seguir) {
            try {
                // Menú principal para acceder a los departamentos al iniciar tuAño.
                System.out.println("\nBienvenido a ACCUDYN");
                System.out.print("\nA qué departamento ingresarás? \n1. Procesos\n2. Mantenimiento\n3. Terminar turno\n");
                int opcion = scanner.nextInt();
                scanner.nextLine();
                if (opcion < 1 || opcion > 4) {
                    throw new IllegalArgumentException("Valor fuera del rango.");
                }
                switch (opcion) {
                    // Menú interno para realizar actividades de procesos.
                    case 1:
                        boolean submenu1 = true;
                        while (submenu1) {
                            try {
                                System.out.print("\n¿Qué quieres hacer?\n1. Agregar proceso\n2. Atender proceso \n3. Ver lista de procesos\n4. Reportar error\
```

Node

Contiene la información de las listas, además de que definen un puntero para dar un orden a la lista.

```java
public class Node {
    String data;
    Node next;

    // Nodos que contienen la información y el puntero para las listas a crear.
    public Node(String data){
        this.data=data;
        this.next=null;
    }
}
```

Pila

En esta clase se encuentran los procesos de **Pop, Push, Peek y Display**. Con estos comandos se puede eliminar, agregar y mostrar todo lo relacionado con las pilas y su contenido.

```java
public class Pila {
    Node front;
    Node rear;

    // Constructor para almacenar una lista enlazada que simule una pila.
    public Pila() {
        this.front = null;
        this.rear = null;
    }

    // Función para introducir un elemento de tipo String al principio de la lista enlazada en una estructura de pila.
    public void push(String data) {
        Node newNode = new Node(data);
        newNode.next = front;
        front = newNode;
    }

    // Función para remover el elemento introducido más reciente.
    public void pop() {
        if (front == null) {
            return;
        }
        if (front == rear){
            front = rear = null;
        } else {
            front = front.next;
        }

    }
    // Función que devuelve el elemento introducido más recientemente.
```

Queue

En esta clase se encuentra la lógica para el funcionamiento de las colas, aqui se usan las funciones de **enqueue, dequeue, peek y display.** Al igual que con las pilas, estos comandos nos permiten agregar, eliminar y mostrar el contenido de las pilas.

```
// Constructor para almacenar una lista enlazada que simule una cola.
public Queue(){
    this.front = null;
    this.rear = null;
}

// Función para introducir un elemento de tipo String al final de la lista enlazada en una estructura de cola.
public void enqueue(String data){
    Node newNode = new Node(data);
    if(rear ==null){
        front = rear = newNode;
        return;
    }
    rear.next = newNode;
    rear =newNode;
}
// Función para remover el elemento que más tiempo tiene dentro.
public String dequeue(){
    if (front == null){
        return "(no hay procesos)";
    }
    String data = front.data;
    front = front.next;
    if (front == null) {
        rear = null;
    }
    return data;
}

// Función que devuelve el primer elemento que se introdujo.
public String peek(){
```