El programa tiene como objetivo simular la operación de dos ferris, llamados "Portos" y "Celebrian", para transportar autos de un lado a otro. El diseño del programa se basa en una clase base "Ferris" y dos clases hijas "Portos" y "Celebrian" que implementan diferentes comportamientos de extracción de autos utilizando una cola y una pila, respectivamente.

//PHYTON//

Comenzamos teniendo la clase “Ferris” con:

Métodos:

- __init__(self, capacidad_maxima): el constructor de la clase, que recibe la capacidad máxima del ferri y inicializa los atributos.
    - Atributos:
    - capacidad máxima: representa la capacidad máxima de autos que puede transportar el ferri.
    - autos: una cola de autos representada como un objeto deque de la biblioteca collections.

```
class Ferris:
    def __init__(self, capacidad_maxima):
        self.capacidad_maxima = capacidad_maxima
        self.autos = deque()  # Se utiliza deque para representar la cola de autos
```

- insertar_auto(self, matricula): permite insertar un auto en el ferri. Verifica si la capacidad máxima se ha alcanzado y, en ese caso, rechaza la inserción. Si hay espacio disponible, agrega la matrícula del auto a la cola.

```
    def insertar_auto(self, matricula):
        if len(self.autos) < self.capacidad_maxima:
            self.autos.append(matricula)  # Se agrega la matrícula al final de la cola
            return True
        else:
            return False
```

- sacar_auto_pila(self): permite sacar un auto del ferri siguiendo el comportamiento de una pila. Extrae el último auto insertado en la cola y lo devuelve.

```
    def sacar_auto_pila(self):
        if len(self.autos) > 0:
            matricula = self.autos.pop()  # Se saca el último elemento de la cola (pila)
            return matricula
        else:
            return None
```

- sacar_auto_cola(self): permite sacar un auto del ferri siguiendo el comportamiento de una cola. Extrae el primer auto insertado en la cola y lo devuelve.

```
    def sacar_auto_cola(self):
        if len(self.autos) > 0:
            matricula = self.autos.popleft()  # Se saca el primer elemento de la cola (cola)
            return matricula
        else:
            return None
```

- imprimir_contenido(self): imprime el contenido actual de ferri, mostrando las matrículas de los autos en la cola.

```
    def imprimir_contenido(self):
        print("Contenido del ferri:")
        for matricula in self.autos:
            print(matricula)
        print()
```

En esta parte del código se encuentra la función main(), que es el punto de entrada principal del programa. Se crean dos instancias de la clase Ferris llamadas portos y Celebrian con una capacidad máxima de 10 autos cada una.

```
def main():
    portos = Ferris(10)
    celebrian = Ferris(10)
```

Se inicia un bucle infinito utilizando while True. Esto significa que el bucle se ejecutará continuamente hasta que se alcance una condición de salida. Dentro del bucle, se solicita al usuario que ingrese la matrícula del auto. Si el usuario ingresa "salir", se rompe el bucle utilizando break.

```
    while True:
        matricula = input("Ingrese la matrícula del auto (o 'salir' para terminar): ")
        matricula = matricula.lower()
        if matricula == 'salir':
            break
```

A continuación, se solicita al usuario que indique a qué ferris irá el auto ingresado. El usuario puede ingresar 'P' para indicar Portos o 'C' para indicar Celebrian. La entrada del usuario se convierte a minúsculas utilizando el método lower() para que la comparación de opciones sea insensible a mayúsculas.

```
        ferri = input("Ingrese el ferri al que irá el auto (P para Portos, C para Celebrian): ")
        ferri = ferri.lower()
        print(ferri)
```

Dependiendo de la opción seleccionada, se realiza lo siguiente:

- Si el usuario selecciona 'P' para Portos, se intenta insertar el auto en el ferris portos utilizando el método insertar_auto() de la instancia portos. Si la inserción tiene éxito, se muestra un mensaje de que el auto fue ingresado en Portos. De lo contrario, si la capacidad máxima de Portos se ha alcanzado, se muestra un

mensaje indicando que no se pudo ingresar el auto.

```
        if ferri == 'p':
            # Inserción en Portos como una pila
            if portos.insertar_auto(matricula):
                print("Auto ingresado en Portos.")
            else:
                print("Capacidad máxima de Portos alcanzada. No se pudo ingresar el auto.")
```

- Si el usuario selecciona 'C' para Celebrian, se intenta insertar el auto en el ferri celebrian utilizando el método insertar_auto() de la instancia celebrian. Si la inserción tiene éxito, se muestra un mensaje de que el auto fue ingresado en Celebrian. De lo contrario, si la capacidad máxima de Celebrian se ha alcanzado, se muestra un mensaje indicando que no se pudo ingresar el auto.

```
        elif ferri == 'c':
            # Inserción en Celebrian como una cola
            if celebrian.insertar_auto(matricula):
                print("Auto ingresado en Celebrian.")
            else:
                print("Capacidad máxima de Celebrian alcanzada. No se pudo ingresar el auto.")
```

- Si el usuario ingresa una opción inválida, se muestra un mensaje indicando que se debe ingresar 'P' o 'C'.

```
        else:
            print("Ferri inválido. Por favor, ingrese 'P' o 'C'.")
```

Después de cada inserción de auto, se llama al método imprimir_contenido() en ambas instancias portos y celebrian para mostrar el contenido actual de cada ferri.

```
    def imprimir_contenido(self):
        print("Contenido del ferri:")
        for matricula in self.autos:
            print(matricula)
        print()
```

En la siguiente parte del código, se simula el viaje de los ferris después de que se hayan ingresado los autos en los ferris "Portos" y "Celebrian".

Se imprime el mensaje "Comenzando el viaje..." para indicar que se está iniciando el proceso de extracción de autos de los ferris.

```
print("Comenzando el viaje...")
```

- Se inicia un bucle infinito utilizando while True. Esto significa que el bucle se ejecutará continuamente hasta que se alcance una condición de salida.

```
print("Comenzando el viaje...")

while True:
    matricula = portos.sacar_auto_pila()  # Extracción en Portos como una pila
    if matricula is not None:
        print("Auto saliendo de Portos:", matricula)
    else:
        break

    matricula = celebrian.sacar_auto_cola()  # Extracción en Celebrian como una cola
    if matricula is not None:
        print("Auto saliendo de Celebrian:", matricula)
    else:
        break

    portos.imprimir_contenido()
    celebrian.imprimir_contenido()

print("Viaje completado.")
```

- Dentro del bucle, se intenta extraer un auto de "Portos" utilizando el método sacar_auto_pila() de la instancia portos. Si la extracción es exitosa y se obtiene una matrícula de auto, se muestra el mensaje "Auto saliendo de Portos:" seguido de la matrícula del auto extraído. Si no se obtiene una matrícula, significa que no hay más autos en "Portos" y se rompe el bucle utilizando break.

```
while True:
    matricula = portos.sacar_auto_pila()  # Extracción en Portos como una pila
    if matricula is not None:
        print("Auto saliendo de Portos:", matricula)
    else:
        break
```

- A continuación, se intenta extraer un auto de "Celebrian" utilizando el método sacar_auto_cola() de la instancia celebrian. Si la extracción es exitosa y se obtiene una matrícula de auto, se muestra el mensaje "Auto saliendo de Celebrian:" seguido de la matrícula del auto extraído. Si no se obtiene una matrícula, significa que no hay más autos en "Celebrian" y se rompe el bucle utilizando break.

```
    matricula = celebrian.sacar_auto_cola()  # Extracción en Celebrian como una cola
    if matricula is not None:
        print("Auto saliendo de Celebrian:", matricula)
    else:
        break
```

- Después de cada extracción de auto, se llama al método imprimir_contenido() en ambas instancias portos y celebrian para mostrar el contenido actual de cada ferri.

```
    portos.imprimir_contenido()
    celebrian.imprimir_contenido()
```

El bucle continúa extrayendo autos de "Portos" y "Celebrian" en el orden indicado hasta que no queden más autos en alguno de los ferris.

- Una vez que el bucle ha terminado, se imprime el mensaje "Viaje completado." para indicar que se han extraído todos los autos de los ferris y se ha completado el viaje.

```
print("Viaje completado.")
```

Resultados:

```
Contenido del ferri:
1
2
3

Contenido del ferri:
a
b
c

Ingrese la matrícula del auto (o 'salir' para terminar): salir
Comenzando el viaje...
Auto saliendo de Portos: 3
Auto saliendo de Celebrian: a
Contenido del ferri:
1
2

Contenido del ferri:
b
c

Auto saliendo de Portos: 2
Auto saliendo de Celebrian: b
Contenido del ferri:
1

Contenido del ferri:
c

Auto saliendo de Portos: 1
Auto saliendo de Celebrian: c
Contenido del ferri:

Contenido del ferri:

Viaje completado.
```

//JAVA//

Primero llamamos las librerías que requerimos para crear la pila, la cola y para poder leer caracteres desde el teclado.

```
1   import java.util.Stack;
2   import java.util.LinkedList;
3   import java.util.Scanner;
```

Posteriormente en nuestra clase “menú” declaramos una lista enlazada “Celebrian” que tendrá la misma función que una cola para almacenar los carros en el ferri Celebrian, y posteriormente una pila “Portos” para almacenar los carros en el ferri Portos.

```
public class Menu {

    private static LinkedList<String> Celebrian = new LinkedList<String>();
    private static Stack<String> Portos = new Stack<String>();
```

Así mismo declaramos las variables enteras “opc”, “ferri” y “bandera” que funcionan como auxiliares para nuestro menú, y corroborar que si se está almacenando un vehículo en el ferri si hay espacio. La variable “capacidad” la utilizamos checar que los ferris estén llenos o no; y la variable “opc” la inicializamos en “2” para que entre al ciclo do-while.

```
    Run | Debug
    public static void main(String[] args) {
        int opc, ferri, bandera;
        int Capacidad = 10;
        Scanner scanner = new Scanner(System.in);
        String Matricula;
        opc = 2;
```

Verificamos si ambos ferris están llenos para iniciar la simulación, en caso contrario accede al menú donde puede decidir si quiere añadir un carro mas al ferri o desea adelantar la simulación.

```
do {
    if(Celebrian.size() == Capacidad && Portos.size() == Capacidad){
        System.out.println();
        Simulación();

        }else{
            System.out.println(x:"Bienvenido al Menu, Selecciona una opcion");
            System.out.println(x:"1. Ingresar un nuevo carro");
            System.out.println(x:"2. Adelantar la simulación de los Ferris");
            opc = scanner.nextInt();
            scanner.nextLine();
            bandera = 0;
```

Después entramos a un switch-case, en el caso 1 pedimos la matricula del carro y entramos a un ciclo do-while que no se detendrá hasta que el carro se añada a un ferri vacío. Seleccionamos un ferri y corroboramos si esta tiene espacio ese ferri para añadirlo a la pila o a la cola, en caso contrario tendremos que guardarlo en el otro ferri. Se imprimen los vehículos que hay en cada ferri con la función “imprimir” y al final “bandera” recibe el valor “1” para salir del do-while y regresar al menú.

```
switch(opc){
    case 1:
        System.out.println(x:"Ingrese la matrícula del Carro: ");
        Matricula = scanner.nextLine();
        do{
            System.out.println(x:"Seleccione en que Ferri desea subir el auto: ");
            System.out.println(x:"1. Portos");
            System.out.println(x:"2. Celebrian");
            ferri = scanner.nextInt();
            if (ferri == 1) {
                if (Portos.size() == Capacidad) {
                    System.out.println(x:"El ferri Portos está lleno");
                } else {
                    Portos.push(Matricula);
                    System.out.println();
                    imprimir();
                    bandera = 1;
                    break;
                }
            } else if (ferri == 2) {
                if (Celebrian.size() == Capacidad) {
                    System.out.println(x:"El ferri Celebrian está lleno");
                } else {
                    Celebrian.offer(Matricula);
                    System.out.println();
                    imprimir();
                    bandera = 1;
                    break;
                }
            } else {
                System.out.println(x:"Selección inválida, intentelo de nuevo");
            }
        } while(bandera!=1);
        break;
```

En el case 2 adelantamos la simulación de viaje con la función “simulación” y termina el programa.

```
            case 2:
                System.out.println();
                Simulación();
                break;

            default:
                System.out.println(x:"Ingrese una opcion valida");
        }
    }
} while(opc!= 2);
```

Aquí tenemos la función para simular el viaje de los ferris.

```
private static void Simulación() {
    System.out.println(x:"Simulación de viaje del ferri Portos:");
    while (!Portos.isEmpty()) {
        System.out.println("Sale el Carro con matrícula " + Portos.pop());
        imprimir();
    }

    System.out.println();

    System.out.println(x:"Simulación de viaje del ferri Celebrian:");
    while (!Celebrian.isEmpty()) {
        System.out.println("Sale el Carro con matrícula " + Celebrian.poll());
        imprimir();
    }
}
```

Aquí tenemos la función para imprimir el estado de los ferris.

```
private static void imprimir() {
    System.out.println("Ferri Portos: " + Portos);
    System.out.println("Ferri Celebrian: " + Celebrian);
    System.out.println();
}
```