CCE 105L: DATA STRUCTURES AND ALGORITHM

CODE: 4382 BS COMPUTER SCIENCE PS 204

LABORATORY ACTIVITY 2: PORT CONTAINER MANAGEMENT

CODE:

```
package LabActivity;

import java.util.ArrayDeque;
import java.util.Scanner;

  class Container {
      private String id;
      private String description;
      private int weight;

  public Container(String id, String description, int weight) {
     this.id = id;
     this.description = description;
     this.weight = weight;
  }
  @Override
  public String toString() {
     return id + " | " + description + " | " + weight+ "kg";
  }
}
  class Ship{
     private String Name;
     private String Captain;

     public Ship(String Name, String Captain){
        this.Name = Name;
        this.Captain = Captain;
     }
     @Override
     public String toString(){
        return Name + " | " + Captain;
     }
  }
  public class PortContainerManagement_LabAct {
     private static ArrayDeque<Container> containerStack = new ArrayDeque<>();
     private static ArrayDeque<Ship> shipQueue = new ArrayDeque<>();

     public static void main(String[] args) {
        Scanner scan = new Scanner(System.in);
        int choice = 0;
        do {
```

CCE 105L: DATA STRUCTURES AND ALGORITHM

CODE: 4382 BS COMPUTER SCIENCE PS 204

```
            System.out.println("\n=== Port Container Management System ===\n\n"
                + "[1] Store container (push)\n"
                + "[2] View port containers\n"
                + "[3] Register arriving ship (enqueue)\n"
                + "[4] View waiting ships\n"
                + "[5] Load next ship (pop container + poll ship)\n"
                + "[0] Exit");
                choice = scan.nextInt();
                scan.nextLine();

            switch (choice) {
               case 1: StoreContainer();break;
               case 2: ViewPortContainer();break;
               case 3: RegisterArrivingShip();break;
               case 4: ViewWaitingShips();break;
               case 5: LoadNextShip();break;
               default: System.out.println("Ends Program.....");
            }
       } while(choice != 0);
       scan.close();
    }
    private static void StoreContainer(){Scanner scan = new Scanner(System.in);
           System.out.println("Enter Container ID: ");
           String id = scan.nextLine();

           System.out.println("Enter Description: ");
           String description = scan.nextLine();

           System.out.println("Enter Weight (kg): ");
           int weight = scan.nextInt();

           Container cont = new Container(id,description,weight);
           containerStack.push(cont);
           System.out.println("Stored: " + cont);
    }
    private static void ViewPortContainer(){
           if(containerStack.isEmpty()){
              System.out.println("No containers stored.");
              return;
           }
           System.out.println("\nTOP → ");
           for (Container cont : containerStack){
              System.out.println(cont);
           }
           System.out.println("← BOTTOM");
    }
    private static void RegisterArrivingShip(){ Scanner scan = new Scanner(System.in);
```

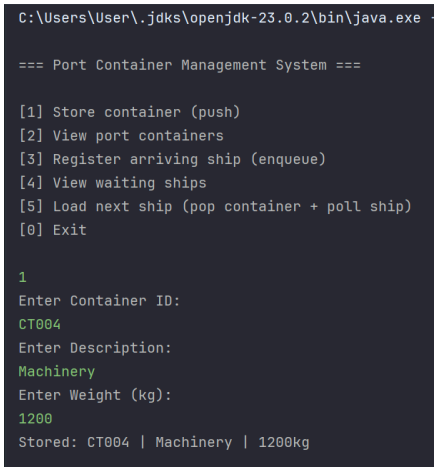
```
        System.out.println("Enter Ship Name: ");
        String shipName = scan.nextLine();

        System.out.println("Enter Name of the Captain: ");
        String capName = scan.nextLine();

        Ship shep = new Ship(shipName,capName);
        shipQueue.offer(shep);
        System.out.println("Registered: " + shep);
  }
  private static void ViewWaitingShips(){
     if(shipQueue.isEmpty()){
        System.out.println("No containers available to load.");
        return;
     }
     System.out.println("\nFRONT → ");
     for (Ship shep : shipQueue){
        System.out.println(shep);
     }
     System.out.println("← REAR");
  }
  private static void LoadNextShip() {
     if (containerStack.isEmpty()) {
        System.out.println("No containers stored.");
        return;
     }
     if (shipQueue.isEmpty()) {
        System.out.println("No containers available to load.");
        return;
     }
     Container cont = containerStack.pop();
     Ship shep = shipQueue.poll();

     System.out.println("Loaded: " + cont + " → " + shep +'\n'
                + "Remaining Containers: "  +containerStack.size() +"\n"
                + "Remaining Ships Waiting: " +shipQueue.size());
  }
}
```

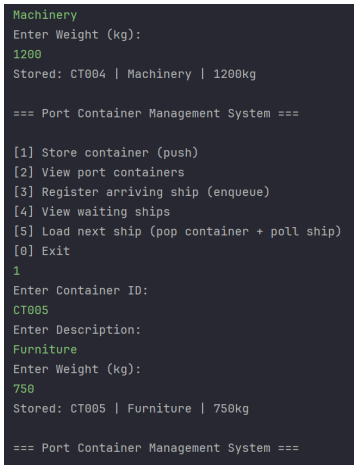
CCE 105L: DATA STRUCTURES AND ALGORITHM

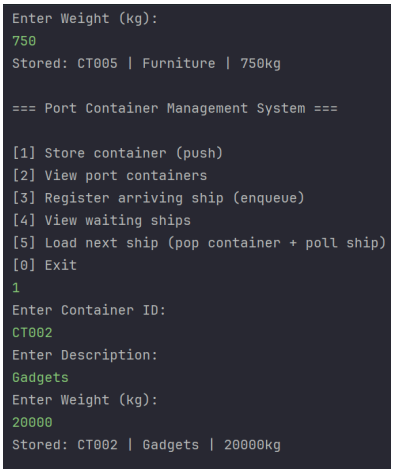
CODE: 4382 BS COMPUTER SCIENCE PS 204

SCREENSHOTS:

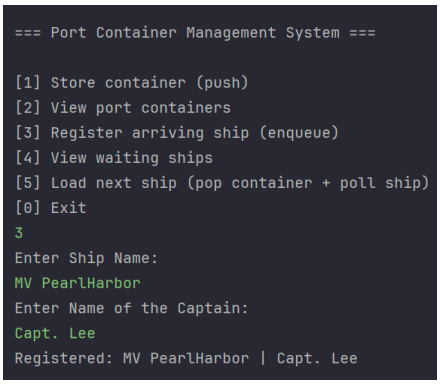
- **Adding at least 3 containers**

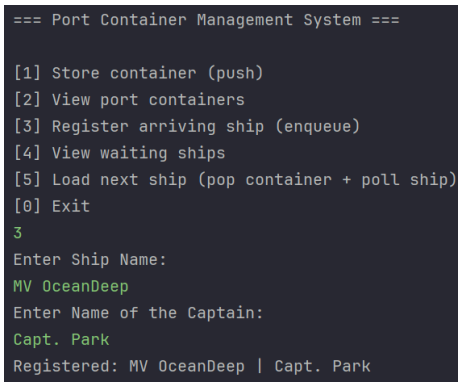




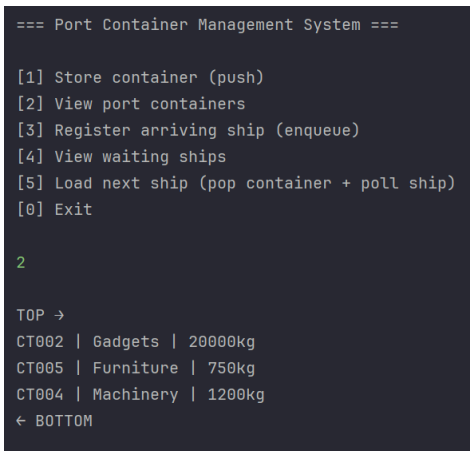


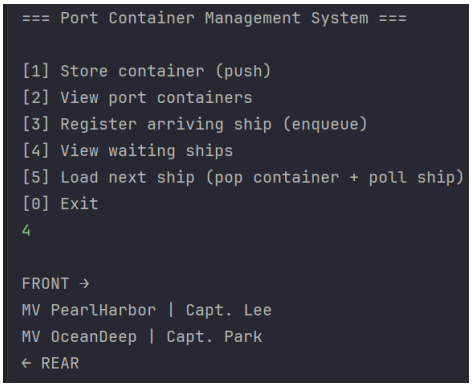
- **Registering at least 2 ships**





- **Viewing both lists**





CCE 105L: DATA STRUCTURES AND ALGORITHM

CODE: 4382 BS COMPUTER SCIENCE PS 204

- **Successfully loading 1 ship**

```
=== Port Container Management System ===

[1] Store container (push)
[2] View port containers
[3] Register arriving ship (enqueue)
[4] View waiting ships
[5] Load next ship (pop container + poll ship)
[0] Exit
5
Loaded: CT002 | Gadgets | 20000kg → MV PearlHarbor | Capt. Lee
Remaining Containers: 2
Remaining Ships Waiting: 1

=== Port Container Management System ===

[1] Store container (push)
[2] View port containers
[3] Register arriving ship (enqueue)
[4] View waiting ships
[5] Load next ship (pop container + poll ship)
[0] Exit
0
Ends Program.....
```