CMPS 347

Mall Management System

Submitted to Dr. Ali El Zaart

By Ibrahim Abou Zahr

CMPS 347

Table of Contents

1. Abstract

1.1 Statement:

The Mall Management System is a comprehensive data structures project designed to improve the efficiency and organization of mall operations. Using different data structures, the program aims to streamline plenty processes within a mall environment, including employment, transactions, and customer tracking.

1.2 Purpose:

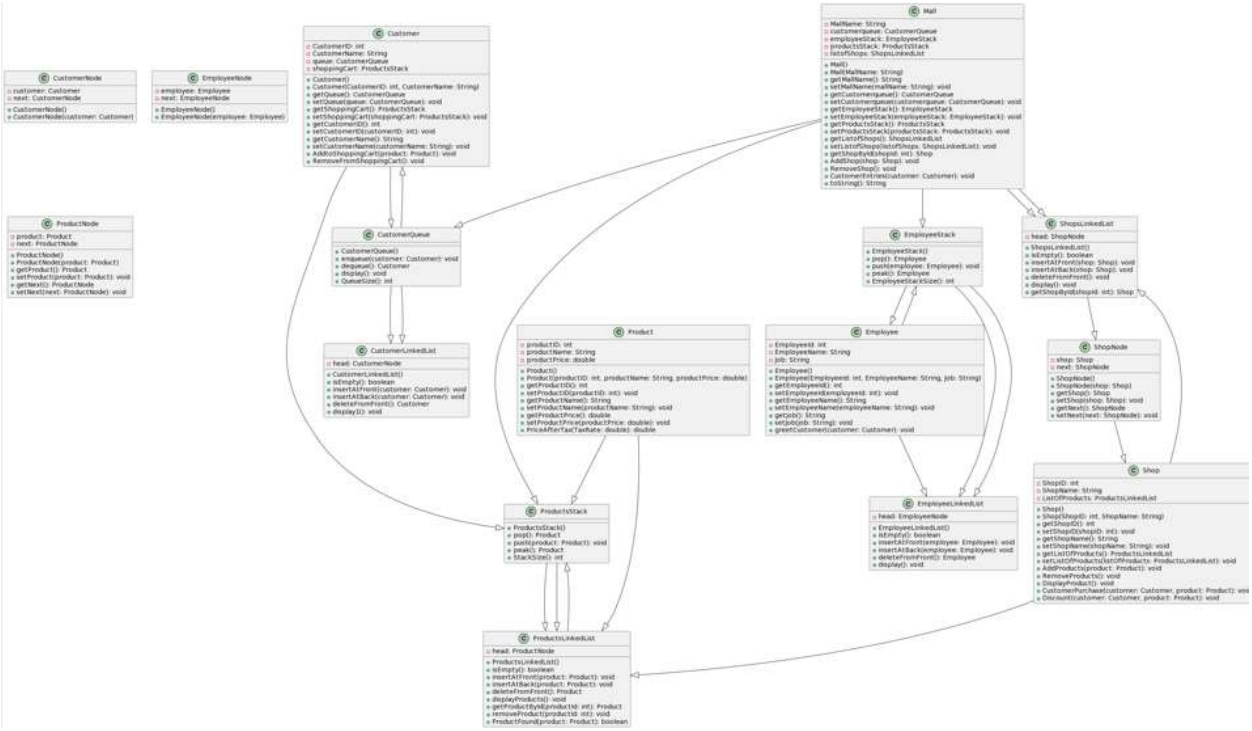
The target purpose and goal behind building such mall program is to ensure customer satisfaction, facilitate operational efficiency and enable data driven decision making.

2. Introduction

2.1 Overview:

Mall Management System is a program that wants to find a way in which customers can navigate through a shopping center and purchase goods as efficiently as possible. Moreover this program aids managers to help in the employment process and to display various employee information of the mall.

3. UML Diagram



4. Classes

4.1 Node Classes

Classes such as the **CustomerNode**, **EmployeeNode**, **ProductNode**, and **ShopNode** all represent a node in the linked list, they contain references to their respective objects in which stores information.

4.2 Linked List Classes

Classes such as **CustomerLinkedList**, **EmployeeLinkedList**, **ProdusctsLinkedList** and **ShopsLinkedList** all manage a linked list for the respective subject. They all contain operations such as insertion at the front and back, deletion from the front, and displaying the list.

4.3 Customer Class

It represents an individual customer within the shopping mall management system. Attributes include: **CustomerID**, **CustomerName**, **CustomerQueue** (queue) and **shoppingCart** (a stack of the customer's selected products). This class also provides **setters and getters** for the above attributes, and it offers functionality to add remove products from the shopping cart.

4.4 Customer Queue Class

This class extends from class **CustomerLinkedList** class, representing a queue of customers in the shopping mall management system. Includes the basics for any queue class such as the **enqueue**, **dequeue** and **display** methods.

4.5 Employee Class

Represents an employee working within the mall's management system. Contains attributes such as **EmployeeId**, **EmployeeName**, and **Job.** This class provides methods such as **setters and getters** and **greetCustomer.** Overall this class encapsulates information for each employee added to the Mall Management System.

4.6 Product Class

Represents a certain product available for purchase within the shopping Mall Management System. Contains attributes such as **produtcID**, **productName** and **productPrice**. This class provides **setter and getters** to each individual attribute. Overall this class encapsulates information for each product added to the Mall Management System.

4.7 Employee and Product Stack Classes

Both classes extend from EmployeeLinkedList Class and **ProductLinkedList** Class respectively. They contain the main methods of

operations for stacks such as: **pop**, **push**, **peak** and **StackSize** methods. Implements the last in first out behavior which reflects the way employees are added and removed and how products are managed in inventory during customer interactions.

4.8 Shop Class

This class includes various attributes such as **ShopID**, **ShopName** and **ListOfProducts** (ProductsLinkedList). Class contains setters and getters for accessing and modifying shop information. Methods such as **AddProducts**, **RemoveProducts** and **DisplayProduct** are found in the above class which help in adding, removing and displaying products in the shop respectively. **CustomerPurchase** method ensures successful purchases from customers by adding their desired products to their shopping cart (stack). **Discount** method is used to apply discounts for customers. Overall, this class plays a crucial role in representing each shop within the mall, including their products and transactions with customers.

4.9 Mall Class

The following class includes attributes such as **MallName**, **customerQueue** (queue), **employeeStack** (stack), **productsStack** (stack) and **listofshops** (Linked List). Provides methods such as getters and setters, **AddShop** and **RemoveShop**, method for handling customer entries which is **CustomerEntries** and using **toString** to provide an overview of the mall's state. The **Mall** class serves as the heart of the program which ties together the different components and data structures used in the Mall Management System.

5. Code

5.1 Node Classes Code

```
public class CustomerNode {
	Customer customer;
	CustomerNode next;

	public CustomerNode() {
		customer = null;
		next = null;
	}

	public CustomerNode(Customer customer) {
		this.customer = customer;
		this.next = null;
	}

}

public class ShopNode {
	public Shop shop;
	ShopNode next;

	public ShopNode() {
		shop = null;
		next = null;
	}
	public ShopNode(Shop shop) {
		this.shop = shop;
		this.next = null;
	}
	public Shop getShop() {
		return shop;
	}
	public void setShop(Shop shop) {
		this.shop = shop;
	}
	public ShopNode getNext() {
		return next;
	}
	public void setNext(ShopNode next) {
		this.next = next;
	}

}

public class ProductNode {
	public Product product;
	ProductNode next;

	public ProductNode() {
		product = null;
		next = null;
	}
```

```
	public ProductNode Product product) {
		this.product = product;
		this.next = null;
	}
	public Product getProduct() {
		return product;
	}
	public void setProduct Product product) {
		this.product = product;
	}
	public ProductNode getNext() {
		return next;
	}
	public void setNext ProductNode next) {
		this.next = next;
	}


}

public class EmployeeNode {
	Employee employee;
	EmployeeNode next;

	public EmployeeNode() {
		employee = null;
		next = null;
	}

	public EmployeeNode Employee employee) {
		this.employee = employee;
		this.next = null;
	}

}
```

5.2 Linked List Classes Code

```
public class CustomerLinkedList {
	CustomerNode head;

	public CustomerLinkedList() {
		head = null;
	}

	public boolean isEmpty() {
		if (head == null) {
			return true;
		}
```

```
		return false;
	}

	public void insertAtFront(Customer customer) {
		CustomerNode c = new CustomerNode(customer);
		if (isEmpty()) {
			head = c;
		}
		c.next = head;
		head = c;
	}
	public void insertAtBack(Customer customer) {
		CustomerNode c = new CustomerNode(customer);
		if (isEmpty()) {
			head = c;
		}
		else {
			CustomerNode current = head;
			while(current.next!= null) {
				current = current.next;
			}

			current.next = c;
		}


	}

	public Customer deleteFromFront() {
		if (!isEmpty()) {
			Customer c = head.customer;
			head = head.next;
			return c;

		}
		return null;
	}

	public void display1() {
		CustomerNode current = head;

		while(current!= null) {
			System.out.println("Customer ID is: " +
current.customer.CustomerID);
		}


	}
}
public class EmployeeLinkedList {
	EmployeeNode head;

	public EmployeeLinkedList() {
		head = null;
	}

	public boolean isEmpty() {
```

```
		if (head == null) {
			return true;

		}
		return false;
	}

	public void insertAtFront(Employee employee) {
		EmployeeNode n = new EmployeeNode(employee);

		if (isEmpty()) {
			head = n;
		} else {
			n.next = head;
			head = n;
		}
	}

	public void insertAtBack(Employee employee) {
		EmployeeNode n = new EmployeeNode(employee);
		if (isEmpty()) {
			head = n;
		} else {
			EmployeeNode current = head;

			while (current.next != null) {
				current = current.next;
			}
			current.next = n;
		}
	}

	public Employee deleteFromFront() {
		if (!isEmpty()) {
			Employee p = head.employee;

			head = head.next;
			return p;
		} else {
			return null;
		}
	}

	public void display() {
		EmployeeNode current = head;

		while (current != null) {
			System.out.println("ID: "+current.employee.EmployeeId);
			System.out.println("Employee Name:" +
current.employee.EmployeeName);
			System.out.println("Employee Job Title: " +
current.employee.Job);
			current = current.next;
		}
	}
```

```
}

public class ProductsLinkedList {
	ProductNode head;

	public ProductsLinkedList() {
		head = null;
	}

	public boolean isEmpty() {
		if (head == null) {
			return true;

		}
		return false;
	}

	public void insertAtFront(Product product) {
		ProductNode n = new ProductNode(product);

		if (isEmpty()) {
			head = n;
		} else {
			n.next = head;
			head = n;
		}
	}

	public void insertAtBack(Product product) {
		ProductNode n = new ProductNode(product);
		if (isEmpty()) {
			head = n;
		} else {
			ProductNode current = head;

			while (current.next != null) {
				current = current.next;
			}
			current.setNext(n);
		}
	}

	public Product deleteFromFront() {
		if (!isEmpty()) {
			Product p = head.product;

			head = head.next;
			return p;
		} else {
			return null;
		}
	}

	public void displayProducts() {
```

```
            ProductNode current = head;

            while (current != null) {
                  System.out.println("ID: " + current.product.productID);
                  System.out.println("Name: " + current.product.productName);
                  System.out.println("Price: " + current.product.productPrice +
"$");
                  System.out.println();
                  current = current.next;
            }
      }

      public Product getProductById(int productId) {
            ProductNode current = head;
            while (current != null) {
                  if (current.getProduct().getProductID() == productId) {
                        return current.getProduct();
                  }
                  current = current.getNext();
            }
            return null;
      }

      public void removeProduct(int productId) {
            if (head == null) {
                  return;
            }

            if (head.getProduct().getProductID() == productId) {

                  deleteFromFront();
                  // in order to remove first node
            }

            ProductNode current = head;
            while (current.getNext() != null &&
current.getNext().getProduct().getProductID() != productId) {
                  current = current.getNext();
            }

            if (current.getNext() != null) {
                  current.setNext(current.getNext().getNext());
            }
      }

      public boolean ProductFound(Product product) {
            ProductNode current = head;
            while (current != null) {
                  if (current.getProduct().equals(product)) {
                        return true;
                  }
                  current = current.next;
            }
            return false;
      }
```

```
}
public class ShopsLinkedList {
	ShopNode head;

	public ShopsLinkedList() {
		head = null;
	}

	public boolean isEmpty() {
		if (head == null) {
			return true;

		}
		return false;
	}

	public void insertAtFront(Shop shop) {
		ShopNode n = new ShopNode(shop);

		if (isEmpty()) {
			head = n;
		} else {
			n.next = head;
			head = n;
		}
	}

	public void insertAtBack(Shop shop) {
		ShopNode n = new ShopNode(shop);
		if (isEmpty()) {
			head = n;
		} else {
			ShopNode current = head;

			while (current.next != null) {
				current = current.next;
			}
			current.next = n;
		}
	}

	public void deleteFromFront() {
		if (!isEmpty()) {
			head = head.next;
		}
	}
	public void display() {
		ShopNode current = head;

		while (current != null) {
			System.out.println(current.shop.ShopID + ")" +
current.shop.ShopName);

			System.out.println();
```

```
                current = current.next;
            }
        }

        public Shop getShopById int shopId) {
          ShopNode current = head;
          while (current != null) {
              if (current.getShop().getShopID() == shopId) {
                  return current.getShop();
              }
              current = current.getNext();
          }
          return null;
      }

}                5
```

5.3 Customer Class Code

```
public class Customer {
        int CustomerID;
        String CustomerName;
        CustomerQueue queue;
        ProductsStack shoppingCart;


        public Customer() {
                CustomerID = 0;
                CustomerName = null;
        }

        public Customer(int CustomerID, String CustomerName) {
                this.CustomerID = CustomerID;
                this.CustomerName = CustomerName;
                this.shoppingCart = new ProductsStack();

        }
        public CustomerQueue getQueue() {
                return queue;
        }

        public void setQueue CustomerQueue queue) {
                this.queue = queue;
        }

        public ProductsStack getShoppingCart() {
                return shoppingCart;
        }

        public void setShoppingCart ProductsStack shoppingCart) {
                this.shoppingCart = shoppingCart;
        }
```

```
	public int getCustomerID() {
		return CustomerID;
	}

	public void setCustomerID(int customerID) {
		CustomerID = customerID;
	}

	public String getCustomerName() {
		return CustomerName;
	}

	public void setCustomerName(String customerName) {
		CustomerName = customerName;
	}

	public void AddtoShoppingCart(Product product) {
		shoppingCart.push(product);
		System.out.println("Added Sucessfully!");

	}

	public void RemoveFromShoppingCart() {
		if (!shoppingCart.isEmpty()) {
			Product removed = shoppingCart.pop();

			System.out.println("The product " + removed.getProductName() + "
of ID " + removed.getProductID()
					+ " has been removed Sucessfully! ");
		} else {
			System.out.println("Your cart is Empty!");
		}
	}



}
```

5.4 Customer Queue Class Code

```
public class CustomerQueue extends CustomerLinkedList {
	public CustomerQueue() {
		super();

	}

	public void enqueue(Customer customer) {
		insertAtBack(customer);
	}

	public Customer dequeue() {
```

```
            return deleteFromFront();
        }

        public void display() {
            super.display1();
        }

        public int QueueSize() {
            int count = 0;
            Customer a;
            CustomerQueue temp = new CustomerQueue();
            while (!isEmpty()) {
                a = dequeue();
                count++;
                temp.enqueue(a);

            }
            while(!temp.isEmpty()) {
                a = temp.dequeue();
                enqueue(a);
            }
            return count;
        }

}
```

5.5 Employee Class Code

```
public class Employee {
    int EmployeeId;
    String EmployeeName;
    String Job;

    public Employee() {
        EmployeeId = 0;
        EmployeeName = null;
        Job = null;
    }

    public Employee int EmployeeId, String EmployeeName, String Job) {
        this.EmployeeId = EmployeeId;
        this.EmployeeName = EmployeeName;
        this.Job = Job;
    }

    public int getEmployeeId() {
        return EmployeeId;
    }

    public void setEmployeeId(int employeeId) {
        EmployeeId = employeeId;
    }
```

```
	public String getEmployeeName() {
		return EmployeeName;
	}

	public void setEmployeeName(String employeeName) {
		EmployeeName = employeeName;
	}

	public String getJob() {
		return Job;
	}

	public void setJob String job) {
		Job = job;
	}

	public void greetCustomer(Customer customer) {
		System.out.println("Hello " + customer.getCustomerName() + " welcome to
Ibrahim's! How may I help you? ");
	}

}
```

5.6 Product Class Code

```
public class Product {
	int productID;
	String productName;
	double productPrice;

	public Product() {
		productID = 0;
		productName = null;
		productPrice = 0.0;
	}

	public Product(int productID, String productName, double productPrice) {
		this.productID = productID;
		this.productName = productName;
		this.productPrice = productPrice;

	}

	public int getProductID() {
		return productID;
	}

	public void setProductID(int productID) {
		this.productID = productID;
	}
```

```
	public String getProductName() {
		return productName;
	}

	public void setProductName(String productName) {
		this.productName = productName;
	}

	public double getProductPrice() {
		return productPrice;
	}

	public void setProductPrice(double productPrice) {
		this.productPrice = productPrice;
	}

	public double PriceAfterTax(double TaxRate) {
		double tax = productPrice * (TaxRate / 100);
		double price = tax + productPrice;

		return price;
	}

}
```

5.7 Employee and Product Stack Classes Code

```
public class EmployeeStack extends EmployeeLinkedList {

	public EmployeeStack() {
		super();
	}

	public Employee pop() {
		return deleteFromFront();
	}

	public void push(Employee employee) {
		insertAtFront(employee);
	}

	public Employee peak() {
		return head.employee;
	}

	public int EmployeeStackSize() {
		int count = 0;
		Employee a;
		EmployeeStack temp = new EmployeeStack();
		while (!isEmpty()) {
			a = pop();
			count++;
			temp.push(a);
```

```
		}
		while (!temp.isEmpty()) {
			a = temp.pop();
			push(a);
		}
		return count;
	}


}

public class ProductsStack extends ProductsLinkedList {
	public ProductsStack() {
		super();
	}

	public Product pop() {
		return deleteFromFront();
	}

	public void push(Product product) {
		insertAtFront(product);
	}

	public Product peak() {
		return head.product;
	}

	public int StackSize() {
		int count = 0;
		Product a;
		ProductsStack temp = new ProductsStack();
		while (!isEmpty()) {
			a = pop();
			count++;
			temp.push(a);
		}
		while (!temp.isEmpty()) {
			a = temp.pop();
			push(a);
		}
		return count;
	}

}
```

5.8 Shop Class Code

```
public class Shop {
	int ShopID;
	String ShopName;
```

```
	ProductsLinkedList ListOfProducts;

	public Shop() {
		ShopID = 0;
		ShopName = null;
		ListOfProducts = null;

	}
	public Shop(int ShopID, String ShopName) {
		this.ShopID = ShopID;
		this.ShopName = ShopName;
		 this.ListOfProducts = new ProductsLinkedList();
	}

	public int getShopID() {
		return ShopID;
	}
	public void setShopID int shopID) {
		ShopID = shopID;
	}
	public String getShopName() {
		return ShopName;
	}
	public void setShopName String shopName) {
		ShopName = shopName;
	}
	public ProductsLinkedList getListOfProducts() {
		return ListOfProducts;
	}
	public void setListOfProducts(ProductsLinkedList listOfProducts) {
		ListOfProducts = listOfProducts;
	}
	public void AddProducts Product product) {
		ListOfProducts.insertAtBack(product);

	}
	public void RemoveProducts() {
		ListOfProducts.deleteFromFront();
	}
	public void DisplayProduct() {
		System.out.println "The products in stock at " + ShopName + " are:" );
		ListOfProducts.displayProducts();
	}
	public void CustomerPurchase(Customer customer, Product product) {

			if ListOfProducts.ProductFound(product)) {
				ListOfProducts.removeProduct product.getProductID());
				// add the selected product to the shopping cart
				customer.getShoppingCart().push product);
				System.out.println product.getProductName() + " purchase
from " + ShopName + " is successful!");

			}
			else {
```

```
                        System.out.println("The selected product is not available
in " + ShopName + ".");
                    }
                }

        public void Discount(Customer customer, Product product) {
                System.out.println(customer.getCustomerName() + " we currently have a
30% discount on most products for Christmas Holidays! ");
                double discount = product.getProductPrice()*0.7;
                System.out.println("The price of "+ product.getProductName() + " after
discounting is: " + discount);

        }

}
```

5.9 Mall Class Code

```
public class Mall {
        String MallName;
        CustomerQueue customerqueue;
        EmployeeStack employeeStack;
        ProductsStack productsStack;
        ShopsLinkedList listofShops;

        public Mall() {
                MallName = null;
                customerqueue = null;
                employeeStack = null;
                productsStack = null;
                listofShops = null;
        }

        public Mall(String MallName) {
                this.MallName = MallName;

                this.listofShops = new ShopsLinkedList();
                this.productsStack = new ProductsStack();
                this.employeeStack = new EmployeeStack();
                this.customerqueue = new CustomerQueue();
        }

        public String getMallName() {
                return MallName;
        }

        public void setMallName(String mallName) {
                MallName = mallName;
        }

        public CustomerQueue getCustomerqueue() {
                return customerqueue;
        }
```

```
public void setCustomerqueue(CustomerQueue customerqueue) {
	this.customerqueue = customerqueue;
}

public EmployeeStack getEmployeeStack() {
	return employeeStack;
}

public void setEmployeeStack(EmployeeStack employeeStack) {
	this.employeeStack = employeeStack;
}

public ProductsStack getProductsStack() {
	return productsStack;
}

public void setProductsStack(ProductsStack productsStack) {
	this.productsStack = productsStack;
}

public ShopsLinkedList getListofShops() {
	return listofShops;
}

public void setListofShops(ShopsLinkedList listofShops) {
	this.listofShops = listofShops;
}

public Shop getShopById(int shopId) {
	return listofShops.getShopById(shopId);
}



public String toString() {
	return "Mall{" +
	   "mallName='" + MallName + '\'' +
	   ", listOfShops=" + listofShops +
	   ", customerQueue=" + customerqueue +
	   ", employeeStack=" + employeeStack +
	   '}';
}

public void AddShop Shop shop) {
	listofShops.insertAtFront(shop);
}

public void RemoveShop() {
	listofShops.deleteFromFront();
}

public void CustomerEntries(Customer customer) {
	customerqueue.enqueue(customer);
	System.out.println("Welcome to " + MallName + " enjoy your stay!");
}
```

}

6. Main Class

6.1 Overview

The main class in this program serves as the entry point of the Mall Management System. It calls all functionalities used in all classes in order to build a well-functioning system for customer transactions, employment and overall management of information and products.

6.2 Code

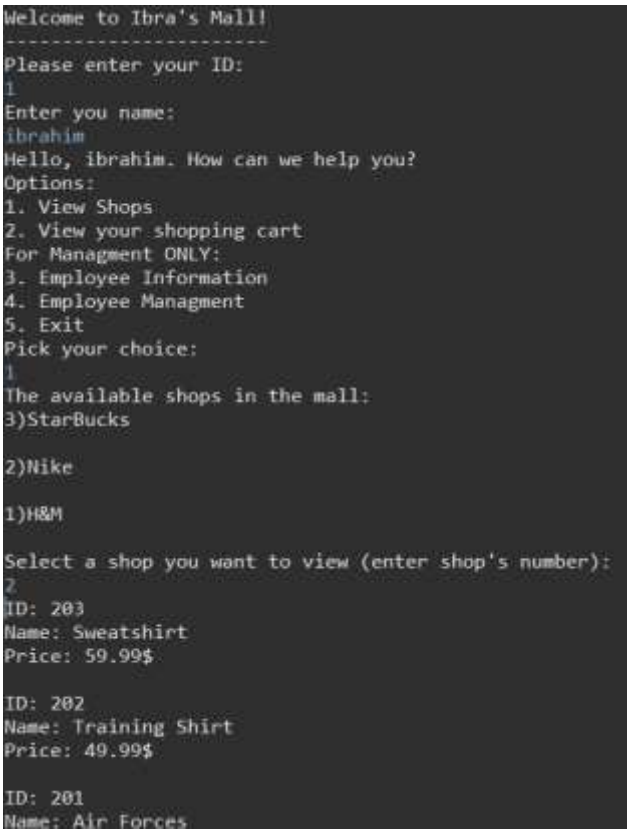
```
import java.util.Scanner;

public class Main {

	public static void main(String[] args) {
		Scanner scan = new Scanner(System.in);

		System.out.println("Welcome to Ibra's Mall!");

		Mall IbraMall = new Mall("IbraMall");

		Shop s1 = new Shop(1, "H&M");
		Shop s2 = new Shop(2, "Nike");
		Shop s3 = new Shop(3, "StarBucks");

		IbraMall.AddShop(s1);
		IbraMall.AddShop(s2);
		IbraMall.AddShop(s3);

		// for H&M
		Product product1 = new Product(101, "Blue Hoodie", 34.99);
		Product product2 = new Product(102, "Sweatpants", 49.99);
		Product product3 = new Product(103, "Denim Jacket", 79.99);
		// for Nike
		Product product4 = new Product(201, "Air Forces", 119.99);
		Product product5 = new Product(202, "Training Shirt", 49.99);
		Product product6 = new Product(203, "Sweatshirt", 59.99);
		// for Starbucks
		Product product7 = new Product(301, "Latte", 4.99);
		Product product8 = new Product(302, "Smoothie", 5.99);
		Product product9 = new Product(303, "Cake", 9.99);

		s1.getListOfProducts().insertAtFront(product1);
```

```
		s1.getListOfProducts().insertAtFront(product2);
		s1.getListOfProducts().insertAtFront(product3);

		s2.getListOfProducts().insertAtFront(product4);
		s2.getListOfProducts().insertAtFront(product5);
		s2.getListOfProducts().insertAtFront(product6);

		s3.getListOfProducts().insertAtFront(product7);
		s3.getListOfProducts().insertAtFront(product8);
		s3.getListOfProducts().insertAtFront(product9);

		System.out.println("-----------------------");

		int customerId;
		String customerName;
		System.out.println("Please enter your ID: ");
		customerId = scan.nextInt();
		System.out.println("Enter you name: ");
		customerName = scan.next();

		Customer customer = new Customer(customerId, customerName);

		IbraMall.getCustomerqueue().enqueue(customer);

		System.out.println("Hello, " + customer.getCustomerName() + ". How can
we help you?");
		int c;
		do {
			System.out.println("Options: ");
			System.out.println("1. View Shops");
			System.out.println("2. View your shopping cart");
			System.out.println("For Managment ONLY:");
			System.out.println("3. Employee Information");
			System.out.println("4. Employee Managment ");
			System.out.println("5. Exit");

			System.out.println("Pick your choice: ");
			c = scan.nextInt();

			switch (c) {
			case 1:
				System.out.println("The available shops in the mall: ");
				IbraMall.getListofShops().display();
				System.out.println("Select a shop you want to view (enter
shop's number): ");
				int shopChoice;
				shopChoice = scan.nextInt();

				Shop selected = IbraMall.getShopById(shopChoice);
				if (selected != null) {
					selected.getListOfProducts().displayProducts();
					System.out.println("Select a product you are
interested in (enter product's id):");
					int selectedProduct = scan.nextInt();
```

```
                                Product selectedProductChoice =
selected.getListOfProducts().getProductById(selectedProduct);
                                if (selectedProductChoice != null) {
                                      selected.CustomerPurchase(customer,
selectedProductChoice);

                                      System.out.println("----------------------
");

                                } else {
                                      System.out.println("Invalid Choice!");
                                      System.out.println("----------------------
");
                                }

                          }
                          break;
                    case 2:
                          System.out.println("Your Shopping Cart: ");
                          customer.getShoppingCart().displayProducts();
                          System.out.println("-----------------------");
                          break;

                    case 3:
                          System.out.println("Employee Inforation: ");
                          IbraMall.getEmployeeStack().display();
                          System.out.println("---------------------------------------
--");

                    case 4:
                          System.out.println("Employee Managment: ");
                          System.out.println("1. Add Employee");
                          System.out.println("2. Remove Employee");
                          System.out.println("3. Go Back");
                          System.out.println("Enter your choice: ");
                          int EmployeeChoice = scan.nextInt();

                          switch (EmployeeChoice) {
                          case 1:
                                int EmployeeId;
                                String EmployeeName;
                                String EmployeeJobTitle;
                                System.out.println("Enter the Employee ID: ");
                                EmployeeId = scan.nextInt();
                                System.out.println("Enter the Employee's Name:");
                                EmployeeName = scan.next();
                                System.out.println("Enter the Employee's Job Title:
");
                                EmployeeJobTitle = scan.next();

                                Employee newEmployee = new Employee(EmployeeId,
EmployeeName, EmployeeJobTitle);

                                IbraMall.getEmployeeStack().push(newEmployee);
                                System.out.println("New Employee assigned!");
```

```
                                System.out.println("----------------------------------
---------");
                                System.out.println();
                                break;
                        case 2:
                                if (!IbraMall.getEmployeeStack().isEmpty()) {
                                        Employee removed =
IbraMall.getEmployeeStack().pop();
                                        System.out.println("Employee removed! ");
                                } else {
                                        System.out.println("No employee to remove!");
                                }
                                System.out.println("----------------------------------
------");

                        case 3:
                                break;

                        default:
                                System.out.println("Invalid Choice!");
                                System.out.println("----------------------------------
--------");
                                break;

                        }


                case 5:
                        System.out.println("Thank you for shopping at Ibra's
Mall!");
                        System.out.println("--------------------------------------
--");
                        break;

                default:
                        System.out.println("Invalid Choice!");
                        System.out.println("--------------------------------------
--");
                        break;
                }
        } while (c != 5);
    }

}
```

7. Output


Welcome to Ibra's Mall!
Please enter your ID:
1
Enter you name:
ibrahim
Hello, ibrahim. How can we help you?
Options:
1. View Shops
2. View your shopping cart
For Managment ONLY:
3. Employee Information
4. Employee Managment
5. Exit
Pick your choice:
1
The available shops in the mall:
3)StarBucks
2)Nike
1)H&M
Select a shop you want to view (enter shop's number):
2
ID: 203
Name: Sweatshirt
Price: 59.99$
ID: 202
Name: Training Shirt
Price: 49.99$
ID: 201
Name: Air Forces

```
3. Employee Information
4. Employee Managment
5. Exit
Pick your choice:
2
Your Shopping Cart:
ID: 201
Name: Air Forces
Price: 119.99$

----------------------
Options:
1. View Shops
2. View your shopping cart
For Managment ONLY:
3. Employee Information
4. Employee Managment
5. Exit
Pick your choice:
3
Employee Inforation:
---------------------------------
Employee Managment:
1. Add Employee
2. Remove Employee
3. Go Back
Enter your choice:
1
Enter the Employee ID:
3
Enter the Employee's Name:
bob
Enter the Employee's Job Title:
clerk
New Employee assigned!
```

8. Conclusion

In conclusion, the Mall Management System presented here, embodies an ideal and comprehensive solution for orchestrating the diverse activities with a shopping complex. The use of the various data structures such as Linked Lists, Stacks, and Queues has made this system perfect for execution and implementation.