COGNIZANT WEEK 2
SENJUTI GHOSAL
RA2111030010096
Exercise 1: Implementing a Product Catalog with Set and HashSet

Objectives:

- Understand and use the Set interface and HashSet class.
- Add, remove, and search for elements in a HashSet.

Business Scenario:

You are developing a product catalog for an online store. The catalog should store unique product names and provide functionality to add, remove, and search for products.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **ProductCatalog**.
2. **Create a ProductCatalog Class:**
    - In the **ProductCatalog** project, create a class named **ProductCatalog**.
    - Use a **HashSet<String>** to store unique product names.
3. **Add Products:**
    - Implement a method **addProduct(String productName)** to add a product to the catalog.
    - Ensure that the product name is unique and not already in the catalog.
4. **Remove Products:**
    - Implement a method **removeProduct(String productName)** to remove a product from the catalog.
5. **Search Products:**
    - Implement a method **searchProduct(String productName)** to check if a product exists in the catalog.
6. **Display Products:**
    - Implement a method **displayProducts()** to display all products in the catalog.
7. **Testing:**
    - Create a main class **ProductCatalogTest** with a main method.
    - Add, remove, and search for products using the **ProductCatalog** class.
    - Print the catalog contents to verify the functionality.

```
PS C:\Users\SENJUTI\OneDrive\srm\neha>  c:; cd 'c:\Users\SENJUTI\OneDrive\srm\neha'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorage\509d94736b74a36156d106c91c3ae94d\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'ProductCatalogTest'
Product Catalog:
Headphones
Laptop
Smartphone
Is Laptop in the catalog? true
Product Catalog:
Headphones
Laptop
PS C:\Users\SENJUTI\OneDrive\srm\neha>
```

Exercise 2: User Registration System with TreeSet

Objectives:

- Understand and use the TreeSet class.
- Store and retrieve elements in a sorted order.

Business Scenario:

You are building a user registration system where users' names need to be stored in alphabetical order.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **UserRegistration**.
2. **Create a UserRegistration Class:**
    - In the **UserRegistration** project, create a class named **UserRegistration**.
    - Use a **TreeSet<String>** to store users' names in alphabetical order.
3. **Register Users:**
    - Implement a method **registerUser(String userName)** to add a user to the registration system.
4. **Remove Users:**
    - Implement a method **removeUser(String userName)** to remove a user from the registration system.
5. **Display Users:**
    - Implement a method **displayUsers()** to display all registered users in alphabetical order.
6. **Testing:**
    - Create a main class **UserRegistrationTest** with a main method.

- Register, remove, and display users using the **UserRegistration** class.
- Verify that users are displayed in alphabetical order.

```
PS C:\Users\SENJUTI\OneDrive\srm\neha>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-agentlib:jdwp=tran
sport=dt_socket,server=n,suspend=y,address=localhost:49814' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp
' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorage\509d94736b74a36156d106c91c3ae94d\redhat.jav
a\jdt_ws\jdt.ls-java-project\bin' 'UserRegistrationTest'
Registered Users:
Alice
Bob
Charlie
Registered Users:
Alice
Charlie
PS C:\Users\SENJUTI\OneDrive\srm\neha>
```

Exercise 3: Managing Book Collection with LinkedHashSet

Objectives:

- Understand and use the LinkedHashSet class.
- Maintain insertion order of elements.

Business Scenario:

You are managing a book collection for a library. The collection should maintain the order in which books were added.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **BookCollection**.
2. **Create a BookCollection Class:**
    - In the **BookCollection** project, create a class named **BookCollection**.
    - Use a **LinkedHashSet<String>** to store book titles while preserving insertion order.
3. **Add Books:**
    - Implement a method **addBook(String bookTitle)** to add a book to the collection.
4. **Remove Books:**
    - Implement a method **removeBook(String bookTitle)** to remove a book from the collection.
5. **Display Books:**

- Implement a method **displayBooks()** to display all books in the collection in the order they were added.

6. **Testing:**
    - Create a main class BookCollectionTest with a main method.
    - Add, remove, and display books using the BookCollection class.
    - Verify that books are displayed in the order they were added.

```
PS C:\Users\SENJUTI\OneDrive\srm\neha>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetail
sInExceptionMessages' '-cp' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorage\509d94736b74a3615
6d106c91c3ae94d\redhat.java\jdt_ws\jdt.ls-java-project\bin' 'BookCollectionTest'
Book Collection:
1984
To Kill a Mockingbird
The Great Gatsby
Book Collection:
To Kill a Mockingbird
The Great Gatsby
PS C:\Users\SENJUTI\OneDrive\srm\neha>
```

Exercise 4: Employee Management System with List and ArrayList

Objectives:

- Understand and use the List interface and ArrayList class.
- Perform CRUD operations on an ArrayList.

Business Scenario:

You are building an employee management system to keep track of employees' names and IDs.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **EmployeeManagement**.
2. **Create an Employee Class:**
    - In the EmployeeManagement project, create a class named **Employee** with attributes **id (int), name (String) and address (String)**.
3. **Create an EmployeeManagement Class:**
    - Create a class named **EmployeeManagement** with an **ArrayList<Employee>** to store employees.
4. **Add Employees:**

- Implement a method **addEmployee(Employee employee)** to add an employee to the list.

5. **Remove Employees:**
    - Implement a method **removeEmployee(int employeeId)** to remove an employee by their ID.
6. **Update Employee Information:**
    - Implement a method **updateEmployee(int employeeId, String newAddress)** to update an employee's address.
7. **Display Employees:**
    - Implement a method **displayEmployees()** to display all employees.
8. **Testing:**
    - Create a main class **EmployeeManagementTest** with a main method.
    - Add, remove, update, and display employees using the EmployeeManagement class.

```
PS C:\Users\SENJUTI\OneDrive\Documents\Desktop\c                                  ...a\jdk-17\bin\java.exe' '
-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorag
e\72a5cd4abe3360c7f7cbaabad7406608\redhat.java\jdt_ws\cognizant_3b617e60\bin' 'Employee.EmployeeManagement
Test'
Employee List:
Employee [id=1, name=John Doe, address=123 Main St]
Employee [id=2, name=Jane Smith, address=456 Oak St]
Employee List:
Employee [id=1, name=John Doe, address=789 Pine St]
Employee [id=2, name=Jane Smith, address=456 Oak St]
Employee List:
Employee [id=1, name=John Doe, address=789 Pine St]
PS C:\Users\SENJUTI\OneDrive\Documents\Desktop\cognizant>
```

Exercise 5: Customer Order Tracking with LinkedList

Objectives:

- Understand and use the LinkedList class.
- Perform operations on a doubly linked list.

Business Scenario:

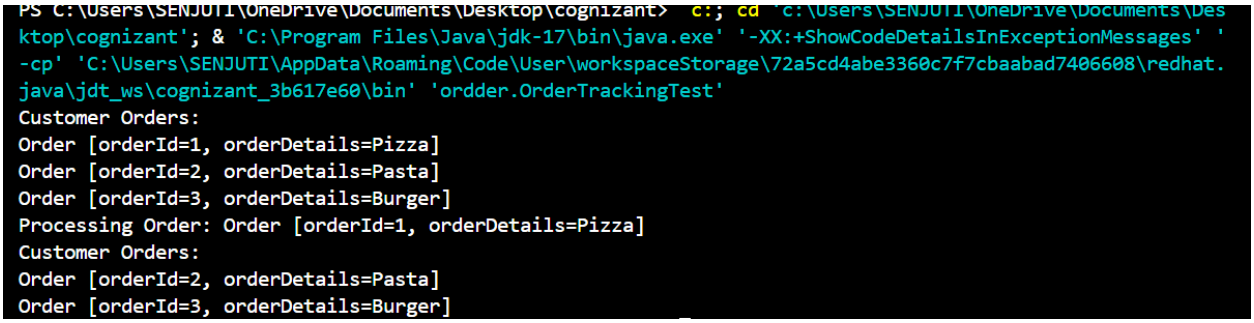
You need to track customer orders for a restaurant. The order list should allow for adding, processing, and displaying orders in a sequence.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **OrderTracking**.
2. **Create an Order Class:**

- In the OrderTracking project, create a class named Order with attributes **orderId (int)** and **orderDetails (String)**.

3. **Create an OrderTracking Class:**
    - Create a class named **OrderTracking** with a **LinkedList<Order>** to store customer orders.
4. **Add Orders:**
    - Implement a method **addOrder(Order order)** to add an order to the list.
5. **Process Orders:**
    - Implement a method **processOrder()** to remove and return the first order from the list (FIFO).
6. **Display Orders:**
    - Implement a method **displayOrders()** to display all orders in the list.
7. **Testing:**
    - Create a main class **OrderTrackingTest** with a main method.
    - Add, process, and display orders using the OrderTracking class.



Exercise 6: Inventory Management with Map and HashMap

Objectives:

- Understand and use the Map interface and HashMap class.
- Perform operations on a key-value pair collection.

Business Scenario:

You are developing an inventory management system for a store. Each product has a unique ID and associated details like name and quantity.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named InventoryManagement.
2. **Create a Product Class:**
    - In the **InventoryManagement** project, create a class named **Product** with attributes **id (int), name (String)**, and **quantity (int)**.
3. **Create an InventoryManagement Class:**
    - Create a class named **InventoryManagement** with a **HashMap<Integer, Product>** to store products.
4. **Add Products:**
    - Implement a method **addProduct(Product product)** to add a product to the inventory.
5. **Remove Products:**
    - Implement a method **removeProduct(int productId)** to remove a product by its ID.
6. **Update Product Quantity:**
    - Implement a method **updateProductQuantity(int productId, int newQuantity)** to update the quantity of a product.
7. **Display Products:**
    - Implement a method **displayProducts()** to display all products in the inventory.
8. **Testing:**
    - Create a main class **InventoryManagementTest** with a main method.
    - Add, remove, update, and display products using the InventoryManagement class.

```
PS C:\Users\SENJUTI\OneDrive\Documents\Desktop\cognizant>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '
-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorag
e\72a5cd4abe3360c7f7cbaabad7406608\redhat.java\jdt_ws\cognizant_3b617e60\bin' 'Inventory.InventoryManageme
ntTest'
Inventory:
Product [id=1, name=Laptop, quantity=10]
Product [id=2, name=Smartphone, quantity=20]
Inventory:
Product [id=1, name=Laptop, quantity=15]
Product [id=2, name=Smartphone, quantity=20]
Inventory:
Product [id=1, name=Laptop, quantity=15]
```

Exercise 7: Customer Accounts with TreeMap

Objectives:

- Understand and use the TreeMap class.
- Store and retrieve key-value pairs in a sorted order.

Business Scenario:

You are building a system to manage customer accounts. Each customer has an ID, and their information needs to be stored in a sorted order based on their ID.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **CustomerAccounts**.
2. **Create a Customer Class:**
    - In the **CustomerAccounts** project, create a class named **Customer** with attributes **id (int)**, **name (String)**, and **email (String)**.
3. **Create a CustomerAccounts Class:**
    - Create a class named CustomerAccounts with a **TreeMap<Integer, Customer>** to store customer accounts sorted by their ID.
4. **Add Customers:**
    - Implement a method **addCustomer(Customer customer)** to add a customer account to the system.
5. **Remove Customers:**
    - Implement a method **removeCustomer(int customerId)** to remove a customer account by its ID.
6. **Display Customers:**
    - Implement a method **displayCustomers()** to display all customer accounts in the system.
7. **Testing:**
    - Create a main class **CustomerAccountsTest** with a main method.
    - Add, remove, and display customer accounts using the CustomerAccounts class.

```
PS C:\Users\SENJUTI\OneDrive\Documents\Desktop\cognizant>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '
-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorag
e\72a5cd4abe3360c7f7cbaabad7406608\redhat.java\jdt_ws\cognizant_3b617e60\bin' 'Customer.CustomerAccountsTe
st'
Customer Accounts:
Customer [id=1, name=Alice, email=alice@example.com]
Customer [id=2, name=Bob, email=bob@example.com]
Customer Accounts:
Customer [id=2, name=Bob, email=bob@example.com]
PS C:\Users\SENJUTI\OneDrive\Documents\Desktop\cognizant>
```

Exercise 8: Student Grades with LinkedHashMap

Objectives:

- Understand and use the LinkedHashMap class.
- Maintain insertion order of key-value pairs.

Business Scenario:

You are developing a system to store and manage students' grades. The system should maintain the order in which students were added.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **StudentGrades**.
2. **Create a Student Class:**
    - In the **StudentGrades** project, create a class named **Student** with attributes **id (int)**, **name (String)**, and **grade (char)**.
3. **Create a StudentGrades Class:**
    - Create a class named **StudentGrades** with a **LinkedHashMap<Integer, Student>** to store students' grades while preserving insertion order.
4. **Add Students:**
    - Implement a method **addStudent(Student student)** to add a student and their grade to the system.
5. **Remove Students:**
    - Implement a method **removeStudent(int studentId)** to remove a student by their ID.
6. **Update Student Grades:**
    - Implement a method **updateStudentGrade(int studentId, char newGrade)** to update a student's grade.
7. **Display Students:**
    - Implement a method **displayStudents()** to display all students and their grades.
8. **Testing:**
    - Create a main class **StudentGradesTest** with a main method.

- Add, remove, update, and display students using the StudentGrades class.

```
ktop\cognizant'; & 'C:\Program Files\Java\jdk-17\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '
-cp' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorage\72a5cd4abe3360c7f7cbaabad7406608\redhat.
java\jdt_ws\cognizant_3b617e60\bin' 'student.StudentGradesTest'
Student Grades:
Student [id=1, name=John, grade=A]
Student [id=2, name=Jane, grade=B]
Student Grades:
Student [id=1, name=John, grade=A]
Student [id=2, name=Jane, grade=B]
Student Grades:
Student [id=1, name=John, grade=A]
```

Exercise 9: Contact Management with Hashtable

Objectives:

- Understand and use the Hashtable class.
- Perform thread-safe operations on a key-value pair collection.

Business Scenario:

You are building a contact management system to store and manage contact information for a company. Each contact has a unique ID and associated details.

Tasks:

1. **Create a New Java Project:**
    - Create a new Java project named **ContactManagement**.
2. **Create a Contact Class:**
    - In the **ContactManagement** project, create a class named **Contact** with attributes **id (int)**, **name (String)**, and **phoneNumber (String)**.
3. **Create a ContactManagement Class:**
    - Create a class named **ContactManagement** with a **Hashtable<Integer, Contact>** to store contacts.
4. **Add Contacts:**
    - Implement a method **addContact(Contact contact)** to add a contact to the system.
5. **Remove Contacts:**
    - Implement a method **removeContact(int contactId)** to remove a contact by its ID.
6. **Display Contacts:**
    - Implement a method **displayContacts()** to display all contacts in the system.
7. **Testing:**

- Create a main class **ContactManagementTest** with a main method.
- Add, remove, and display contacts using the ContactManagement class.

```
PS C:\Users\SENJUTI\OneDrive\Documents\Desktop\cognizant>  & 'C:\Program Files\Java\jdk-17\bin\java.exe' '
-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\SENJUTI\AppData\Roaming\Code\User\workspaceStorag
e\72a5cd4abe3360c7f7cbaabad7406608\redhat.java\jdt_ws\cognizant_3b617e60\bin' 'contact.ContactManagementTe
st'
Contact List:
Contact [id=2, name=Bob, phoneNumber=0987654321]
Contact [id=1, name=Alice, phoneNumber=1234567890]
Contact List:
Contact [id=2, name=Bob, phoneNumber=0987654321]
PS C:\Users\SENJUTI\OneDrive\Documents\Desktop\cognizant>
```