**Day-7 Basics of Java**

Problem Statement 1: Implement the flexible data structures using Collection

1. **package** com.mycom.collections.array;

**import** java.util.HashSet;

**import** java.util.Scanner;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

Scanner sc = **new** Scanner(System.***in***);

HashSet<String> students = **new** HashSet<>();

**while** (**true**) {

System.***out***.println("1. Add student name");

System.***out***.println("2. Check if student exists");

System.***out***.println("3. Exit");

System.***out***.print("Choose an option: ");

**int** option = sc.nextInt();

sc.nextLine(); // Consume newline left-over

**switch** (option) {

**case** 1:

System.***out***.print("Enter student name: ");

String name = sc.nextLine();

students.add(name);

**break**;

**case** 2:

System.***out***.print("Enter name to search: ");

String searchName = sc.nextLine();

**if** (students.contains(searchName)) {

System.***out***.println(searchName + " exists in the collection");

} **else** {

System.***out***.println(searchName + " does not exist in the collection");

}

**break**;

**case** 3:

System.*exit*(0);

**break**;

**default**:

System.***out***.println("Invalid option");

}

}

}

}

2. //Product

**package** com.mycom.collections.array;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.Scanner;

**import** java.util.Objects;

**class** Product {

**private** String productId;

**private** String productName;

**public** Product(String productId, String productName) {

**this**.productId = productId;

**this**.productName = productName;

}

**public** String getProductId() {

**return** productId;

}

**public** String getProductName() {

**return** productName;

}

@Override

**public** **boolean** equals(Object o) {

**if** (**this** == o) **return** **true**;

**if** (o == **null** || getClass() != o.getClass()) **return** **false**;

Product product = (Product) o;

**return** Objects.*equals*(productId, product.productId);

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(productId);

}

@Override

**public** String toString() {

**return** "Product Id: " + productId + ", Product Name: " + productName;

}

}

**public** **class** MainProduct {

**public** **static** **void** main(String[] args) {

HashSet<Product> products = **new** HashSet<>();

products.add(**new** Product("P001", "Maruti 800"));

products.add(**new** Product("P002", "Maruti Zen"));

products.add(**new** Product("P003", "Maruti Dezire"));

products.add(**new** Product("P004", "Maruti Alto"));

products.add(**new** Product("P005", "Maruti WagonR"));

products.add(**new** Product("P006", "Maruti Swift"));

products.add(**new** Product("P007", "Maruti Brezza"));

products.add(**new** Product("P008", "Maruti Baleno"));

products.add(**new** Product("P009", "Maruti Celerio"));

products.add(**new** Product("P010", "Maruti Ignis"));

System.***out***.println("Product List:");

**for** (Product product : products) {

System.***out***.println(product);

}

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("\nEnter a product ID to search: ");

String searchId = scanner.nextLine();

Iterator<Product> iterator = products.iterator();

**while** (iterator.hasNext()) {

Product product = iterator.next();

**if** (product.getProductId().equals(searchId)) {

System.***out***.println("Found: " + product);

**break**;

}

}

System.***out***.print("\nEnter a product ID to remove: ");

String removeId = scanner.nextLine();

iterator = products.iterator();

**while** (iterator.hasNext()) {

Product product = iterator.next();

**if** (product.getProductId().equals(removeId)) {

products.remove(product);

System.***out***.println("Product removed: " + product);

**break**;

}

}

System.***out***.println("\nUpdated Product List:");

**for** (Product product : products) {

System.***out***.println(product);

}

scanner.close();

}

}

3. // Employee class

**package** com.mycom.collections.array;

**public** **class** Employee {

**private** **int** employeeNo;

**private** String employeeName;

**private** String address;

**public** Employee(**int** employeeNo, String employeeName, String address) {

**this**.employeeNo = employeeNo;

**this**.employeeName = employeeName;

**this**.address = address;

}

**public** **int** getEmployeeNo() {

**return** employeeNo;

}

**public** String getEmployeeName() {

**return** employeeName;

}

**public** String getAddress() {

**return** address;

}

@Override

**public** String toString() {

**return** "EmployeeNo: " + employeeNo + ", EmployeeName: " + employeeName + ", Address: " + address;

}

}

// Main class

**package** com.mycom.collections.array;

**import** java.util.LinkedList;

**import** java.util.Scanner;

**public** **class** MainEmployee {

**public** **static** **void** main(String[] args) {

LinkedList<Employee> employees = **new** LinkedList<>();

Scanner scanner = **new** Scanner(System.***in***);

*addInput*(employees, scanner);

*display*(employees);

scanner.close();

}

**public** **static** **void** addInput(LinkedList<Employee> employees, Scanner scanner) {

System.***out***.print("Enter the number of employees: ");

**int** numEmployees = scanner.nextInt();

scanner.nextLine();

**for** (**int** i = 0; i < numEmployees; i++) {

System.***out***.print("Enter Employee Number: ");

**int** employeeNo = scanner.nextInt();

scanner.nextLine();

System.***out***.print("Enter Employee Name: ");

String employeeName = scanner.nextLine();

System.***out***.print("Enter Address: ");

String address = scanner.nextLine();

employees.add(**new** Employee(employeeNo, employeeName, address));

}

}

**public** **static** **void** display(LinkedList<Employee> employees) {

System.***out***.println("\nEmployee List:");

**for** (Employee employee : employees) {

System.***out***.println(employee);

}

}

}

4.

**package** com.mycom.collections.array;

**import** java.util.HashMap;

**import** java.util.Scanner;

**public** **class** PhoneBook {

**public** **static** **void** main(String[] args) {

HashMap<String, String> phoneBook = **new** HashMap<>();

Scanner scanner = **new** Scanner(System.***in***);

**boolean** running = **true**;

**while** (running) {

System.***out***.println("Phone Book Menu:");

System.***out***.println("1. Add new phone book entry");

System.***out***.println("2. Search Phone Number");

System.***out***.println("3. Quit");

System.***out***.print("Choose an option: ");

**int** choice = scanner.nextInt();

scanner.nextLine();

**switch** (choice) {

**case** 1:

*addEntry*(phoneBook, scanner);

**break**;

**case** 2:

*searchPhoneNumber*(phoneBook, scanner);

**break**;

**case** 3:

running = **false**;

System.***out***.println("Exiting the program.");

**break**;

**default**:

System.***out***.println("Invalid option. Please try again.");

}

}

scanner.close();

}

**public** **static** **void** addEntry(HashMap<String, String> phoneBook, Scanner scanner) {

System.***out***.print("Enter name: ");

String name = scanner.nextLine();

System.***out***.print("Enter phone number: ");

String phoneNumber = scanner.nextLine();

phoneBook.put(name, phoneNumber);

System.***out***.println("Entry added.");

}

**public** **static** **void** searchPhoneNumber(HashMap<String, String> phoneBook, Scanner scanner) {

System.***out***.print("Enter name to search: ");

String name = scanner.nextLine();

String phoneNumber = phoneBook.get(name);

**if** (phoneNumber != **null**) {

System.***out***.println("Phone number for " + name + " is " + phoneNumber);

} **else** {

System.***out***.println("No entry found for " + name);

}

}

}

5. // Book Class

**package** com.mycom.collections.array;

**import** java.util.\*;

**public** **class** Book **implements** Comparable<Book> {

**int** bookId;

String title;

**double** price;

Date dateOfPublication;

String author;

**public** Book(**int** bookId, String title, **double** price, Date dateOfPublication, String author) {

**this**.bookId = bookId;

**this**.title = title;

**this**.price = price;

**this**.dateOfPublication = dateOfPublication;

**this**.author = author;

}

**public** **int** getBookId() {

**return** bookId;

}

**public** **void** setBookId(**int** bookId) {

**this**.bookId = bookId;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** Date getDateOfPublication() {

**return** dateOfPublication;

}

**public** **void** setDateOfPublication(Date dateOfPublication) {

**this**.dateOfPublication = dateOfPublication;

}

**public** String getAuthor() {

**return** author;

}

**public** **void** setAuthor(String author) {

**this**.author = author;

}

@Override

**public** **int** compareTo(Book other) {

**return** **this**.author.compareTo(other.author);

}

@Override

**public** String toString() {

**return** "Book [bookId=" + bookId + ", title=" + title + ", price=" + price +

", dateOfPublication=" + dateOfPublication + ", author=" + author + "]";

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(bookId);

}

@Override

**public** **boolean** equals(Object obj) {

**if** (**this** == obj) **return** **true**;

**if** (obj == **null** || getClass() != obj.getClass()) **return** **false**;

Book other = (Book) obj;

**return** bookId == other.bookId;

}

}

//Main class

**import** java.util.\*;

**public** **class** MainBook {

**public** **static** **void** main(String[] args) {

TreeSet<Book> books = **new** TreeSet<>();

books.add(**new** Book(1003, "Java Programming", 523.0, **new** ~~Date~~(84, 11, 23), "Gilad Bracha"));

books.add(**new** Book(1004, "Read C++", 295.0, **new** ~~Date~~(84, 11, 9), "Henry Harvin"));

books.add(**new** Book(1005, ".Net Platform", 3497.0, **new** ~~Date~~(84, 11, 6), "Mark J. Price"));

books.add(**new** Book(1001, "Python Learning", 715.0, **new** ~~Date~~(120, 1, 2), "Martic C. Brown"));

books.add(**new** Book(1002, "Modern Mainframe", 295.0, **new** ~~Date~~(97, 4, 15), "Sharad"));

System.***out***.println("a. Sorting by Author Name in Ascending Order:");

**for** (Book book : books) {

System.***out***.println(book);

}

System.***out***.println("\n b. Sorting by Date of Publication in Descending Order:");

List<Book> listByDate = **new** ArrayList<>(books);

listByDate.sort(Comparator.*comparing*(Book::getDateOfPublication).reversed());

**for** (Book book : listByDate) {

System.***out***.println(book);

}

System.***out***.println("\n c. Sorting by Title of the Book in Ascending Order:");

List<Book> listByTitle = **new** ArrayList<>(books);

listByTitle.sort(Comparator.*comparing*(Book::getTitle));

**for** (Book book : listByTitle) {

System.***out***.println(book);

}

System.***out***.println("\n d. Sorting by Book ID in Descending Order:");

List<Book> listById = **new** ArrayList<>(books);

listById.sort(Comparator.*comparing*(Book::getBookId).reversed());

**for** (Book book : listById) {

System.***out***.println(book);

}

}

}

Problem Statement 2: Processing Data with Java SE 8 Streams

6. **// Person class**

**package** com.mycom.collections.array;

**import** java.util.\*;

**import** java.util.stream.\*;

**class** Person {

**int** id;

String name;

**int** age;

**double** salary;

Person(**int** id, String name, **int** age, **double** salary) {

**this**.id

= id;

**this**.name = name;

**this**.age = age;

**this**.salary = salary;

}

@Override

**public** String toString() {

**return** String.*format*("Person [id=%d, name=%s, age=%d, salary=%.1f]", id, name, age, salary);

}

@Override

**public** **boolean** equals(Object o) {

**if** (**this** == o) **return** **true**;

**if** (o == **null** || getClass() != o.getClass()) **return** **false**;

Person person = (Person) o;

**return** id ==

person.id

&& age == person.age && Double.*compare*(person.salary, salary) == 0 && Objects.*equals*(name, person.name);

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(id, name, age, salary);

}

}

**import** java.util.Arrays;

**import** java.util.Comparator;

**import** java.util.HashSet;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.Set;

**import** java.util.stream.Collectors;

**public** **class** MainPerson {

**public** **static** **void** main(String[] args) {

Set<Person> persons = **new** HashSet<>(Arrays.*asList*(

**new** Person(4, "Jones", 22, 6999.0),

**new** Person(6, "Tom", 42, 3999.0),

**new** Person(1, "Jerry", 12, 999.0),

**new** Person(5, "John", 32, 1999.0),

**new** Person(2, "Smith", 22, 2999.0),

**new** Person(3, "Popeye", 21, 5999.0)

));

// 6. Print all person records

System.***out***.println("\n All person record");

persons.forEach(System.***out***::println);

// 7. Print all persons sorted by id in ascending order

System.***out***.println("\n All person record sorted by id in ascending order");

persons.stream()

.sorted(Comparator.*comparingInt*(p -> p.id))

.forEach(System.***out***::println);

// 8. Print all persons sorted by name in ascending order

System.***out***.println("\n All person record sorted by name in ascending order");

persons.stream()

.sorted(Comparator.*comparing*(p -> p.name))

.forEach(System.***out***::println);

// 9. Print all persons sorted by name in descending order

System.***out***.println("\n All person record sorted by name in descending order");

persons.stream()

.sorted(Comparator.*comparing*((Person p) -> p.name).reversed())

.forEach(System.***out***::println);

// 10. Print all persons whose name starts with 'J'

System.***out***.println("\n All person record whose name starts with J");

persons.stream()

.filter(p -> p.name.startsWith("J"))

.forEach(System.***out***::println);

// 11. Count number of persons

System.***out***.println("\n Count of all person record");

**long** count = persons.stream().count();

System.***out***.println("Count number of persons: " + count);

// 12. Print the max salary among all persons

System.***out***.println("\n Person record who has max salary");

persons.stream()

.mapToDouble(p -> p.salary)

.max()

.ifPresent(max -> System.***out***.println("Max salary: " + max));

// 13. Print the min salary among all persons

System.***out***.println("\n Person record who has min salary");

persons.stream()

.mapToDouble(p -> p.salary)

.min()

.ifPresent(min -> System.***out***.println("Min salary: " + min));

// 14. Print the average of all salaries

System.***out***.println("\n The average of all salaries");

persons.stream()

.mapToDouble(p -> p.salary)

.average()

.ifPresent(avg -> System.***out***.println("Average salary: " + avg));

// 15. Print the sum of all salaries

System.***out***.println("\n Sum of all salary");

**double** sum = persons.stream()

.mapToDouble(p -> p.salary)

.sum();

System.***out***.println("Sum of all salaries: " + sum);

// 16. Print the first person whose name starts with 'J'

System.***out***.println("\n First person whose name starts with J");

persons.stream()

.filter(p -> p.name.startsWith("J"))

.findFirst()

.ifPresent(p -> System.***out***.println("First person whose name starts with 'J': " + p));

// 17. Check whether all the persons age is greater than 10 using Streams – allMatch method System.***out***.println("\n Person records whose ages are greater than 10");

**boolean** allAgesGreaterThan10 = persons.stream()

.allMatch(p -> p.age > 10);

System.***out***.println("All persons' ages are greater than 10: " + allAgesGreaterThan10);

// 18. Print the average of all salaries using Collectors

System.***out***.println("\n Average of all salaries using collector");

**double** averageSalary = persons.stream()

.collect(Collectors.*averagingDouble*(p -> p.salary));

System.***out***.println("Average salary using Collectors: " + averageSalary);

// 19. Group by salary

System.***out***.println("\n Group by salary");

Map<Double, List<Person>> groupedBySalary = persons.stream()

.collect(Collectors.*groupingBy*(p -> p.salary));

groupedBySalary.forEach((salary, people) -> {

System.***out***.println("Salary: " + salary);

people.forEach(System.***out***::println);

});

// 20. Print names of persons whose age is greater than 18

System.***out***.println("\n Person record whose age is greater than 18");

String names = persons.stream()

.filter(p -> p.age > 18)

.map(p -> p.name)

.collect(Collectors.*joining*(", "));

System.***out***.println("Names of persons whose age is greater than 18: " + names);

// 21. Check if no persons are older than 50

System.***out***.println("\n "

+ "Check if no person are older than 50");

**boolean** noneOlderThan50 = persons.stream()

.noneMatch(p -> p.age > 50);

System.***out***.println("No persons older than 50: " + noneOlderThan50);

}

}