**Problem Statement 1: Implement the flexible data structures using Collection.**

**1. Write a program to add list of student names to ArrayList and it should find a particular name whetherit exists or not in the list.**

**package** day7.com;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** Students\_List {

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

           ArrayList<String> StudentNames = **new** ArrayList<>();

           StudentNames.add("Vasudev");

           StudentNames.add("Samyukthaa");

           StudentNames.add("Afrin");

           StudentNames.add("Lasya");

           StudentNames.add("Shreya");

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

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

           String nameToSearch = scanner.nextLine();

**if** (studentNames.contains(nameToSearch)) {

                  System.***out***.println(nameToSearch + " is in the list.");

           } **else** {

                  System.***out***.println(nameToSearch + " is not in the list.");

           }

    }

}

**2. Create a Product class with Product Id & Product Name. Write a program to accept information of 10 products and store that in HashSet. Do following operations, a. Search a particular product in the HashSet. b. Remove a particular product from the HashSet by using product id. c. (Refer below table for the product list)**

**package** day7.com;

**import** java.util.HashSet;

**import** java.util.Iterator;

**class** Product {

    String productId;

    String productName;

    Product(String productId, String productName) {

**this**.productId = productId;

**this**.productName = 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** productId.equals(product.productId);

    }

    @Override

**public** **int** hashCode() {

**return** productId.hashCode();

    }

    @Override

**public** String toString() {

**return** "Product{" + "productId='" + productId + '\'' + ", productName='" + productName + '\'' + '}';

    }

}

**public** **class** ProductManagement {

**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"));

           // Search a particular product

           String searchId = "P002";

           Product searchProduct = **new** Product(searchId, "");

**if** (products.contains(searchProduct)) {

                  System.***out***.println(searchId + " is found in the set.");

           } **else** {

                  System.***out***.println(searchId + " is not found in the set.");

           }

           // Remove a particular product by productId

           String removeId = "P003";

           products.remove(**new** Product(removeId, ""));

           System.***out***.println("After removal:");

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

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

           }

    }

}

**3. Implement LinkedList class for this problem a. Create an Employee class which will have details like EmployeeNo, EmployeeName and Address. You should pass value for EmployeeNo, EmployeeName and Address through constructor. b. Create a method addInput( ) which will add employee details to LinkedList. c. Create method display( ) which should display all data from LinkedList using forward and reverse order using Iterator and ListIterator interfaces. d. Note: addInput( ) and display( ) should not be member functions of Employee class.**

**package** day7.com;

**import** java.util.LinkedList;

**import** java.util.ListIterator;

**import** java.util.Scanner;

**class** Employee {

**int** employeeNo;

    String employeeName;

    String address;

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

**this**.employeeNo = employeeNo;

**this**.employeeName = employeeName;

**this**.address = address;

    }

    @Override

**public** String toString() {

**return** "Employee{" + "employeeNo=" + employeeNo + ", employeeName='" + employeeName + '\'' + ", address='"

                        + address + '\'' + '}';

    }

}

**public** **class** EmployeeManagement {

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

**public** **void** addInput() {

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

           System.***out***.print("Enter Employee No: ");

**int** no = scanner.nextInt();

           scanner.nextLine();

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

           String name = scanner.nextLine();

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

           String address = scanner.nextLine();

           employeeList.add(**new** Employee(no, name, address));

    }

**public** **void** display() {

           System.***out***.println("Displaying in forward order:");

           ListIterator<Employee> iterator = employeeList.listIterator();

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

                  System.***out***.println(iterator.next());

           }

           System.***out***.println("Displaying in reverse order:");

**while** (iterator.hasPrevious()) {

                  System.***out***.println(iterator.previous());

           }

    }

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

           EmployeeManagement management = **new** EmployeeManagement();

           management.addInput();

           management.addInput();

           management.display();

    }

}

**4. Create a Phone Book having user interface like, a. Add new phone book entry b. Search Phone Number c. Quit Option i :it allows add name and Phone no. Option ii: it must take name as input from the user and based on that it should return phone No. Option iii: will terminate the program. Note: Use HashMap to store phone book entries.**

**package** day7.com;

**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***);

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

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

                  System.***out***.println("2. Search phone number by name");

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

**int** choice = scanner.nextInt();

                  scanner.nextLine();

**if** (choice == 1) {

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

                        String name = scanner.nextLine();

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

                        String number = scanner.nextLine();

                        phoneBook.put(name, number);

                  } **else** **if** (choice == 2) {

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

                        String name = scanner.nextLine();

**if** (phoneBook.containsKey(name)) {

                               System.***out***.println("Phone Number: " + phoneBook.get(name));

                        } **else** {

                               System.***out***.println("Name not found.");

                        }

                  } **else** **if** (choice == 3) {

**break**;

                  }

           }

    }

}

**5. Create a Book class with bookId, tile, price, date of publication and author. Override all the required methods such as toString, hashcode, equals, and compareTo. Implement natural ordering. Write a program that accepts information of 5 Book details and stores it in TreeSet.**

**package** day7.com;

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

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

**int** bookId;

    String title;

**double** price;

    String dop; // date of publication

    String author;

    Book(**int** bookId, String title, **double** price, String dop, String author) {

**this**.bookId = bookId;

**this**.title = title;

**this**.price = price;

**this**.dop = dop;

**this**.author = author;

    }

    @Override

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

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

    }

    @Override

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

**if** (**this** == o)

**return** **true**;

**if** (o == **null** || getClass() != o.getClass())

**return** **false**;

           Book book = (Book) o;

**return** bookId == book.bookId;

    }

    @Override

**public** **int** hashCode() {

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

    }

    @Override

**public** String toString() {

**return** "Book{" + "bookId=" + bookId + ", title='" + title + '\'' + ", price=" + price + ", dop='" + dop + '\''

                        + ", author='" + author + '\'' + '}';

    }

}

**public** **class** BookManagement {

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

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

           books.add(**new** Book(1003, "Java Programming", 523.8, "23/11/1984", "Gilad Bracha"));

           books.add(**new** Book(1004, "Read C++", 295.0, "19/11/1984", "Henry Harvin"));

           books.add(**new** Book(1005, ".Net Platform", 3497.0, "6/3/1984", "Mark J. Price"));

           books.add(**new** Book(1001, "Python Learning", 715.0, "2/2/2020", "Martic C. Brown"));

           books.add(**new** Book(1002, "Modern Mainframe", 295.0, "19/5/1997", "Sharad"));

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

**for** (Book book : books) {

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

           }

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

           TreeSet<Book> sortedByDop = **new** TreeSet<>(Comparator.*comparing*((Book b) -> b.dop).reversed());

           sortedByDop.addAll(books);

**for** (Book book : sortedByDop) {

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

           }

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

           TreeSet<Book> sortedByTitle = **new** TreeSet<>(Comparator.*comparing*(b -> b.title));

           sortedByTitle.addAll(books);

**for** (Book book : sortedByTitle) {

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

           }

           System.***out***.println("Sorting Book Id in Descending Order and Date of Publication in Ascending Order:");

           TreeSet<Book> sortedByIdAndDop = **new** TreeSet<>(

                        Comparator.*comparing*((Book b) -> b.bookId).reversed().thenComparing(b -> b.dop));

           sortedByIdAndDop.addAll(books);

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

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

           }

    }

}

Problem Statement 2: Processing Data with J... by Chinnari SaiChinnari Sai4:33 PM

**Problem Statement 2: Processing Data with Java SE 8 Streams Create a Person class with id, name, age, and salary and override all the required methods such as toString, hashcode, equals, and compareTo.**

**package** day7.com;

**import** java.util.Objects;

**class** Person **implements** Comparable<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;

    }

**public** String getName() {

**return** name;

    }

    @Override

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

**return** Integer.*compare*(**this**.id, other.id);

    }

    @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;

    }

    @Override

**public** **int** hashCode() {

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

    }

    @Override

**public** String toString() {

**return** "Person{id=" + id + ", name='" + name + '\'' + ", age=" + age + ", salary=" + salary + '}';

    }

}

**package** day7.com;

//Implement the Main Class with Required Functionalities

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

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

**public** **class** PersonManagement {

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

           Set<Person> persons = **new** HashSet<>();

           persons.add(**new** Person(1, "Jerry", 12, 999.0));

           persons.add(**new** Person(2, "Smith", 22, 2999.0));

           persons.add(**new** Person(3, "Popeye", 21, 5999.0));

           persons.add(**new** Person(4, "Jones", 22, 6999.0));

           persons.add(**new** Person(5, "John", 32, 1999.0));

           persons.add(**new** Person(6, "Tom", 42, 3999.0));

           // 06) Print all persons details using Streams and Method Reference

           System.***out***.println("10.Print all the person records:");

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

           // 07) Print all persons sorted by id using Comparable and Streams

           System.***out***.println("\nSorted Asc Id:");

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

           // 08) Print all persons sorted by name using Comparator and Streams

           System.***out***.println("\nSorted Asc Name:");

           persons.stream().sorted(Comparator.*comparing*(person -> person.name)).forEach(System.***out***::println);

           // 09) Print all persons sorted by name in descending order using Comparator and

           // Streams

           System.***out***.println("\nSorted Desc Name:");

     persons.stream().sorted(Comparator.*comparing*(Person::getName).reversed()).forEach(System.***out***::println);

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

           System.***out***.println("\nName start with J:");

           persons.stream().filter(person -> person.name.startsWith("J")).forEach(System.***out***::println);

           // 11)Print the count number of persons using Streams

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

           System.***out***.println("\n11.Count number of persons:");

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

           // 12)Print the maximum salary among all persons using Streams

**double** maxSalary = persons.stream().mapToDouble(person -> person.salary).max().orElse(0);

           System.***out***.println("\n12.Max salary among all persons:");

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

           // 13)Print the minimum salary among all persons using Streams

**double** minSalary = persons.stream().mapToDouble(person -> person.salary).min().orElse(0);

           System.***out***.println("\n13.Min salary among all persons:");

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

           // 14)Print the average of all salaries using Streams

**double** averageSalary = persons.stream().mapToDouble(person -> person.salary).average().orElse(0);

           System.***out***.println("\n14.Average of salaries:");

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

           // 15)Print the sum of all salaries using Streams

**double** sumSalary = persons.stream().mapToDouble(person -> person.salary).sum();

           System.***out***.println("\n15.Sum of all salaries:");

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

           // 16)Print the first person whose name starts with 'J' using Streams - filter

           // and

           // findFirst method

           System.***out***.println("\n16.First Person whose Name start with J:");

           persons.stream().filter(person -> person.name.startsWith("J")).findFirst().ifPresent(System.***out***::println);

           // 17)Check whether all persons age is greater than 10 using Streams - allMatch

           // method

**boolean** allAgeGreaterThan10 = persons.stream().allMatch(person -> person.age > 10);

           System.***out***.println("\n17.All persons age greater than 10:");

           System.***out***.println(allAgeGreaterThan10);

           // 18)Print the average of all salaries using Streams and Collectors

**double** averageSalaryCollector = persons.stream().collect(Collectors.*averagingDouble*(person -> person.salary));

           System.***out***.println("\n18.Average salaries to Double:");

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

           // 19)Print all persons details grouped by salary using Streams and Collectors

           System.***out***.println("\n19.Grouped By Salary:");

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

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

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

                  System.***out***.println("Person Grouped By: " + salary);

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

           });

           // 20)Print all names joined whose age is greater than 18 using Streams and

           // Collectors

           String namesJoined = persons.stream().filter(person -> person.age > 18).map(person -> person.name)

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

           System.***out***.println("\n20.Joining all the names whose age is greater than 18:");

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

           // 21)Check whether no person's age is greater than 50 using Streams - noneMatch

           // method

**boolean** noAgeGreaterThan50 = persons.stream().noneMatch(person -> person.age > 50);

           System.***out***.println("\n21.Return true if no person age greater than 50:");

           System.***out***.println(noAgeGreaterThan50);

    }

}

has context menu