**Basics of Java Practice Exercise – Day 7**

Submitted by: Aniket Singh (2605511)

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

**Code –**

**package** practiceday7;

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

**public** **class** StudentNames {

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

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

stu.add("Aniket");

stu.add("Kiran");

stu.add("Umesh");

stu.add("Raghu");

stu.add("Rachit");

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

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

String name=sc.nextLine();

**if**(stu.contains(name)) {

System.***out***.println("Name present in List");

}

**else** {

System.***out***.println("Name not in 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.**

**Code –**

**package** practiceday7;

**import** java.util.HashSet;

**import** java.util.Iterator;

**import** java.util.Scanner;

**class** Product{

String productId;

String productName;

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

**super**();

**this**.productId = productId;

**this**.productName = productName;

}

**public** String getProductName() {

**return** productName;

}

**public** **void** setProductName(String productName) {

**this**.productName = productName;

}

**public** String getProductId() {

**return** productId;

}

**public** **void** setProductId(String productId) {

**this**.productId = productId;

}

@Override

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

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

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

Product product=(Product) obj;

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

}

@Override

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

**return** productId.hashCode();

}

@Override

**public** String toString() {

**return** productName;

}

}

**public** **class** ProductDemo {

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

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

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

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

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

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

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

productArray.add(**new** Product("P005","Maruti Swift"));

productArray.add(**new** Product("P006","Tata Punch"));

productArray.add(**new** Product("P007","Tata Safari"));

productArray.add(**new** Product("P008","Tata Nexon"));

productArray.add(**new** Product("P009","Tata EV"));

productArray.add(**new** Product("P010","Mercedes Benz"));

System.***out***.println("Enter Product ID to Search: ");

String search=sc.nextLine();

**boolean** found=**false**;

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

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

System.***out***.println("Product Found");

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

found=**true**;

**break**;

}

}

**if**(!found) {

System.***out***.println("Product Not Found");

}

System.***out***.println("\nEnter Product ID to delete: ");

String removeId=sc.nextLine();

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

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

Product product=iterator.next();

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

iterator.remove();

System.***out***.println("Product with ID "+removeId+" deleted");

**break**;

}

}

}

}

**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.**

**Code –**

Employee.java

**package** practiceday7;

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

**int** employeeNo;

String employeeName,address;

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

**super**();

**this**.employeeNo = employeeNo;

**this**.employeeName = employeeName;

**this**.address = address;

}

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

**return** employeeNo;

}

**public** **void** setEmployeeNo(**int** employeeNo) {

**this**.employeeNo = employeeNo;

}

**public** String getEmployeeName() {

**return** employeeName;

}

**public** **void** setEmployeeName(String employeeName) {

**this**.employeeName = employeeName;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

@Override

**public** String toString() {

**return** "Employee Number: " + employeeNo + " Employee Name: " + employeeName + " Address: " + address;

}

}

EmployeeTest.java

package practiceday7;

import java.util.Iterator;

import java.util.LinkedList;

import java.util.ListIterator;

import java.util.Scanner;

public class EmployeeTest {

private LinkedList<Employee> emp;

public EmployeeTest() {

emp=new LinkedList<>();

}

void addInput(Employee employee) {

emp.add(employee);

}

void display() {

System.out.println("Employee Details in forward order:");

Iterator<Employee> iterator=emp.iterator();

while(iterator.hasNext()) {

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

}

System.out.println("\nEmployee Details in reverse order: ");

ListIterator<Employee> listIterator=emp.listIterator(emp.size());

while(listIterator.hasPrevious()) {

System.out.println(listIterator.previous());

}

}

public static void main(String[] args) {

EmployeeTest empTest=new EmployeeTest();

Scanner sc=new Scanner(System.in);

for(int i=0;i<5;i++) {

System.out.println("Enter Employee No.: ");

int employeeNo=sc.nextInt();

sc.nextLine();

System.out.println("Enter Employee Name: ");

String employeeName=sc.nextLine();

System.out.println("Enter Employee Address: ");

String address=sc.nextLine();

Employee employee =new Employee(employeeNo,employeeName,address);

empTest.addInput(employee);

}

empTest.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 phoneNo.**

**Option iii: will terminate the program.**

**Note: Use HashMap to store phone book entries**

**Code –**

PhoneBook.java

**package** practiceday7;

**import** java.util.HashMap;

**import** java.util.Scanner;

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

**private** HashMap<String,String> phoneBook;

**public** PhoneBook() {

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

}

**public** **void** addEntry(String name,String phoneNumber) {

phoneBook.put(name, phoneNumber);

System.***out***.println("Entry added: "+name+" : "+"phoneNumber");

}

**public** **void** searchPhoneNumber(String name) {

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

System.***out***.println("Found!!!");

System.***out***.println("Phone No.: "+phoneBook.get(name));

}

**else** {

System.***out***.println("!!!! No Entry Found !!!!");

}

}

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

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

PhoneBook phoneBook=**new** PhoneBook();

**boolean** quit=**false**;

**while**(!quit) {

System.***out***.println("1. Add new Phone Book Entry");

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

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

System.***out***.println("\nEnter Choice: ");

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

sc.nextLine();

**switch**(ch) {

**case** 1:

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

String name=sc.nextLine();

System.***out***.println("Enter Phone Number");

String phoneNumber=sc.nextLine();

phoneBook.addEntry(name, phoneNumber);

**break**;

**case** 2:

System.***out***.println("Enter Name to Search: ");

String searchName=sc.nextLine();

phoneBook.searchPhoneNumber(searchName);

**break**;

**case** 3:

quit=**true**;

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

**break**;

**default**:

System.***out***.println("Wrong Choice");

}

}

}

}

**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. Do following operations,**

**a. Print all the Book details by sorting the author names in ascending order using Comparable.**

**b. Print all the Book details by sorting the date of publication in descending order using**

**Comparator.**

**c. Print all the Book details by sorting the title of the book in ascending order using**

**Comparator.**

**d. Print all the Book details by sorting the bookid in descending order and date of publication in ascending order using Comparator.**

**Code –**

Book.java

**package** practiceday7;

**import** java.util.Date;

**import** java.util.Objects;

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

String bookId;

String title;

**double** price;

Date dop;

String author;

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

**super**();

**this**.bookId = bookId;

**this**.title = title;

**this**.price = price;

**this**.dop = dop;

**this**.author = author;

}

**public** String getBookId() {

**return** bookId;

}

**public** **void** setBookId(String 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 getDop() {

**return** dop;

}

**public** **void** setDop(Date dop) {

**this**.dop = dop;

}

**public** String getAuthor() {

**return** author;

}

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

**this**.author = author;

}

@Override

**public** String toString() {

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

+ author + "]";

}

@Override

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

**return** Objects.*hash*(bookId,title,price, dop,author );

}

@Override

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

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Book book = (Book) obj;

**return** Objects.*equals*(author, book.author) && Objects.*equals*(bookId, book.bookId)

&& Objects.*equals*(dop, book.dop)

&& Double.*doubleToLongBits*(price) == Double.*doubleToLongBits*(book.price)

&& Objects.*equals*(title, book.title);

}

@Override

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

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

}

}

BookMain.java

**package** practiceday7;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.util.Comparator;

**import** java.util.TreeSet;

**public** **class** BookMain {

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

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

SimpleDateFormat dateFormat=**new** SimpleDateFormat("yyyy-MM-dd");

booksByAuthor.add(**new** Book("1003","Java Programming",523.0,dateFormat.parse("1984-11-23"),"Gilad Bracha"));

booksByAuthor.add(**new** Book("1004","Read C++",295.0,dateFormat.parse("1984-11-19"),"Henry Harvin"));

booksByAuthor.add(**new** Book("1005",".Net Platform",3497.0,dateFormat.parse("1984-03-06"),"Mark J Price"));

booksByAuthor.add(**new** Book("1001","Python Learning",715.0,dateFormat.parse("2020-02-02"),"Martin C Brown"));

booksByAuthor.add(**new** Book("1002","Modern Mainframe",295.0,dateFormat.parse("1991-05-19"),"Sharad"));

System.***out***.println("Books by ascending Author Names: ");

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

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

}

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

TreeSet<Book> booksByDateDesc=**new** TreeSet<>(Comparator.*comparing*(Book::getDop).reversed());

booksByDateDesc.addAll(booksByAuthor);

System.***out***.println("Books by descending publication date: ");

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

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

}

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

TreeSet<Book> booksByTitleAsc=**new** TreeSet<>(Comparator.*comparing*(Book::getTitle));

booksByTitleAsc.addAll(booksByAuthor);

System.***out***.println("Books by descending publication date: ");

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

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

}

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

TreeSet<Book> booksByIdandDate=**new** TreeSet<>(Comparator.*comparing*(Book::getBookId).reversed().thenComparing(Book::getDop));

booksByIdandDate.addAll(booksByAuthor);

System.***out***.println("Books sorted by book ID in descending and date of publication in ascending order:");

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

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

}

}

}

**Create a Person class with id, name, age, and salary and override all the required methods such as toString, hashcode, equals, and compareTo. Write a program to accept information of 6 person details and store that in HashSet. Do following operations,**

**6. Print all the persons details using the Streams and Method Reference features.**

**7. Print all the persons details by sorting the id in ascending order using Comparable and Streams.**

**8. Print all the persons details by sorting the name in ascending order using Comparator and Streams.**

**9. Print all the persons details by sorting the names in descending order using Comparator and Streams.**

**10. Print all the persons details whose Name start with J using Streams.**

**11. Print the count number of persons using Streams.**

**12. Print the Max salary among all persons using Streams.**

**13. Print the Min salary among all persons using Streams.**

**14. Print the average of all salaries using Streams.**

**15. Print the sum of all salaries using Streams.**

**16. Print the First Person whose Name start with J using Streams - filter and findFirst method.**

**17. Check whether all the persons age is greater than 10 using Streams – allMatch method.**

**18. Print the average of all salaries using Streams and Collectors.**

**19. Print all the persons details group by salary using Streams and Collectors.**

**20. Print all the names after joining whose age is greater than 18 using Streams and**

**Collectors.**

**21. Check whether all the persons age is greater than 50 using Streams – noneMatch method.**

**Code –**

Person.java

**package** practiceday7;

**import** java.util.Objects;

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

**int** id;

String name;

**int** age;

**double** salary;

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

**super**();

**this**.id = id;

**this**.name = name;

**this**.age = age;

**this**.salary = salary;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) {

**this**.age = age;

}

**public** **double** getSalary() {

**return** salary;

}

**public** **void** setSalary(**double** salary) {

**this**.salary = salary;

}

@Override

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

// **TODO** Auto-generated method stub

**return** 0;

}

@Override

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

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

}

@Override

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

**if** (**this** == obj)

**return** **true**;

**if** (obj == **null**)

**return** **false**;

**if** (getClass() != obj.getClass())

**return** **false**;

Person other = (Person) obj;

**return** age == other.age && id == other.id && Objects.*equals*(name, other.name)

&& Double.*doubleToLongBits*(salary) == Double.*doubleToLongBits*(other.salary);

}

@Override

**public** String toString() {

**return** "Person [id=" + id + ", name=" + name + ", age=" + age + ", salary=" + salary + "]";

}

}

PersonMain.java

**package** practiceday7;

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

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

**public** **class** PersonMain {

**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)

));

System.***out***.println("----------Print all person records-----------");

persons.stream()

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

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

System.***out***.println("----------Sorted Asc Id--------------");

persons.stream()

.sorted(Comparator.*comparing*(Person::getId))

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

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

System.***out***.println("----------Sorted Asc Name--------------");

persons.stream()

.sorted(Comparator.*comparing*(Person::getName))

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

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

System.***out***.println("----------Sorted Desc Name--------------");

persons.stream()

.sorted(Comparator.*comparing*(Person::getName).reversed())

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

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

System.***out***.println("----------Name start with J--------------");

persons.stream()

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

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

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

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

System.***out***.println("-----------Count number of persons----------");

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

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

System.***out***.println("----------Max salary among all persons------");

persons.stream()

.map(Person::getSalary)

.max(Double::*compare*)

.ifPresent(maxSalary->System.***out***.println(maxSalary));

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

System.***out***.println("----------Min Slary among all persons-------");

persons.stream()

.map(Person::getSalary)

.min(Double::*compare*)

.ifPresent(minSalary->System.***out***.println(minSalary));

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

System.***out***.println("---------Average of salaries---------");

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

.mapToDouble(Person::getSalary)

.average()

.orElse(0);

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

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

System.***out***.println("--------Sum of all salaries----------");

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

.mapToDouble(Person::getSalary)

.sum();

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

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

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

persons.stream()

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

.findFirst()

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

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

System.***out***.println("------Return true if All person age greater than 10------");

**boolean** allAbove10 = persons.stream()

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

System.***out***.println(allAbove10);

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

System.***out***.println("--------Average salaries to Double----------");

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

.collect(Collectors.*averagingDouble*(Person::getSalary));

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

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

System.***out***.println("---------Group By Salary----------");

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

.collect(Collectors.*groupingBy*(Person::getSalary));

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

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

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

});

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

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

String names = persons.stream()

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

.map(Person::getName)

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

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

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

System.***out***.println("----Return true if All perosn age greater than 50--------");

**boolean** noneAbove50 = persons.stream()

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

System.***out***.println(noneAbove50);

}

}