Java Hands-on Mastery – S1 - Practice Exercise

Practice Exercise

This document provides a list of exercises to be practiced by learners. Please raise feedback in Talent Next, should you have any queries.

Skill	Java Hands-on Mastery
Proficiency	S1
Document Type	Lab Practice Exercises
Author	L & D
Current Version	1.0
Current Version Date	01-Apr-2024
Status	Active

Java Hands-on Mastery – S1 - Practice Exercise

Document Control

Version	Change Date	Change Description	Changed By
1.0	01-Apr-2024	Baseline version	Vimala R

Java Hands-on Mastery – S1 - Practice Exercise

Contents

Java Hands-on Mastery – S1 - Practice Exercise

Instructions Set:

1. If you do not have a GitHub account, please create one using your personal email id. Share the GitHub id as instructed by the trainer.
2. In your GitHub account, create a new public repository with the name "**java-core**". This repository will hold your assignment solution code for the entire duration of the course.
3. Clone the empty repository to your laptop.
4. In this empty repository, create a java project as instructed by the trainer.
5. Use the naming conventions as described below.
6. Code and test the solution locally.
7. Commit and push the updated code to GitHub repository before 0900 IST the next day.

Naming convention:

1. Create one package per day (e.g., com.learning.core. day2session1).
2. Main class names should be of the form "**DmmPnn**", where mm is day number and nn is the problem number as provided in the document. For example, on **Day2,** you can create two primary classes for Problem Statements 1 and 2: **D02P01** and **D02P02**. There could be other supporting classes too.

Java Hands-on Mastery – S1 - Practice Exercise

Note: Every Problem Statement starts on a new page

Problem Statement 1: Classes and Objects in Java

Solve following sub problems,

1. Create a class **Book** which describes its **book_title** and **book_price**. Follow the below steps,
    - Use getter and setter methods to get & set the Books description.
    - Implement createBooks and showBooks methods to create an object of Book.

    Note: createBooks & showBooks should not be member functions of **Book** class.

    Input:

    Java Programming

    350.00

    Output:

    Book Title: Java Programming and Price: 350.00

Problem Statement 2: Runtime Polymorphism in Java OOPs

Solve following sub problems,

1. Create an interface **MedicineInfo** to represent a drug manufactured by a pharmaceutical company. Provide an abstract method displayLabel(). Do following tasks,

    - Implement MedicineInfo interface with Tablet, Syrup and Ointment classes.
    - Override the displayLabel() method in each of these classes to print information suitableto the type of medicine. For example, in case of tablets, it could be "store in a cool dry place", in case of ointments it could be "for external use only" etc.

    Create a class **TestClass** . Write main function to do the following:

    - Declare Medicine instances.
    - Check the polymorphic behavior of the displayLabel() method.

    Output:

    Store Tablets in a cool dry place.
    Syrup is consumable only on prescription.
    Ointment is for external use only.

Problem Statement 3: Control Structures in Java

1. Write a program that takes a number from the user between 1 to 12 and displays the name of the month.

   Input:
   3
   -2
   14
   Output:
   March
   Invalid Input
   Invalid Input

2. Write a Program to print weekday for a given input week using if-else.

   Input:
   2
   -4
   9
   Output:
   Tuesday
   Invalid Input
   Invalid Input

3. Write a Program to check the grade based on marks obtained by students.
   For Example:
   Percentage >= 60% : Grade A.
   Percentage >= 45% : Grade B.
   Percentage >= 35% : Grade C.
   Percentage < 35% : Fail.

   Input:
   77
   Output:
   A Grade

4. Write a Program to display the pattern like right angle triangle with a number.

   Input:
   5
   Output:
   1
   2 2
   3 3 3
   4 4 4 4
   5 5 5 5 5

5. Write a program to find the factorial value of any number entered through the keyboard.

   Input:
   5
   Output:
   Factorial of 5 is 120

Problem Statement 4: Array Programs in Java

Solve following sub problems,

1. Write a program that accepts two numbers in the range from 1 to 40 from the user. Then compare these numbers against a single dimension array of five integer elements ranging in value from 1 to 40. The program displays the message "Bingo" if both the inputted values are found in the array element.

   Input:
   3 29
   7 25 5 19 30
   8 17
   9 17 3 39 8
   Output:
   Not Found
   Its Bingo

2. Write a program that allows you to create an integer array of 18 elements with the following values:

   int A[] = {3, 2, 4, 5, 6, 4, 5, 7, 3, 2, 3, 4, 7, 1, 2, 0, 0, 0}.

   Perform the following computations,

   a. Compute the sum of elements from index 0 to 14 and stores it at element 15.
   b. Compute the average of all numbers and stores it at element 16.
   c. Identifies the smallest value from the array and stores it at element 17.

   Input:
   2 4 5 6 4 5 7 3 2 3 4 7 1 2 3 0 0 0
   Output:
   2 4 5 6 4 5 7 3 2 3 4 7 1 2 3 58 4 1

3. Given an Array arr[] of size N. Write a Program to find the index of first repeating element.

   Input:
   7
   10 5 3 4 3 5 6
   Output:
   5

4. Given an array find all the distinct combinations of numbers according to given value of k.

   Input:
   1 2 3
   2
   Output:
   1 2 , 1 3 , 2 3

Problem Statement 5: String Programs in Java

Solve following sub problems,

1. Write a program that takes a String as input and display the length of the string. Also display the string into uppercase and check whether it is a palindrome or not. (Refer Java API Documentation)

   Input:
   madam
   teaching
   Output:
   It is a Palindrome
   It is not a Palindrome.

2. Write a program that takes String as input and count the last 'n' vowels of a given String. If the number is greater than the vowels found, then print 'Mismatch in Vowel Count
   Input:
   Testing
   2
   Output:
   ei

3. Write a program to print all subsequences of a string.

   Input:
   abc
   Output:
   a, b, c, ab, bc, ac, abc

4. Write a program to find minimum number of deletions to make a string palindrome.

   Input:
   aebcbda
   Output :
   2

5. Given an array of strings, find if the given strings can be chained to form a circle. A string X can be put before another string Y in a circle if the last character of X is the same as the first character of Y.

   Input:
   5
   abc efg cde ghi ija
   Output:
   Yes
   abc cde efg ghi ija
   Input:
   Ijk kji abc cba
   Output:
   No

6. Given a string **s**, find the length of the longest prefix, which is also a suffix. The prefix and suffix should not overlap.

   Input :
   aabcdaabc
   Output : 4
   Input :

abcab
Output : 2

7. Find all strings that match specific pattern in a dictionary.

   Input:
   abb abc xyz xyy
   foo
   Output:
   xyy abb

8. Write a program to check if given string can be split into four distinct strings.

   Input:
   Helloworld
   Output:
   Yes

   Input:
   aaabb
   Output:
   No

9. Write a program to replace all the spaces in a string with '%20'.

   Input:
   Mr John Smith
   Output:
   Mr%20John%20Smith

Problem Statement 6: Exception Handling in Java

Solve following sub problems,

1. Create a class **BankAccount** having the members as given below:
   Field Names:
   - accNo (int)
   - custName (String)
   - accType (String)
   - balance (float)

   Method Description:
   - **public void deposit(float amt)** : This method allows you to credit an amount into the current balance. If amount is negative, throw an exception **NegativeAmount** toblock the operation from being performed.
   - **public float getBalance()** : This method returns the current balance. If the current balance is below the minimum required balance, then throw an exception **LowBalanceException** accordingly.

Have a constructor to which you will pass, accNo, custName, acctype and initial balance. And check whether the balance is less than 1000 or not in case of savings account and less than 5000 in case of a current account.

If so, then raise a **LowBalanceException**. In either case if the balance is negative then raise the **NegativeAmount** exception accordingly.

Input 1:
123 John Saving 900
Output 1:
LowBalance

Input 2:
123 John Saving -900
Output 2:
NegativeAmount

2. WestCity Union is a cricket club that maintains an average rating of the players and provides them with coins based on the rating obtained by the critics. The club can only have three critics. Create a class called CricketRating with members as given below:

Field Names:

- playerName (String)
- critic1 (float)
- critic2 (float)
- critic3 (float
- avgRating (float)
- Coins (String)

Method Description:

- void calculateAverageRating(critic1,critic2) : This method Calculates Rating based on twocritics.
- void calculateAverageRating(critic1,critic2,critic3) : This method Calculates Rating based on three critics.
- String calculateCoins() : This method returns the type of coin achieved by the player basedon the rating.
- void display() : This method displays all the information.

The type of coin achieved by the player based on the rating is given below:

- If the avgRating is greater than or equal to 7 then the player gains **gold** coin.
- If the avgRating is greater than or equal to 5 and less than 7 then the player gains **silver** coin.
- If the avgRating is greater than or equal to 2 and less than 5 then the player gains **copper** Coin.
- If the avgRating is less than 2 then throw a **NotEligibleException**.

Provide appropriate constructor(s) that accept values to be passed to the attributes.

Implement the Tester class.

Input 1:
John 9.3 9.67 8.7
Output 1:
John 9.22 Gold

Input 2:
Henry 1.5
Output 2:
NotEligible

Problem Statement 7: 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 whether it exists or not in the list.
   Input 1:
   Jack Paul Henry Mary Ronaldo
   Henry
   Output 1:
   Found
2. Create a Product class with Product Id & Product Name. Write a program with predefined information of 4 products and store that in a HashSet. Do the following operation, List all the Product details.

   Output:
   P001 Maruti 800
   P002 Maruti Zen
   P003 Maruti Dezire
   P004 Maruti Alto
3. Create a Product class with Product Id & Product Name. Write a program with predefined information of 4 products and store that in a HashSet. Do the following operation, Search a particular product in a HashSet.

   Input:
   P003 Maruti Dezire
   Output:
   Product Found
4. Create a Product class with Product Id & Product Name. Write a program with predefined information of 4 products and store that in a HashSet. Do the following operation, remove a particular product from the HashSet by using product id.

   Input:
   P002
   Output:
   P001 Maruti 800
   P003 Maruti Dezire
   P004 Maruti Alto
5. Create a Person class with id, name, age, and salary with the natural sorting using id and override all the required methods such as toString, hashcode, equals, and compareTo. Write a program that has predefined information of 6 person details and store it in TreeSet. Do the following operation, print all the Persons details whose age is greater than 25.

   Output:
   Id=5, name=John, age=32, salary=1999.0
   Id=6, name=Tom, age=42, salary=3999.0

6. Create a Person class with id, name, age, and salary with the natural sorting using id and override all the required methods such as toString, hashcode, equals, and compareTo. Write a program that has predefined information of 6 person details and store it in TreeSet. Do the following operation, print the id, name, and salary of each person.

   Output:
   1 Jerry 999.0
   2 Smith 2999.0
   3 Popeye 5999.0
   4 Jones 6999.0
   5 John 1999.0
   6 Tom 3999.0

7. Create a Person class with id, name, age, and salary with the natural sorting using id and override all the required methods such as toString, hashcode, equals, and compareTo. Write a program that has predefined information of 6 person details and store it in TreeSet. Do the following operation, print all the names of person in uppercase.

   Output:
   Jerry
   Smith
   Popeye
   Jones
   John
   Tom

8. Create a Person class with id, name, age, and salary with the natural sorting using id and override all the required methods such as toString, hashcode, equals, and compareTo. Write a program that has predefined information of 6 person details and store it in TreeSet. Do the following operation, print the sum of all salaries.

   Output:
   22994.0

9. Create a Person class with id, name, age, and salary with the natural sorting using id and override all the required methods such as toString, hashcode, equals, and compareTo. Write a program that has predefined information of 6 person details and store it in TreeSet. Do the following operation, print the First Person whose name starts with J.

   Output:
   Id=4, name=Jones, age=22, salary=6999.0

Problem Statement 8: Implement the flexible data structures using Maps.

1. Create a Phone Book having details name and Phone no. Write a program with predefined information of 5 phone book details and store that in a HashMap. Do the following operation, List all the Phone Book details.
   Output:
   Amal 998787823
   Manvitha 937843978
   Joseph 7882221113
   Smith 8293893311
   Kathe 7234560011

2. Create a Phone Book having details name and Phone no. Write a program with predefined information of 5 phone book details and store that in a HashMap. Do the following operation, to search the phone number.
   Input:
   Joseph
   Output:
   7882221113

3. 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 with predefined information of 5 Book details and store it in TreeSet. Do the following operations, print all the Book details.
   Output:
   1001 Python Learning 715.0 Martic C. Brown 2/2/2020
   1002 Modern Mainframe 295.0 Sharad 19/5/1997
   1003 Java Programming 523.0 Gilad Bracha 23/11/1984
   1004 Read C++ 295.0 Henry Harvin 19/11/1984
   1005 .Net Platform 3497.0 Mark J. Price 6/3/1984

4. 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 with predefined information of 5 Book details and store it in TreeSet. Do the following operations, print all the Book details by sorting the author names in ascending order using Comparable.
   Output:
   1003 Java Programming 523.0 Gilad Bracha 23/11/1984
   1004 Read C++ 295.0 Henry Harvin 19/11/1984
   1005 .Net Platform 3497.0 Mark J. Price 6/3/1984
   1001 Python Learning 715.0 Martic C. Brown 2/2/2020
   1002 Modern Mainframe 295.0 Sharad 19/5/1997

5. Create a Car with price and name. Override all the required methods such as toString, hashcode, equals, and compareTo. Implement natural ordering. Write a program with predefined information of 4 Car details and store it in a TreeMap.
   Do the following operations, retrieve all car details.
   Output:
   Bugatti 80050.0
   Swift 305000.0
   Audi 600100.0
   Benz 900000.0

6. Create a Car with price and name. Override all the required methods such as toString, hashcode, equals, and compareTo. Implement natural ordering. Write a program with predefined information of 4 Car details and store it in a TreeMap.
   Do the following operations, retrieve the price of the car in reverse order.
   Output:
   Benz 900000.0
   Audi 600100.0
   Swift 305000.0
   Bugatti 80050.0

7. Create a Car with price and name. Override all the required methods such as toString, hashcode, equals, and compareTo. Implement natural ordering. Write a program with predefined information of 4 Car details and store it in a TreeMap.
   Do the following operations, retrieve the key-value mapping associated with the greatest and the least price.
   Output:
   Benz 900000.0
   Bugatti 80050.0

8. Create a Car with price and name. Override all the required methods such as toString, hashcode, equals, and compareTo. Implement natural ordering. Write a program with predefined information of 4 Car details and store it in a TreeMap.
   Do the following operations, remove, and get a key-value mapping associated with the greatest key.

   Output:
   Bugatti 80050.0
   Swift 305000.0
   Audi 600100.0Create a Car with price and name. Override all the required methods such as toString, hashcode, equals, and compareTo. Implement natural ordering. Write a program with predefined information of 4 Car details and store it in a TreeMap.
   Do the following operations, replace the value mapped by the specified key with the new value.
   Input:
   80050.0
   Output:
   Reva 80050.0

9. Create an Employee with id, name, department, and designation. Override all the required methods such as toString, hashcode and equals. Write a program with predefined information of 4 Employee details and store it in a HashTable and use employee id as key.
   Do the following operations, verify whether the HashTable is empty or not.

   Output:
   false

10. Create an Employee with id, name, department, and designation. Override all the required methods such as toString, hashcode and equals. Write a program with predefined information of 4 Employee details and store it in a HashTable and use employee id as key.
    Do the following operations, search for a specific Employee.
    Input:
    1003
    Output:
    1003 Robert Product Manager Development

11. Create an Employee with id, name, department, and designation. Override all the required methods such as toString, hashcode and equals. Write a program with predefined information of 4 Employee details and store it in a HashTable and use employee id as key.
    Do the following operations, add some Employees if not exists.
    Input:
    1005 Charles QA Lead Testing
    Output:
    1005 Charles QA Lead Testing
    1004 Grace Tech Support HR
    1003 Robert Product Manager Development
    1002 Thomas Tester Testing
    1001 Daniel Analyst L&D

Java Hands-on Mastery – S1 - Practice Exercise

Problem Statement 9: Working with Stacks & Queues

1. Write a Program to implement Stack using Array.

   Output:
   After Pushing 4 Elements: Hello world java Programming
   After a Pop: Hello world java

2. Write a Program to implement Stack using Linked List.

   Input:
   10.0 20.0 30.0 40.0
   Output:
   The elements of the stack are: 40.0 30.0 20.0 10.0 null
   After popping twice: 20.0 10.0 null

3. Write a Program to evaluate an Expression using Stacks.

   Input:
   10 + 2 * 6
   Output:
   22

4. Write a Program to implement Queue using Array.

   Output:
   Elements in queue: 10 20 30 40
   After removing first element: 20 30 40

5. Write a Program to implement Queue using Linked List.

   Output:
   Elements in queue: 89 99 109 209 309
   After removing two elements: 109 209 309

6. Write a Program to Implement Circular Queue using Array.

   Output:
   Elements in circular queue: 14 13 22 -8
   After removing first element: 13 22 -8

Problem Statement 10: Sorting & Searching Algorithms in Java

1. Write a Program to sort an array of given integers using the Quick sort.

   Input:
   6
   10 7 8 9 1 5
   Output:
   1 5 7 8 9 10

2. Write a Program to sort an array using Merge sort algorithm.

   Input:
   6
   12 11 13 5 6 7
   Output:
   5 6 7 11 12 13

3. Write a Program to find a specific element in each array of elements using Linear Search.

   Input:
   5
   2 3 4 10 40
   10
   Output
   Element is Present

   Input:
   5
   2 3 4 10 40
   90
   Output
   Element is not Present.