Java Language Exercises

Language Elements

1. Accept two numbers and display whether they are divisible by 5.
2. Write a program to take month and year and display number of days in the month.
3. Accept height in CMs and weight in KGs and display BMI (Body Mass Index).
4. Write a program to take a number and display whether it is prime number or not.
5. Write a program to take two integers and display GCD.
6. Write a program to take a number and display sum of digits.
7. Accept a number and display the smallest factor other than 1. For prime numbers, display number itself as smallest factor.
8. Write a program to take 10 integers and display the biggest of the given numbers.
9. Accept day of a week and number of hours worked and calculate the wage:

   1,2,3 - Rs200, 4, 5 - Rs400, 6 - Rs600, 7 - Rs800

   1 to 7 is Monday to Sunday.

   If total amount exceeds Rs2000 then there will be a bonus of 10%.

Arrays and Functions

1. Write a program to create an array, read values from keyboard and display array in reverse order.
2. Create an array of 10 elements, fill it with random numbers and display even numbers first and then odd numbers.
3. Write a function that takes a collection of numbers using varying args and displays the largest of all numbers.
4. Write a function that takes a number and returns largest factor of the number other than itself.
5. Write a function that takes an array of 10 integers and returns the average of positive numbers.

OOP - Encapsulation, Constructors and Overloading

1. Create a class to store details (name, price and qoh) of a **Product** and provide methods like print(), getNetPrice(), setPrice(), sale(int), purchase(int).
2. Create a class **Student** to store details of student like Admno, Name, Fee Paid, Course. Course is 1 for Java and 2 for Python. Total fee is 12000 for Java and 10000 for Python.

   Provide overloaded constructors, methods - payment(), getDue(), getTotalFee(), getFeePaid(), getCourseName() etc.

3. Create a class **Counter** that stores counter and provides methods increment(), decrement() and getValue().
4. Create a class Interest that stores amount, interest rate and provide the following methods:
   - Constructor to take both amount and interest rate
   - Constructor to take only amount and assume 10% for interest rate
   - Methods getRate(), setRate(), getAmount(), setAmount(), getInterest()

Static and Final members, Records

1. Add static variable taxRate and static method getTaxRate() to **Product** class.
2. Add static members to **Student** class so that course fee is not stored for each student or hard-coded.
3. Add functionality to get number of objects created in **Counter** class.
4. Create a record for **Circle** with x, y and radius and provide area() method to return area. Create objects and use toString() and getter methods.

Inheritance and Runtime Polymorphism

1. Create the classes to store details of Doctors - Resident doctor, Consultant. Resident doctors have salary and Consultants have visits and rate per visit.
   Provide method like print(), getPay() apart from constructor and other getter and setter methods.
2. Create classes to store details of different types of Courses – Online and Classroom. For all courses, we store title, fee and duration. For online course, we store URL to join online session. For classroom session, we store the address where sessions are conducted. For classroom sessions, we give 10% discount on course fee. Provide methods like print(), getFee() etc.

Arrays, String, StringJoiner, Math and Object classes

1. Accept a string and display it in reverse order.
2. Accept a string and display each word on separate line.
3. Take a String and invert the case of characters.
4. Accept 10 strings and display them in sorted order.
5. Accept strings until user enters END and display them by separating them with hyphen(-).
6. Accept a string, substring and display all indexes where substring is found in string.
7. Design guessing number game.
8. Implement toString(), equals() and hashCode() methods for all classes created so far.

Date classes, Wrapper classes, Boxing

1. Accept two dates and display number of days between those two dates. Dates must be accepted in dd-mm-yyyy format.
2. Understand the difference between Type casting, Upcasting and Downcasting, Boxing and unboxing.

Interfaces, Nested Classes, Packages

1. Create an interface called Shape with methods area() and circumference(). Implement it in classes like Circle and Square.
2. Create an interface called account operation with methods deposit, getBalance. Implement this interface in account class.
3. Create a package called **sales** with classes - Order, Customer and OrderItem. Access these classes from Test class in **testing** package.
4. Create Person class with name and age. Make Person objects sortable by age.
5. Design classes that need to sort collection of Person objects either by name or by age.

Exception Handling

1. Accept 10 numbers and display average. Make sure invalid numbers are silently ignored.
2. Create custom exceptions **InsufficientQuantityException** and **InvalidQuantityException**. Throw these exceptions from methods like sale() and purchase() of **Product** class.
3. Create InsufficientBalanceException with message "Insufficient Balance X for withdraw of Y".

IO Streams, Collections

1. Accept a filename from user and display how many uppercase, lowercase and digits the file has.
2. Remove all blank lines from a file. Make sure you display the original file eventually.
3. Take data from marks.txt file, which contains marks for each student in a line, separated by comma (,). Display rollno (line no) and total marks for each student. Make your program robust.
4. Display lines that contain more than 5 characters from the given file along with line numbers.
5. Accept names from user until END is given and write them in sorted order into names.txt file.
6. Create a class to store details of Circle. Create a collection of Circle objects and sort them by radius of the circle. Display all circle objects after sorting.
7. Take emails from *emails.txt*, which contains emails separated by comma (,) in each line. Write all valid email addresses in the sorted order into *validemails.txt* file.
8. Write 20 marks (random values) into Marks.dat.
9. Accept rollno and display marks for the student with that rollno.
10. Accept rollno and display current marks and ask user to enter new marks and change marks in the file.
11. Create a TreeSet of strings and sort them by length (use Comparator).

Maps, Regular Expressions

1. Accept customer name and phone number and display all in the sorted order of customer name.
2. Write a program to count how many times each word occurs in the given file.
3. Write a program to replace one or more spaces with a single space in the given file.
4. Accept PAN and check whether it is valid.
5. Write a function that takes a password (string) and returns true if it contains at least one digit, one uppercase letter and one special character.
6. Accept a file and replace one or more spaces with a single space and write new contents back to file.

Lambda and Streams

1. Take lines from a file and display top 5 lines with length more than 10.
2. Sort all unique names from names.txt and display them along with length of each name.
3. Display all the numbers that are greater than average of all numbers. Assume each line contains a single number in a file.
4. List all .txt files in a given folder recursively. Use Files.walk().
5. Delete all .tmp files recursively from the given folder.