Introduction to Computer Programming (CSE1001)

Assignment-6

Methods & Strings

METHODS

Question No	Questions	Course Outcome

1.	Design a **Simple Calculator** using methods in Java that provides the following operations: Addition, Subtraction, Multiplication, Division, Remainder, and Square Root. The method signatures are given below:
	`public static int additionSimple(int x, int y)`
	• Takes two integers x and y. Returns x + y.
	`public static int subtractionSimple(int x, int y)`
	• Takes two integers x and y. Returns y – x.
	`public static int multiplicationSimple(int x, int y)`
	• Takes two integers x and y. Returns x * y.
	`public static double divisionSimple(int x, int y)`
	• Takes two integers x and y. Returns y / x.
	• Check if x is zero before dividing.
	`public static int remainderSimple(int n, int m)`
	• Takes two integers n and m. Returns n % m.
	`public static double squareRootSimple(int n)`
	• Takes one integer n. Returns the square root of n.
	• Ensure n is positive.
	The calculator must function like a real calculator. Therefore, the program should display a menu and run continuously. The user should be able to choose an operation repeatedly until selecting an option to exit the program.
	
	Sample run1:
	`----- SIMPLE CALCULATOR -----`
	`1. Addition`
	`2. Subtraction`
	`3. Multiplication`
	`4. Division`
	`5. Remainder`
	`6. Square Root`
	`7. Exit`
	`Choose an option: 1`
	
	`Enter first number (x): 5`
	`Enter second number (y): 2`
	`Result (x + y) = 7`
	
	`----- SIMPLE CALCULATOR -----`
	`1. Addition`
	`2. Subtraction`
	`3. Multiplication`
	`4. Division`
	`5. Remainder`
	`6. Square Root`
	`7. Exit`
		CO4

```
Choose an option: 2

Enter first number (x): 5
Enter second number (y): 2
Result (y - x) = -3

----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
Choose an option: 3

Enter first number (x): 5
Enter second number (y): 2
Result (x * y) = 10

----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
Choose an option: 4

Enter first number (x): 5
Enter second number (y): 2
Result (y / x) = 0.4

----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
Choose an option: 5

Enter first number (n): 17
Enter second number (m): 5
Result (n % m) = 2

----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
```

```
Choose an option: 6

Enter number for square root: 81
Result = 9.0

----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
Choose an option: 7
Exiting... Thank you!
```

Sample run2:

```
----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
Choose an option: 4

Enter first number (x): 0
Enter second number (y): 15
Error! Division by zero is not allowed.

----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
Choose an option: 6

Enter number for square root: -9
Error! Square root of a negative number is not allowed.

----- SIMPLE CALCULATOR -----
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Remainder
6. Square Root
7. Exit
Choose an option: 7

Exiting... Thank you!
```

2.	**An Armstrong number** (also known as a **Narcissistic number**) is a number that is equal to the sum of its digits, with each digit raised to the power of the total number of digits in the number.
	
	Example:
	• **153** is an Armstrong number because: $1^3 + 5^3 + 3^3 = 153$
	• **9474** is an Armstrong number because: $9^4 + 4^4 + 7^4 + 4^4 = 9474$
	
	Write a Java program that identifies and displays **all Armstrong numbers between 100 and 10,000**. Also display its count.
	Implement and use **the following three methods**:
	
	`public static int countDigits(int n)`
	returns number of digits
	`public static int power(int a, int b) –`
	returns a^b
	`public static boolean isArmstrong(int n) –`
	checks if a number is Armstrong
	
	Sample run:
	`Armstrong Numbers from 100 to 10000:`
	`153`
	`370`
	`371`
	`407`
	`1634`
	`8208`
	`9474`
	
	`Total Armstrong Numbers Found: 7`
		CO4

3.	A **pentagonal number** is defined as n (3n–1)/2 for n = 1, 2, . . ., and so on. Therefore, the first few numbers are 1, 5, 12, 22, ... Write a method with the following header that returns a pentagonal number: `public static int getPentagonalNumber(int n).` Write a **Java Program** that uses this method to display the first 100 pentagonal numbers with 10 numbers on each line. **Sample run:** 1 5 12 22 35 51 70 92 117 145 176 210 247 287 330 376 425 477 532 590 651 715 782 852 925 1001 1080 1162 1247 1335 1426 1520 1617 1717 1820 1926 2035 2147 2262 2380 2501 2625 2752 2882 3015 3151 3290 3432 3577 3725 3876 4030 4187 4347 4510 4676 4845 5017 5192 5370 5551 5735 5922 6112 6305 6501 6700 6902 7107 7315 7526 7740 7957 8177 8400 8626 8855 9087 9322 9560 9801 10045 10292 10542 10795 11051 11310 11572 11837 12105 12376 12650 12927 13207 13490 13776 14065 14357 14652 14950	CO4

4.	A ***regular polygon*** is an n-sided polygon in which all sides are of equal length and all interior angles are equal. The formula for calculating the area of a regular polygon is:
	$$\text{Area} = \frac{n * s^2}{4 * \tan\left(\frac{\pi}{n}\right)}$$
	• n = number of sides
	• s = length of each side
	Write a **Java method** named area that returns the area of a regular polygon using the following method header:
	`public static double area(int n, double side)`
	The method should return the computed area using the formula above.
	
	Sample run1:
	`Enter number of sides: 5`
	`Enter length of each side: 6`
	`Area of the regular polygon: 61.93718642120281`
	
	Sample run2:
	`Enter number of sides: 8`
	`Enter length of each side: 10`
	`Area of the regular polygon: 482.842712474619`
		CO6

5.	The *Collatz sequence* is defined using the following rules for any positive integer **n**:
	• If **n is even**, replace n with **n / 2**
	• If **n is odd**, replace n with **3n + 1**
	• Continue repeating this process until the value of n becomes **1**
	The number of steps required to reach **1** is called the **Collatz length**.
	Write a Java method with the following header:
	`public static int collatzLength(int n)`
	The method should:
	• Accept a **positive integer n**
	• Apply the Collatz rules repeatedly
	• Count how many steps it takes for n to reach **1**
	• Return this count
	
	Sample run1:
	`Enter a number: 6`
	`Collatz length: 8`
	
	Sample run2:
	`Enter a number: 11`
	`Collatz length: 14`
		CO4

6.	Two numbers m and n are said to be a **friendly pair** if the ratio of the sum of their proper divisors to the number itself is equal. `sumOfDivisors(m) / m = sumOfDivisors(n) / n` Example: (6, 28) → friendly pair (30, 140) → friendly pair (248, 620) → friendly pair (12, 36) → not a friendly pair (18, 20) → not a friendly pair (10, 16) → not a friendly pair	CO4

	Write a method with the header
	`public static int sumOfDivisors(int n)`
	It returns the sum of all proper divisors of the number n (excluding the number itself).
	Then write another method with the header
	`public static Boolean isFriendlyPair(int a, int b)`
	It checks whether the two numbers a and b satisfy the friendly pair condition.
	
	Write a Java program that reads two integers from the user and checks whether they form a friendly pair or not.
	
	Sample run:
	`Enter first number: 30`
	`Enter second number: 140`
	
	`Sum of proper divisors of 30 = 42`
	`Sum of proper divisors of 140 = 196`
	
	`30 and 140 are a Friendly Pair.`	

7.	Write a method with the following header:
	`public static int numberOfDaysInAYear(int year)`
	
	This method should return the number of days in the given year.
	A year has **366 days** if it is a **leap year**, and **365 days** otherwise.
	
	Write a **Java Program** that displays the number of days in year from 2000 to 2020.
	
	Sample run:
	`Year Days`
	`2000 366`
	`2001 365`
	`2002 365`
	`2003 365`
	`2004 366`
	`2005 365`
	`2006 365`
	`2007 365`
	`2008 366`
	`2009 365`
	`2010 365`
	`2011 365`
	`2012 366`
	`2013 365`
	`2014 365`
	`2015 365`
	`2016 366`
	`2017 365`
	`2018 365`
	`2019 365`
	`2020 366`
		CO4

8.	Write a method that returns the **largest digit** in a given integer. Use the following method header: `public static int largestDigit(int n)` The method should: • Examine each digit of the integer **n**	CO4

	• Use a **loop** to compare digits
	• Return the **largest digit** in the number
	
	For example:
	• largestDigit(4829) → 9
	• largestDigit(7031) → 7
	• largestDigit(5) → 5
	
	Write a **Java Program** that prompts the user to enter an integer and then displays the largest digit in the number.
	
	Sample run 1:
	`Enter an integer: 4829`
	`Largest digit: 9`
	
	Sample run 2:
	`Enter an integer: 7031`
	`Largest digit: 7`	

9.	Write the methods with the following headers
	`public static int reverseint number)`
	It returns the reversal of an integer.
	Example:
	reverse (456), returns 654
	
	`public static boolean isPalindrome(int number)`
	It returns true if number is a palindrome
	
	Use the reverse method to implement isPalindrome( ).
	
	A number is a palindrome if its reversal is the same as itself.
	Write a **Java Program** that prompts the user to enter an integer and reports whether the integer is a palindrome.
	
	Sample run1:
	`Enter an integer: 121`
	`121 is a palindrome.`
	
	Sample run2:
	`Enter an integer: 456`
	`456 is not a palindrome.`
		CO5

10.	Write a java program to calculate the area of triangle, square, circle, rectangle by using method overloading.
	Sample run1:
	`=== AREA CALCULATOR (Method Overloading) ===`
	`1. Area of Square`
	`2. Area of Rectangle`
	`3. Area of Circle`
	`4. Area of Triangle`
	`5. Exit`
	`Enter your choice: 1`
	`Enter side: 5`
	`Area of Square: 25`
	
	Sample run2:
		CO5

	`=== AREA CALCULATOR (Method Overloading) ===`
	`1. Area of Square`
	`2. Area of Rectangle`
	`3. Area of Circle`
	`4. Area of Triangle`
	`5. Exit`
	`Enter your choice: 2`
	`Enter length: 10`
	`Enter width: 4`
	`Area of Rectangle: 40.0`	

	STRINGS	
11	Write a method that finds the number of occurrences of a specified character in a string using the following header: `public static int count(String str, char a)` **For example,** count ("Welcome", 'e') returns 2. Write a **Java Program** that prompts the user to enter a string followed by a character and displays the number of occurrences of the character in the string. **Sample run1:** Enter a string: Welcome Enter a character: e The number of occurrences of 'e' in "Welcome" is 2 **Sample run2:** Enter a string: Mississippi Enter a character: s The number of occurrences of 's' in "Mississippi" is 4	CO4

12	Write a method that finds the number of vowels in a given string using the following method header:
	
	`public static int countVowels(String str)`
	
	A vowel is any of the following characters: a, e, i, o, u (both uppercase and lowercase).
	For example:
	`countVowels("Welcome") returns 3.`
	
	Sample run1:
	`Enter a string: Welcome`
	`The number of vowels in "Welcome" is 3`
	
	Sample run2:
	`Enter a string: Programming in Java`
	`The number of vowels in "Programming in Java" is 7`
		CO4

13	Write a method that counts the number of words in a given string using the following method header: `public static int countWords(String str)` A *word* is defined as a sequence of characters separated by spaces. **For example:**	CO4

	`countWords("Java is fun") returns 3.`
	
	Write a **Java Program** that prompts the user to enter a sentence and then displays the total number of words in the sentence. Provide a sample run of the program.
	
	Sample run1:
	`Enter a sentence: Java is fun`
	`The number of words in the sentence is 3`
	
	Sample run2:
	`Enter a sentence: Welcome to the world of programming`
	`The number of words in the sentence is 6`	

14	Write a **Java Program** that converts a given string so that the first letter of every word becomes uppercase, effectively converting the string to **title case.**
	
	The program should include a method with the following header:
	`public static String toTitleCase(String str)`
	
	The method should take a string as input and return the converted string where the first letter of each word is capitalized.
	
	Example:
	• Input: "welcome to java"
	• Output: "Welcome To Java"
	
	The program should prompt the user to enter a string and then display the converted title-case string.
	
	Sample run1:
	`Enter a string: welcome to java`
	`Converted string: Welcome To Java`
	
	Sample run2:
	`Enter a string: programming in python is fun`
	`Converted string: Programming In Python Is Fun`
		CO4

15	Write a **Java Program** that checks whether a given string is a palindrome. The program should include a method with the following header: `public static boolean isPalindrome(String str)` • The method should take a string as input and return true if the string is a palindrome, and false otherwise. • A *palindrome* is a string that reads the same backward as forward (e.g., "madam" or "level"). The program should prompt the user to enter a string and then display whether the string is a palindrome or not. **Example:** • Input: "madam" → Output: "madam is a palindrome" • Input: "java" → Output: "java is not a palindrome"	CO5

	Sample run1:
	`Enter a string: madam`
	`madam is a palindrome`
	
	Sample run2:
	`Enter a string: level`
	`level is a palindrome`	

16	Some websites impose specific rules for passwords to ensure security. Write a **Java program** that checks whether a given string is a valid password based on the following rules: **Password Rules** 1. A password must have **at least eight characters**. 2. A password must consist of **only letters and digits**. 3. A password must contain **at least two digits**. The program should include a method with the following header: `public static boolean isValidPassword(String password)` • The method should take a string as input and return true if the password satisfies all the rules, and false otherwise. **Program Requirement** • The program should prompt the user to enter a password. • It should then display: o "Valid Password" if the password meets all the rules, or o "Invalid Password" if any rule is violated. **Example** • Input: "Pass1234" → Output: Valid Password • Input: "password" → Output: Invalid Password • Input: "Pass12!" → Output: Invalid Password	CO6
17	Write a **Java method** to display the middle character of a string. Note: a) If the length of the string is odd there will be one middle characters. b) If the length of the string is even, then there would be two middle characters, we need to print the second middle character. **Sample run:** Input a string: ABC The middle character in the string: B Input a string: JAVA The middle character in the string: V	CO4
	HOME ASSIGNMENT	
1.	Write a java program to calculate the volume of 3D shapes like cube, cuboid, sphere, cylinder by using method overloading. **Formula and method header:** Volume of cube V = $side^3$ `public static double volume(int side)` Volume of cuboid V = length × breadth × height	**CO5**

	`public static double volume(double length, double width, double height)`
	
	Volume of sphere $V = \frac{4}{3} \times \pi \times r^3$
	`public static double volume(double radius)`
	
	Volume of Cylinder $V = \pi \times r^2 \times h$
	`public static double volume(double radius, double height)`
	
	Sample run1:
	`=== VOLUME CALCULATOR (Method Overloading) ===`
	`1. Volume of Cube`
	`2. Volume of Rectangular Prism (Cuboid)`
	`3. Volume of Sphere`
	`4. Volume of Cylinder`
	`5. Exit`
	`Enter your choice: 2`
	
	`Enter length: 10`
	`Enter width: 4`
	`Enter height: 6`
	`Volume of Rectangular Prism: 240.0`
	
	`Sample run2:`
	`=== VOLUME CALCULATOR (Method Overloading) ===`
	`1. Volume of Cube`
	`2. Volume of Rectangular Prism (Cuboid)`
	`3. Volume of Sphere`
	`4. Volume of Cylinder`
	`5. Exit`
	`Enter your choice: 4`
	
	`Enter radius: 4`
	`Enter height: 10`
	`Volume of Cylinder: 502.6548245743669`	

2.	Write a **Java Method** that accept three integers and check whether they are consecutive are not. Return true or false. **Sample run:** Input the first number: 15 Input the second number: 16 Input the third number: 17 Check whether the three said numbers are consecutive or not! True	**CO4**
3.	Write a method that counts the number of spaces in a given string using the following method header: `public static int countSpaces(String str)` The method should return the total number of space characters found in the string. Write a Java program that prompts the user to enter a string and then displays the number of spaces in the string. **Sample run1:** Enter a string: Welcome to Java programming	**CO3**

	Number of spaces in the string: 3 **Sample run2:** Enter a string: NoSpacesHere Number of spaces in the string: 0	
4.	Write a **Java Program** that reads a given string and counts the number of uppercase letters (A–Z), lowercase letters (a–z), digits (0–9), and special characters (any character that is not a letter or digit). Implement a method with the following header `public static void countCharacters(String str)` It takes a string as input and prints these counts. **Sample run:** Enter a string: Hello World! 123 Uppercase letters: 2 Lowercase letters: 8 Digits: 3 Special characters: 3	**CO3**
5.	Understand and use the following **basic String methods** such as: length(), charAt(), toUpperCase(), toLowerCase(), contains(), startsWith(), endsWith(), trim(), replace(), substring() for the following questions: • Display the length of the string. • Display the first and last character of the string. • Convert the string to uppercase and display it. • Convert the string to lowercase and display it. • Check if the string contains the word "Java" and display the result. • Check if the string starts with "Hello" and display the result. • Check if the string ends with "World" and display the result. • Trim leading and trailing spaces and display the trimmed string. • Replace all occurrences of 'a' with '@' and display the result. • Extract a substring starting from the 3rd character to the 7th character (inclusive) and display it. **Sample run:** Enter a string: Hello Java World Length of the string: 15 First character: 'H' Last character: 'd' String in uppercase: "HELLO JAVA WORLD" String in lowercase: "hello java world" Does the string contain "Java"? true Does the string start with "Hello"? true Does the string end with "World"? true Trimmed string: "Hello Java World" String after replacing 'a' with '@': "Hello J@v@ World" Substring from index 2 to 7: "llo J"	**CO2**

****