**1.**  **Question: Password Strength Checker Application**

**Objective:** Design and implement a password-strength checker application that evaluates the strength of a given password based on two criteria: length and complexity.

**Requirements:**

1. **Interface Definition:**
   * Define an interface named PasswordChecker with the following methods:
     + checkLength(String password): This method should determine the length of the given password and display the length.
     + checkComplexity(String password): This method should analyze the complexity of the password and return a string indicating its strength:
       - If the length is less than 8 characters, the password is considered **"Weak."**
       - If the length is exactly 8 characters, it is labeled as **"Medium."**
       - If the length is greater than 8 characters, it is marked as **"Strong."**
2. **Class Implementation:**
   * Implement a class named SimplePasswordChecker that implements the PasswordChecker interface.
   * Provide the necessary logic for both the checkLength and checkComplexity methods.
3. **Note:**
   * The main class has been provided in the editor.
   * Refer to the displayed test cases for the input and output format.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

Enter a password:

password@123

* + **Expected Output:**

Length: 12

Strong

1. **Test Case 2:**
   * **Input:**

Enter a password:

password

* + **Expected Output:**

Length: 8

Medium

1. **Test Case 3:**
   * **Input:**

Enter a password:

s@123

* + **Expected Output:**

Length: 5

Weak

1. **Test Case 4:**
   * **Input:**

Enter a password:

def2323@12

* + **Expected Output:**

Length: 10

Strong

1. **Test Case 5:**
   * **Input:**

Enter a password:

defrhj89

* + **Expected Output:**

Length: 8

Medium

1. **Test Case 6:**
   * **Input:**

Enter a password:

w213

* + **Expected Output:**

Length: 4

Weak

**CheckPassword.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**interface** PasswordChecker {

// write your code here..

String checkLength(String password);

String checkComplexity(String password);

}

**class** SimplePasswordChecker **implements** PasswordChecker {

@Override

**public** String checkLength(String password) {

**return** "Length: " + password.length();

}

@Override

**public** String checkComplexity(String password) {

**if** (password.length() < 8) {

**return** "Weak";

} **else** **if** (password.length() == 8) {

**return** "Medium";

} **else** {

**return** "Strong";

}

}

}

**public** **class** CheckPassword {

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

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

System.***out***.print("Enter a password: ");

String password = scanner.nextLine();

// Create an instance of the SimplePasswordChecker class

PasswordChecker simpleChecker = **new** SimplePasswordChecker();

// Perform length and complexity checks

System.***out***.println(simpleChecker.checkLength(password));

System.***out***.println(simpleChecker.checkComplexity(password));

scanner.close();

}

}

**2.** **Question: Anagram Checker Program**

**Objective:** Develop a Java program that helps students understand the concept of anagrams and practice identifying them.

**Definition:** An anagram is a word or phrase formed by rearranging the letters of another word or phrase. For example, "listen" and "silent" are anagrams because they use the same letters, just in a different order.

**Requirements:**

1. **Input:**
   * The program should prompt the user to enter two strings.
   * The strings can contain any characters, including alphabets, numbers, and special characters.
   * The input strings are case-sensitive.
2. **Output:**
   * The program should output a message indicating whether the two input strings are anagrams of each other or not.
3. **Implementation:**
   * Utilize a HashMap to determine if the entered strings are anagrams.
4. **Note:**
   * The main method has been provided in the editor.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

silent

listen

* + **Expected Output:**

silent and listen are anagrams

1. **Test Case 2:**
   * **Input:**

question

quest

* + **Expected Output:**

question and quest are not anagrams

1. **Test Case 3:**
   * **Input:**

racecar

carrace

* + **Expected Output:**

racecar and carrace are anagrams

1. **Test Case 4:**
   * **Input:**

wellbeing

* + **Expected Output:**

wellness and wellbeing are not anagrams

1. **Test Case 5:**
   * **Input:**

Amb123@

123bma@

* + **Expected Output:**

Amb123@ and 123bma@ are not anagrams

1. **Test Case 6:**
   * **Input:**

AND123@

123AND@

* + **Expected Output:**

AND123@ and 123AND@ are anagrams

**/AnagramChecker.java**

**//==================================**

**package** q22623;

**import** java.util.HashMap;

**import** java.util.Scanner;

**public** **class** AnagramChecker {

// write the code..

**public** **static** **boolean** areAnagrams(String str1, String str2) {

**if** (str1.length() != str2.length()) {

**return** **false**;

}

HashMap<Character, Integer> charCount = **new** HashMap<>();

**for** (**char** c : str1.toCharArray()) {

charCount.put(c, charCount.getOrDefault(c, 0) + 1);

}

**for** (**char** c : str2.toCharArray()) {

charCount.put(c, charCount.getOrDefault(c, 0) - 1);

**if** (charCount.get(c) < 0) {

**return** **false**;

}

}

**return** **true**;

}

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

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

String str1 = scanner.nextLine();

String str2 = scanner.nextLine();

**if** (*areAnagrams*(str1, str2)) {

System.***out***.println(str1 + " and " + str2 + " are anagrams");

} **else** {

System.***out***.println(str1 + " and " + str2 + " are not anagrams");

}

scanner.close();

}

}

**3.** **Question: Voting System for an Election**

**Objective:** Develop a voting system that calculates voting results and determines the winner based on the votes cast by voters.

**Requirements:**

1. **Interface Definition:**
   * Define an interface named VoteCounter with the following methods:
     1. **countVotes(String[] candidates, int[] votes):**
        + This method should calculate the total number of votes received by each candidate based on the provided arrays of candidates and votes.
        + It should return an array containing the vote counts for each candidate in the same order as the candidates array.
     2. **determineWinner(int[] voteCounts):**
        + This method should determine the winner of the election based on the provided vote counts.
        + It should return an integer indicating the index of the winning candidate in the voteCounts array.
2. **Class Implementation:**
   * Implement a class named ElectionSystem that implements the VoteCounter interface.
   * Provide the necessary logic for both the countVotes() and determineWinner() methods.
3. **Note:**
   * The main class with the driver code has been provided in the editor.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

no of candidates:

2

name of candidate 1:

Candy1

name of candidate 2:

Candy2

votes for candidate Candy1:

256

votes for candidate Candy2:

275

* + **Expected Output:**

Vote Counts:

Candy1: 256

Candy2: 275

Winner: Candy2

1. **Test Case 2:**
   * **Input:**

no of candidates:

5

name of candidate 1:

C1

name of candidate 2:

C2

name of candidate 3:

C3

name of candidate 4:

C4

name of candidate 5:

C5

votes for candidate C1:

248

votes for candidate C2:

268

votes for candidate C3:

258

votes for candidate C4:

289

votes for candidate C5:

294

* + **Expected Output:**

Vote Counts:

C1: 248

C2: 268

C3: 258

C4: 289

C5: 294

Winner: C5

1. **Test Case 3:**
   * **Input:**

no of candidates:

2

name of candidate 1:

C1

name of candidate 2:

C2

votes for candidate C1:

499

votes for candidate C2:

499

* + **Expected Output:**

Vote Counts:

C1: 499

C2: 499

It's a tie!

1. **Test Case 4:**
   * **Input:**

no of candidates:

5

name of candidate 1:

as

name of candidate 2:

sd

name of candidate 3:

df

name of candidate 4:

fg

name of candidate 5:

gh

votes for candidate as:

125

votes for candidate sd:

225

votes for candidate df:

145

votes for candidate fg:

228

votes for candidate gh:

300

* + **Expected Output:**

Vote Counts:

as: 125

sd: 225

df: 145

fg: 228

gh: 300

Winner: gh

**Main.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**interface** VoteCounter { // fill the missing code...

// write your code here...

**int**[] countVotes(String[] candidates, **int**[] votes);

**int** determineWinner(**int**[] voteCounts);

}

**class** ElectionSystem { // fill the missing code...

// write your code here...

**public** **int**[] countVotes(String[] candidates, **int**[] votes) {

**return** votes;

}

**public** **int** determineWinner(**int**[] voteCounts) {

**int** maxVotes = -1;

**int** winnerIndex = -1;

**boolean** isTie = **false**;

**for** (**int** i = 0; i < voteCounts.length; i++) {

**if** (voteCounts[i] > maxVotes) {

maxVotes = voteCounts[i];

winnerIndex = i;

isTie = **false**;

} **else** **if** (voteCounts[i] == maxVotes) {

isTie = **true**;

}

}

**if** (isTie) {

**return** -1;

}

**return** winnerIndex;

}

}

**public** **class** Main {

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

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

ElectionSystem electionSystem = **new** ElectionSystem();

System.***out***.print("no of candidates: ");

**int** numCandidates = scanner.nextInt();

scanner.nextLine();

String[] candidates = **new** String[numCandidates];

**int**[] votes = **new** **int**[numCandidates];

**for** (**int** i = 0; i < numCandidates; i++) {

System.***out***.print("name of candidate " + (i + 1) + ": ");

candidates[i] = scanner.nextLine();

}

**for** (**int** i = 0; i < numCandidates; i++) {

System.***out***.print("votes for candidate " + candidates[i] + ": ");

votes[i] = scanner.nextInt();

}

// Count votes

**int**[] voteCounts = electionSystem.countVotes(candidates, votes);

System.***out***.println("Vote Counts:");

**for** (**int** i = 0; i < candidates.length; i++) {

System.***out***.println(candidates[i] + ": " + voteCounts[i]);

}

// Determine the winner

**int** winningCandidateIndex = electionSystem.determineWinner(voteCounts);

**if** (winningCandidateIndex == -1) {

System.***out***.println("It's a tie!");

} **else** {

System.***out***.println("Winner: " + candidates[winningCandidateIndex]);

}

scanner.close();

}

}

**Question: String Parsing for Numeric Values**

**Objective:** Develop a Java program that parses strings into numeric values and handles invalid formats with appropriate error messages.

**Input Format:**

* The input consists of strings that need to be parsed into numeric values.

**Output Format:**

* If the input string can be parsed successfully, print the number in the format:

Parsed number: <parsed\_value>

* If the input string cannot be parsed, display the message:

Error: Invalid number format

**Sample Test Cases**

**Difficulty Level:** Very Easy

**Test Case 1:**

* **Input:**

25.6

* **Expected Output:**

Parsed number: 25.6

**Test Case 2:**

* **Input:**

12-3

* **Expected Output:**

Error: Invalid number format

**Test Case 3:**

* **Input:**

28

* **Expected Output:**

Parsed number: 28.0

**Test Case 4:**

* **Input:**

s3

* **Expected Output:**

Error: Invalid number format

**StringParser.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**public** **class** StringParser {

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

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

String input = sc.nextLine();

**try** {

**double** number = Double.*parseDouble*(input);

System.***out***.println("Parsed number: " + number);

}

**catch** (NumberFormatException e) {

System.***out***.println("Error: Invalid number format");

} **finally** {

sc.close();

}

}

}

**17.** **Question: Remove Duplicates from a List of Integers**

**Objective:** Rahul is working on a program to remove duplicate elements from a list of integers. Write a Java program using ArrayList to assist him in implementing this functionality.

**Input Format:**

* The user will input a list of integers, one by one.
* The user will enter any non-numeric character to indicate the end of the input.

**Output Format:**

* The first line should display the original list of integers.
* The second line should display the list with duplicate elements removed.

**Sample Test Cases**

**Difficulty Level:** Very Easy

**Test Case 1:**

* **Input:**

1

2

3

2

2

1

3

4

1

2

s

* **Expected Output:**

1 2 3 2 2 1 3 4 1 2

1 2 3 4

**Test Case 2:**

* **Input:**

-1

-9

25

12

14

62

23

\*

* **Expected Output:**

-1 -9 25 12 14 62 23

-1 -9 25 12 14 62 23

**Test Case 3:**

* **Input:**

11

13

12

13

12

11

11

12

p

* **Expected Output:**

11 13 12 13 12 11 11 12

11 13 12

**Test Case 4:**

* **Input:**

10

-5

14

25

-6

@

* **Expected Output:**

10 -5 14 25 -6

10 -5 14 25 -6

**DuplicateRemover.java**

**//==================================**

**package** q22623;

**import** java.util.ArrayList;

**import** java.util.Scanner;

**public** **class** DuplicateRemover {

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

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

ArrayList<Integer> numbers = **new** ArrayList<>();

**while** (scanner.hasNextInt()) {

numbers.add(scanner.nextInt());

}

**for** (**int** number : numbers) {

System.***out***.print(number + " ");

}

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

ArrayList<Integer> uniqueNumbers = **new** ArrayList<>();

**for** (**int** number : numbers) {

**if** (!uniqueNumbers.contains(number)) {

uniqueNumbers.add(number);

}

}

**for** (**int** number : uniqueNumbers) {

System.***out***.print(number + " ");

}

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

scanner.close();

}

}

**18.** **Question: Sort a List of Strings in Lexicographical Order**

**Objective:** Develop a Java program that assists Priya in sorting a list of strings in lexicographical (dictionary) order using ArrayList.

**Input Format:**

1. The user will input a list of strings separated by spaces.
2. To indicate the end of input, the user will enter any non-alphabetic character.

**Output Format:**

The program should display the sorted list of strings in lexicographical order.

**Sample Test Cases**

**Difficulty Level: Very Easy**

**Test Case 1**

**Input:**

one two three four five six seven @

**Expected Output:**

five four one seven six three two

**Test Case 2**

**Input:**

banana grapes apples orange avacado plum strawberry 23

**Expected Output:**

apples avacado banana grapes orange plum strawberry

**Test Case 3**

**Input:**

ab aa ac abc ace ade afr 6

**Expected Output:**

aa ab abc ac ace ade afr

**Test Case 4**

**Input:**

code java easy learn with CodeTantra !

**Expected Output:**

CodeTantra code easy java learn with

**StringSorter.java**

**//==================================**

**package** q22623;

**import** java.util.ArrayList;

**import** java.util.Collections;

**import** java.util.Scanner;

**public** **class** StringSorter {

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

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

// Input the list of strings

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

**while** (scanner.hasNext()) {

String word = scanner.next();

**if** (!word.matches("[a-zA-Z]+")) {

**break**;

}

strings.add(word);

}

Collections.*sort*(strings);

**for** (String string : strings) {

System.***out***.print(string + " ");

}

// write the code..

scanner.close();

}

}

Diffculty Level Medium

===========================

**1.** **Question: Simple Calculator Application**

**Objective:** You are tasked with creating a simple calculator application that performs addition and multiplication operations.

**Class Definitions:**

1. **Base Class: Calculation**
   * **Constructor:** Takes two integer parameters.
   * **Method:**
     + addition: This method takes the two parameters and returns their sum.
2. **Child Class: My\_Calculation**
   * Inherits from the Calculation class.
   * **Method:**
     + multiplication: This method returns the product of the two parameters inherited from the base class.

**Input Format:**

* The first line of input contains an integer.
* The second line of input contains another integer.

**Output Format:**

* The first line should display the result of the addition operation.
* The second line should display the result of the multiplication operation.

**Notes:**

* The code for handling inputs, creating objects, invoking methods, and printing results is already provided in the editor. Your task is to implement the Calculation class based on the given specifications.

**Example Test Cases:**

1. **Test Case 1:**
   * **Input:**

1

2

* + **Expected Output:**

3

2

1. **Test Case 2:**
   * **Input:**

-10

-9

* + **Expected Output:**

-19

90

1. **Test Case 3:**
   * **Input:**

-10

8

* + **Expected Output:**

-2

-80

**MainCalculation.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**class** Calculation {

**protected** **int** num1, num2;

**public** Calculation(**int** num1, **int** num2) {

**this**.num1 = num1;

**this**.num2 = num2;

}

**public** **int** addition() {

**return** num1 + num2;

}

}

**class** My\_Calculation **extends** Calculation {

**public** My\_Calculation(**int** num1, **int** num2) {

**super**(num1, num2);

}

**public** **int** multiplication() {

**return** num1 \* num2;

}

}

**public** **class** MainCalculation {

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

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

**int** num1 = scanner.nextInt();

**int** num2 = scanner.nextInt();

My\_Calculation myCalculation = **new** My\_Calculation(num1, num2);

**int** sum = myCalculation.addition();

**int** product = myCalculation.multiplication();

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

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

scanner.close();

}

}

**2.** **Question: Shape Calculator Application**

• Design a class hierarchy with a base class Shape and two derived classes Cube and Cuboid.

• The Shape class should have a method calculateVolume() that prints "Calculating volume of Shape."

• The Cube class should override the calculateVolume() method to calculate and display the volume of a cube by taking any side (double) of the cube as input from the user.

• The Cuboid class should also override the calculateVolume() method to calculate and display the volume of a cuboid by taking the length (double), width (double), and height (double) of the cuboid as input from the user. Formulas: Volume of cube = (side) Volume of the cuboid = length \* width \* height

Note: • Print the volume up to 2 decimal places.

• The main class has been provided to you in the editor

. Sample Test Cases

Test Case 1

Expected Output

Side of the cube: 9.2

Volume of Cube: 778.69

Length of the cuboid: 1.2

Width of the cuboid: 5.0

Height of the cuboid: 2.0

Volume of Cuboid: 12.00

Test Case 2

Expected Output

Side of the cube: 14.0

Volume of Cube: 2744.00

Length of the cuboid: 2.0

Width of the cuboid: 3.0

Height of the cuboid: 1.0

Volume of Cuboid: 6.00

Test Case 3

Expected Output

Side of the cube: 2.5

Volume of Cube: 15.63

Length of the cuboid: 6.2

Width of the cuboid: 6.1

Height of the cuboid: 6.3

Volume of Cuboid: 238.27

Test Case 4

Expected Output

Side of the cube: 14.2

Volume of Cube: 2863.29

Length of the cuboid: 5.2

Width of the cuboid: 6.2

Height of the cuboid: 3.2

Volume of Cuboid:103.17

**ShapeCalculator.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**class** Cube {

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

**public** **void** calculateVolume()

{

System.***out***.print("Side of the cube: ");

**double** side=sc.nextFloat();

**double** sum=side\*side\*side;

System.***out***.printf("Volume of Cube: %.2f\n",+sum);

}

}

**class** Cuboid **extends** Cube {

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

**public** **void** calculateVolume() {

System.***out***.print("Length of the cuboid: ");

**double** length = sc.nextDouble();

System.***out***.print("Width of the cuboid: ");

**double** width = sc.nextDouble();

System.***out***.print("Height of the cuboid: ");

**double** heigth = sc.nextDouble();

**double** sum = length \* width \* heigth;

System.***out***.printf("Volume of Cuboid: %.2f\n", +sum);

}

}

**public** **class** ShapeCalculator {

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

// Creating instances of the subclasses

Cube cubeInstance = **new** Cube();

Cuboid cuboidInstance = **new** Cuboid();

// Calling calculateVolume() for each instanc

cubeInstance.calculateVolume();

cuboidInstance.calculateVolume();

}

}

**3.** **Question: Bookstore Purchase Calculator**

**Objective:** You are developing a program for a bookstore to calculate the total cost of a book purchase, including applicable discounts.

**Task:**

Write a Java program that takes the number of books purchased as input and calculates the total cost according to the following criteria:

* If the number of books is **1-5**: No discount (each book costs **$10**).
* If the number of books is **6-10**: **10% discount**.
* If the number of books is **11-20**: **15% discount**.
* If the number of books is **21 or more**: **20% discount**.

**Input Format:**

* The program should take a single input, which is the number of books purchased (an integer).

**Output Format:**

* The program should output the total cost based on the criteria mentioned above.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

mathematica

Enter the number of books purchased:

2

* + **Expected Output:**

bash

Total Cost: $20.0

1. **Test Case 2:**
   * **Input:**

mathematica

Enter the number of books purchased:

6

* + **Expected Output:**

bash

Total Cost: $54.0

1. **Test Case 3:**
   * **Input:**

mathematica

Enter the number of books purchased:

12

* + **Expected Output:**

bash

Total Cost: $102.0

1. **Test Case 4:**
   * **Input:**

mathematica

Enter the number of books purchased:

22

* + **Expected Output:**

mathematica

Total Cost: ...

**BookDiscountCalculator.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**public** **class** BookDiscountCalculator {

// Method to calculate the total cost based on the number of books

**public** **static** **double** calculateTotalCost(**int** numberOfBooks) {

**double** discount = (numberOfBooks >= 21) ? 0.2 : (numberOfBooks >= 11) ? 0.15 : (numberOfBooks >= 6) ? 0.1 : 0.0;

**double** totalCost = numberOfBooks \* 10 \* (1 - discount);

**return** totalCost;

}

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

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

System.***out***.print("Enter the number of books purchased: ");

**int** numberOfBooks = scanner.nextInt();

**double** totalCost = BookDiscountCalculator.*calculateTotalCost*(numberOfBooks);

System.***out***.println("Total Cost: $" + totalCost);

scanner.close();

}

}

**4.** **Question: Armstrong Number Checker**

**Objective:** Write a Java program that determines if a given integer NNN is an Armstrong number.

**Definition:**

An **Armstrong number** (also known as a narcissistic number) is a number that is equal to the sum of its own digits each raised to the power of the number of digits in that number. For example, the number 153 is an Armstrong number because:

13+53+33=1531^3 + 5^3 + 3^3 = 15313+53+33=153

**Input Format:**

* The program should take a single input, which is a positive integer.

**Output Format:**

* The program should output whether the given integer is an Armstrong number, as shown in the displayed test cases.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

153

* + **Expected Output:**

153 is an Armstrong number

1. **Test Case 2:**
   * **Input:**

1634

* + **Expected Output:**

1634 is an Armstrong number

1. **Test Case 3:**
   * **Input:**

154

* + **Expected Output:**

154 is not an Armstrong number

1. **Test Case 4:**
   * **Input:**

202

* + **Expected Output:**

202 is not an Armstrong number

**package** q22623;

**import** java.util.Scanner;

**public** **class** ArmstrongNumberChecker {

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

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

// Prompt the user to enter a positive integer

System.***out***.print("Enter a positive integer: ");

**int** number = scanner.nextInt();

// Check if the number is an Armstrong number

**if** (*isArmstrong*(number)) {

System.***out***.println(number + " is an Armstrong number");

} **else** {

System.***out***.println(number + " is not an Armstrong number");

}

scanner.close();

}

**public** **static** **boolean** isArmstrong(**int** number) {

**int** originalNumber = number;

**int** sum = 0;

**int** digits = String.*valueOf*(number).length(); // Get the number of digits

// Calculate the sum of each digit raised to the power of the number of digits

**while** (number > 0) {

**int** digit = number % 10; // Extract the last digit

sum += Math.*pow*(digit, digits); // Raise it to the power of digits and add to sum

number /= 10; // Remove the last digit

}

// Compare the sum with the original number

**return** sum == originalNumber;

}

}

**5.** **Question: Nutritional Calculator**

**Objective:** You are developing a nutritional calculator that helps users estimate the calorie content of a meal based on their choices.

**Task:**

Create a Java program that takes the user's meal preferences and calculates the approximate calorie count based on the following criteria:

* If the user chooses a **salad**, estimate **150 calories**.
* If the user chooses a **sandwich**, estimate **300 calories**.
* If the user chooses **pasta**, estimate **400 calories**.
* If the user adds **cheese** to their meal, add **100 calories**.
* If the user adds **meat** to their meal, add **150 calories**.

**Input Format:**

The program should take the following inputs:

1. The user's choice of meal (a string, either "salad," "sandwich," or "pasta").
2. Whether the user wants to add cheese (a boolean, true or false).
3. Whether the user wants to add meat (a boolean, true or false).

**Output Format:**

The program should output the approximate calorie count of the selected meal.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

salad

Do you want to add cheese? (true/false):

true

Do you want to add meat? (true/false):

false

* + **Expected Output:**

Approximate Calorie Count: 250 calories

1. **Test Case 2:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

salad

Do you want to add cheese? (true/false):

false

Do you want to add meat? (true/false):

true

* + **Expected Output:**

Approximate Calorie Count: 300 calories

1. **Test Case 3:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

sandwich

Do you want to add cheese? (true/false):

false

Do you want to add meat? (true/false):

false

* + **Expected Output:**

Approximate Calorie Count: 300 calories

1. **Test Case 4:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

sandwich

Do you want to add cheese? (true/false):

true

Do you want to add meat? (true/false):

true

* + **Expected Output:**

Approximate Calorie Count: 550 calories

1. **Test Case 5:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

sandwich

Do you want to add cheese? (true/false):

true

Do you want to add meat? (true/false):

false

* + **Expected Output:**

Approximate Calorie Count: 400 calories

1. **Test Case 6:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

sandwich

Do you want to add cheese? (true/false):

false

Do you want to add meat? (true/false):

true

* + **Expected Output:**

Approximate Calorie Count: 450 calories

1. **Test Case 7:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

pasta

Do you want to add cheese? (true/false):

true

Do you want to add meat? (true/false):

true

* + **Expected Output:**

Approximate Calorie Count: 650 calories

1. **Test Case 8:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

pasta

Do you want to add cheese? (true/false):

true

Do you want to add meat? (true/false):

false

* + **Expected Output:**

Approximate Calorie Count: 500 calories

1. **Test Case 9:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

pasta

Do you want to add cheese? (true/false):

false

Do you want to add meat? (true/false):

true

* + **Expected Output:**

Approximate Calorie Count: 550 calories

1. **Test Case 10:**
   * **Input:**

Choose your meal (salad/sandwich/pasta):

salad

Do you want to add cheese? (true/false):

true

Do you want to add meat? (true/false):

true

* + **Expected Output:**

Approximate Calorie Count: 400 calories

**NutritionalCalculator.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**public** **class** NutritionalCalculator {

// Method to calculate the approximate calorie count based on user'smeal preferences

**public** **static** **int** calculateCalories(String mealChoice, **boolean** addCheese, **boolean** addMeat) {

**int** caloriecount = 0;

**switch** (mealChoice.toLowerCase()) {

**case** "salad":

caloriecount = 150;

**break**;

**case** "sandwich":

caloriecount = 300;

**break**;

**case** "pasta":

caloriecount = 400;

**break**;

**default**:

System.***out***.println("Invalid meal Choice");

**return** 0;

}

**if** (addCheese) {

caloriecount += 100;

}

**if** (addMeat) {

caloriecount += 150;

}

**return** caloriecount;

}

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

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

System.***out***.print("Choose your meal (salad/sandwich/pasta): ");

String mealChoice = scanner.nextLine();

System.***out***.print("Do you want to add cheese? (true/false): ");

**boolean** addCheese = scanner.nextBoolean();

System.***out***.print("Do you want to add meat? (true/false): ");

**boolean** addMeat = scanner.nextBoolean();

**int** calorieCount = *calculateCalories*(mealChoice, addCheese, addMeat);

System.***out***.println("Approximate Calorie Count: " + calorieCount + "calories");

scanner.close();

}

}

**11.** **Question: Pyramid Pattern Generator**

**Objective:** Help Moulya, a passionate traveler, create a Java program that prints a pyramid pattern based on user input for the number of rows.

**Task:**

Write a Java program that captures the essence of the pyramids of Egypt by generating a pyramid pattern. The pattern should reflect numbers in a specific arrangement based on the number of rows provided by the user.

**Input Format:**

* The input is an integer that represents the number of rows for the pyramid.

**Output Format:**

* The output should be a pyramid pattern as demonstrated in the sample test cases below.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

5

* + **Expected Output:**

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1

1. **Test Case 2:**
   * **Input:**

7

* + **Expected Output:**

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1

1 2 3 4 5 6 5 4 3 2 1

1 2 3 4 5 6 7 6 5 4 3 2 1

1. **Test Case 3:**
   * **Input:**

10

* + **Expected Output:**

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1

1 2 3 4 5 4 3 2 1

1 2 3 4 5 6 5 4 3 2 1

1 2 3 4 5 6 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 9 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 9 10 9 8 7 6 5 4 3 2 1

1. **Test Case 4:**
   * **Input:**

4

* + **Expected Output:**

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1

**PyramidPaƩern.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**public** **class** PyramidPattern {

// write your code here..

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

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

**int** n = s.nextInt();

**for** (**int** i = 1; i <= n; i++) {

**for** (**int** j = 1; j <= n - i; j++) {

System.***out***.print(" ");

}

**for** (**int** j = 1; j <= i; j++) {

System.***out***.print(j + " ");

}

**for** (**int** j = i - 1; j >= 1; j--) {

System.***out***.print(j + " ");

}

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

}

s.close();

}

}

**12.** **Question: Car Rental System - Car Class Design**

**Objective:** Design a class named Car to represent individual cars within a car rental system.

**Class Definition:**

You need to create a class called Car with the following attributes and methods:

**Attributes:**

* **model**: Represents the specific model of the car (e.g., Camry, Accord).
* **year**: Represents the manufacturing year of the car.

**Methods:**

* getModel(): Returns the model of the car.
* getYear(): Returns the manufacturing year of the car.
* setModel(String model): Sets the model of the car.
* setYear(int year): Sets the manufacturing year of the car.

**Sample Test Cases:**

1. **Test Case 1:**
   * **Input:**

Model of car:

Toyota Camry

Manufacturing year of car:

2022

* + **Expected Output:**

Details of the Rental Car:

Model: Toyota Camry

Year: 2022

1. **Test Case 2:**
   * **Input:**

Model of car:

Honda Accord

Manufacturing year of car:

2023

* + **Expected Output:**

Details of the Rental Car:

Model: Honda Accord

Year: 2023

1. **Test Case 3:**
   * **Input:**

Model of car:

Ford Mustang

Manufacturing year of car:

2024

* + **Expected Output:**

Details of the Rental Car:

Year: 2024

1. **Test Case 4:**
   * **Input:**

Model of car:

Ford F-150

Manufacturing year of car:

2016

* + **Expected Output:**

Details of the Rental Car:

Model: Ford F-150

Year: 2016

**Car.java**

**//==================================**

**package** q22623;

**import** java.util.Scanner;

**public** **class** Car {

// write your code here..

**private** String model;

**private** **int** year;

/\*

\* public car(String model, int year) { this.model = model; this.year = year; }

\*/

**public** **void** setModel(String model) {

**this**.model = model;

}

**public** String getModel() {

**return** model;

}

**public** **void** setYear(**int** year) {

**this**.year = year;

}

**public** **int** getYear() {

**return** year;

}

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

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

Car rentalCar = **new** Car();

System.***out***.print("Model of car: ");

String userEnteredModel = scanner.nextLine();

rentalCar.setModel(userEnteredModel);

System.***out***.print("Manufacturing year of car: ");

**int** userEnteredYear = scanner.nextInt();

rentalCar.setYear(userEnteredYear);

// Display details

System.***out***.println("Details of the Rental Car:");

System.***out***.println("Model: " + rentalCar.getModel());

System.***out***.println("Year: " + rentalCar.getYear());

// close the scanner

scanner.close();

}

}