**NAME: SOWJANYA BS**

**DAY 2 - ASSIGNMENT**

**SECTION 1 (Introduction to Loops: The Power of Repetition)**

**Activity 1:** Repetitive Tasks List three tasks you perform regularly that involve repetition. For

each:

1. What is being repeated?

2. What determines when it stops?

**Pseudo code** **:**

Start

Prompt the user to enter the number of minutes to brush

Read the user input and store it as totalminutes

Set minutes brushed = 0

While minutes brushed < totalminutes do

display "brushing teeth... Minute (minutes brushed + 1)"

increment minutes brushed by 1

End while

Display "done brushing!"

End

**Algorithm: steps**

* Start the program.
* Create a Scanner object to accept user input.
* Prompt the user: "Enter how many minutes you want to brush your teeth:"
* Read the input value and store it in totalminutes.
* Initialize minutes brushed to 0.
* While minutesbrushed is less than totalminutes:
* Display: "Brushing teeth... Minute (minutes brushed + 1)"
* Increment minutes brushed by 1.
* After the loop ends, display "Done brushing!".
* End the program

**Code** :

import java.util.Scanner;

public class BrushingTeeth {

public static void main(String[] args) {

Scanner scanner = new Scanner([System.in](http://system.in));

System.out.print("Enter how many minutes you want to brush your teeth: ");

int totalMinutes = scanner.nextInt();

int minutesBrushed = 0 ;

while (minutesBrushed < totalMinutes) {

System.out.println("Brushing teeth... minute " + (minutesBrushed + 1));

minutesBrushed++;

}

System.out.println("Done brushing!");

}

}

TC2 : public class TypingPractice {

public static void main(String[] args) {

int linesTyped = 0;

int maxLines = 10;

while (linesTyped < maxLines) {

System.out.println("Typing practice line " + (linesTyped + 1));

linesTyped++;

}

System.out.println("Typing practice complete!");

}

}

| Test cases | Input | Expected Out put |
| --- | --- | --- |
| TC1 | Enter how many minutes you want to brush your teeth: 2 |  |
| TC2 | Enter how many mails : 5 |  |
| TC3 |  |  |

**Observation**

* This program simulates the real-life task of brushing teeth based on user-defined duration.It uses a while loop to repeat the brushing action for the number of minutes specified.
* The user can input any valid integer to customize the brushing duration.The output shows each minute of brushing and ends with a confirmation message.This demonstrates the use of loops, user input, and counters in Java programming.

**Activity 2:** Code Duplication Write how you would print “Hello!” 10 times without loops. Reflect on how loops make this easier for 1000 times.

**Pseudo code** **:**

Start

Set counter to 1

Repeat while counter is less than or equal to 1000

print "hello!"

increment counter by 1

End repeat

Stop

**Algorithm: steps**

1. Start the program
2. Initialize a counter to 1
3. Repeat the following steps until counter > 1000:
   * Print "Hello!"
   * Increase counter by 1
4. End the program

**Code** :

Using loop Statement

public class HelloPrinter {

public static void main(String[] args) {

for (int i = 1; i <= 1000; i++) {

System.out.println("Hello!");

}

}

}

Without Using loop Statement

public class HelloPrinter {

public static void main(String[] args) {

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

System.out.println("Hello!");

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 | 5 |  |
| TC2 | 1 |  |
| TC3 | 2 |  |

**Observation**

* The program uses a loop to print "Hello!" 1000 times.
* It shows how loops help avoid repetition.
* for loop controls how many times the message is printed.
* Efficient, clean, and easy to modify.

**SECTION 2 (The while Loop - Repeating Based on a Condition)**

**1.**  **Problem Statement** : Countdown Print numbers from 10 to 1, then “Blastoff!”

**Pseudo code** **:**

Start

Read starting number from user

For i from starting number down to 1

print i

End for

Print "blastoff!"

Stop

**Algorithm: steps**

1. Start the program
2. Read starting number from user
3. Loop from starting number down to 1

* Print the current number

1. After the loop ends, print "Blastoff!"
2. End the program

**Code** :

import java.util.Scanner;

public class Countdown {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the starting number for the countdown: ");

int start = scanner.nextInt();

for (int i = start; i >= 1; i--) {

System.out.println(i);

}

System.out.println("Blastoff!");

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program takes a number and counts down to 1.
* It uses a for loop with decrement to control the countdown.
* Ends by printing "Blastoff!" after the loop.
* Shows how loops can manage reverse sequences efficiently.

**2.**  **Problem Statement** : Sum Until Zero Ask users for numbers repeatedly until they enter 0. Sum and print the total.

**Pseudo code** **:**

Start

Set sum to 0

Do

prompt user to enter a number (0 to stop)

read number

add number to sum

While number is not 0

Print total sum

Stop

**Algorithm: steps**

1. Start the program
2. Initialize sum = 0
3. Repeat:

* Ask the user to enter a number
* Read the number
* Add it to sum

1. Continue loop until the user enters 0
2. Print the total sum
3. End the program

**Code** :

import java.util.Scanner;

public class SumUntilZero {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int number;

int sum = 0;

do {

System.out.print("Enter a number (0 to stop): ");

number = scanner.nextInt();

sum += number;

} while (number != 0);

System.out.println("Total sum: " + sum);

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program sums numbers entered by the user until 0 is entered.
* Uses a do-while loop to ensure at least one input is taken.
* Efficiently adds input values using a loop and stops when condition is met.
* Demonstrates how to use loops for repeated input and conditional exit.

**3.**  **Problem Statement** : Guess the Number Generate a random number between 1 and 10. Ask user to guess. Provide feedback and loop until correct.

**Pseudo code** **:**

Start

Generate a random number between 1 and 10

Repeat

ask the user to enter a guess

if guess < number

show "too low"

else if guess > number

show "too high"

else

show "correct!"

Until user guesses the correct number

End

**Algorithm: steps**

1. Start the program.
2. Generate a random number between 1 and 10.
3. Ask the user to input a number (guess).
4. Compare the guess with the random number:

* If the guess is low → show "Too low".
* If the guess is high → show "Too high".
* If correct → show "Correct!" and stop.

1. Repeat steps 3–4 until the correct number is guessed.
2. End the program.

**Code** :

import java.util.Scanner;

import java.util.Random;

public class SimpleGuessGame {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Random random = new Random();

int number = random.nextInt(10) + 1;

int guess = 0;

System.out.println("Guess a number between 1 and 10:");

while (guess != number) {

guess = scanner.nextInt();

if (guess < number) {

System.out.println("Too low! Try again.");

} else if (guess > number) {

System.out.println("Too high! Try again.");

} else {

System.out.println("You guessed it right!");

}

}

scanner.close();

}

}

| Test cases | Input /Expected Output |
| --- | --- |
| TC1 |  |
| TC2 |  |
| TC3 |  |

**Observation**

* The program generates a random number between 1 and 10.
* Uses a do-while loop to keep asking the user until the correct guess.
* Provides feedback: "Too low", "Too high", or "Correct".
* Demonstrates use of conditional statements and looping for validation.

**4.**  **Problem Statement** :Infinite Loop Debugging Analyze and fix:

int counter = 0;

while (counter < 5) {

System.out.println("Hello");

}

**Pseudo code** **:**

Start

Prompt user to enter number of times to print "hello"

Read times

Set counter to 0

While counter < times

print "hello"

increment counter

End while

Stop

**Algorithm: steps**

1. Start the program
2. Ask user how many times to print "Hello"
3. Read the user input into times
4. Set counter = 0
5. Repeat while counter < times:

* Print "Hello"
* Increment counter

1. End the program

**Code** :

import java.util.Scanner;

public class UserInputLoop {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("How many times do you want to print 'Hello' ");

int times = scanner.nextInt();

int counter = 0;

while (counter < times) {

System.out.println("Hello");

counter++;

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program prints "Hello" based on user input.
* Uses a while loop to control repetition.
* Shows how loops and counters help manage repeated tasks.
* Efficiently handles dynamic input for repeated output.

**SECTION 3 ( The for Loop - Iterating Through Sequences)**

**1.**  **Problem Statement** : Even Numbers Print even numbers from 2 to 20 using a for loop.

**Pseudo code** **:**

Start

Prompt user to enter the maximum number (>= 2)

Read max

For i from 2 to max, increment by 2

print i

End for

Stop

**Algorithm: steps**

1. Start the program
2. Ask the user to enter the maximum number (≥ 2)
3. Read the value into max
4. Loop from i = 2 to max, increasing by 2 each time

* Print the current value of i

1. End the program

**Code** :

import java.util.Scanner;

public class EvenNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the maximum number (>= 2): ");

int max = scanner.nextInt();

System.out.println("Even numbers from 2 to " + max + ":");

for (int i = 2; i <= max; i += 2) {

System.out.println(i);

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program prints even numbers from 2 up to a number given by the user.
* Uses a for loop with step i += 2 to skip odd numbers.
* Demonstrates efficient looping and conditional number generation.

**1.**  **Problem Statement** : Factorial Calculator Calculate n! for user input n. Handle edge case when n == 0.

**Pseudo code** **:**

Start

Prompt user to enter a number

Read n

If n < 0 then

Print "factorial not defined for negative numbers"

Else

Set factorial = 1

For i from 1 to n

Factorial = factorial \* i

End for

Print factorial

End if

Stop

**Algorithm: steps**

1. Start the program
2. Ask user to enter a number
3. Read input n
4. If n is negative, print error message
5. Else:

* Initialize factorial = 1
* Loop from 1 to n and multiply factorial by each i

1. Print the final factorial
2. End the program

**Code** :

import java.util.Scanner;

public class FactorialCalculator {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number to calculate its factorial: ");

int n = scanner.nextInt();

if (n < 0) {

System.out.println("Factorial is not defined for negative numbers.");

} else {

long factorial = 1;

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

factorial \*= i;

}

System.out.println("Factorial of " + n + " is: " + factorial);

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program calculates the factorial of a user-input number.
* It handles negative input with a warning message.
* Uses a for loop to multiply numbers from 1 to n.
* Demonstrates use of loops, conditionals, and input handling.

**1.**  **Problem Statement** : Count ‘a’ in String Ask for a string input. Count how many times ‘a’ or ‘A’ appears.

**Pseudo code** **:**

Start

Prompt user to enter a string

Read input string

Set count = 0

For each character in the string

If character is 'a' or 'a' then

Increment count

End for

Print total count of 'a' or 'a'

Stop

**Algorithm: steps**

1. Start the program
2. Ask user to enter a string
3. Read the input string
4. Initialize count = 0
5. Loop through each character in the string

* If character is 'a' or 'A', increase count

1. Print the total count
2. End the program

**Code** :

import java.util.Scanner;

public class CountA {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a string: ");

String input = scanner.nextLine();

int count = 0;

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

char ch = input.charAt(i);

if (ch == 'a' || ch == 'A') {

count++;

}

}

System.out.println("Number of 'a' or 'A' in the string: " + count);

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program counts how many times 'a' or 'A' appears in a string.
* Uses a for loop to check each character.
* Applies conditional checks to match both uppercase and lowercase.
* Demonstrates string traversal and character comparison.

**1.**  **Problem Statement** : Simple Star Pattern Print: \*\*\*\*\* Using one for loop.

**Pseudo code** **:**

Start

Prompt user to enter number of stars

Read n

For i from 1 to n

print "\*"

End for

Stop

**Algorithm: steps**

1. Start the program
2. Ask the user for the number of stars to print
3. Read input n

* Loop from 1 to n

1. Print \* without a newline
2. End the program

**Code** :

import java.util.Scanner;

public class StarPattern {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the number of stars to print: ");

int n = scanner.nextInt();

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

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

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program prints a line of stars based on user input.
* Uses a **for loop** for controlled repetition.
* Demonstrates simple use of loops and user input handling.

**SECTION 4 ( Loop Control Statements & Nested Loops)**

**1.**  **Problem Statement** : Prime Checker Check if a number is prime using a loop and break.

**Pseudo code** **:**

Start

Prompt user to enter a number

Read num

Set isprime = true

If num <= 1 then

Set isprime = false

Else

For i from 2 to num - 1

If num % i == 0 then

Set isprime = false

Break

End for

End if

If isprime then

Print "prime number"

Else

Print "not a prime number"

Stop

**Algorithm: steps**

1. Start the program
2. Ask the user to enter a number
3. Read the number into num
4. Set isPrime = true
5. If num <= 1, set isPrime = false
6. Else, loop from 2 to num - 1:

* If num is divisible by any i, set isPrime = false and break

1. After the loop, check isPrime

* If true → print prime number
* Else → print not a prime

1. End the program

**Code** :

import java.util.Scanner;

public class SimplePrimeCheck {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter a number: ");

int num = scanner.nextInt();

boolean isPrime = true;

if (num <= 1) {

isPrime = false;

} else {

for (int i = 2; i < num; i++) {

if (num % i == 0) {

isPrime = false;

break;

}

}

}

if (isPrime) {

System.out.println(num + " is a prime number.");

} else {

System.out.println(num + " is not a prime number.");

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program checks if a number is prime using a simple loop.
* It uses a boolean flag and break to stop early if a factor is found.
* Handles edge cases like 0 and 1 properly.
* Demonstrates use of loops, conditionals, and flags effectively.

**2.**  **Problem Statement** :Skip Negatives Input 5 numbers. Use continue to skip negative ones and sum the rest.

**Pseudo code** **:**

Start

Set sum = 0

For i from 1 to 5

prompt user to enter number

read num

if num < 0 then

continue to next iteration

end if

add num to sum

End for

Print sum

Stop

**Algorithm: steps**

1. Start the program
2. Set sum = 0
3. Loop 5 times:

* Prompt and read a number
* If the number is negative, skip to the next iteration
* Else, add it to sum

1. After the loop, print the total sum
2. End the program

**Code** :

import java.util.Scanner;

public class SkipNegativeNumbers {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int sum = 0;

System.out.println("Enter 5 number:");

for (int i = 1; i <= 5; i++) {

System.out.print("Num " + i + ": ");

int num = scanner.nextInt();

if (num < 0) {

continue;

}

sum += num;

}

System.out.println("Sum of non-negative numbers: " + sum);

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program takes 5 numbers and skips negative values using continue.
* Only non-negative numbers are added to the sum.
* Demonstrates input filtering with conditional logic inside a loop.
* Efficiently handles unwanted inputs without stopping the program.

**3.**  **Problem Statement** : Rectangle Pattern Input rows and cols, print a rectangle of \*.

**Pseudo code** **:**

Start

Prompt user to enter number of rows

Read rows

Prompt user to enter number of columns

Read cols

For i from 1 to rows

for j from 1 to cols

print "\*"

end for

print new line

End for

Stop

**Algorithm: steps**

1. Start the program
2. Ask user for number of rows and columns
3. Read rows and cols
4. Use nested loops:

* Outer loop from 1 to rows (controls rows)
* Inner loop from 1 to cols (prints stars in each row)
* After inner loop, print a new line

1. End the program

**Code** :

import java.util.Scanner;

public class RectanglePattern {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter number of rows: ");

int rows = scanner.nextInt();

System.out.print("Enter number of columns: ");

int cols = scanner.nextInt();

for (int i = 1; i <= rows; i++) {

for (int j = 1; j <= cols; j++) {

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

}

System.out.println();

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program prints a rectangle of stars based on user input for rows and columns.
* Uses nested loops: outer loop for rows, inner loop for columns.
* Demonstrates basic pattern printing and loop nesting.

**3.**  **Problem Statement** :Triangle Pattern Input height. Print right-angled triangle with \*.

**Pseudo code** **:**

Start

Prompt user to enter height of triangle

Read height

For i from 1 to height

for j from 1 to i

print "\*"

end for

print new line

End for

Stop

**Algorithm: steps**

1. Start the program
2. Ask the user to enter the triangle height
3. Read the height
4. Loop from i = 1 to height:

* Inner loop from j = 1 to i → print stars
* After inner loop, print a new line

1. End the program

**Code** :

import java.util.Scanner;

public class TrianglePattern {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the height of the triangle: ");

int height = scanner.nextInt();

for (int i = 1; i <= height; i++) {

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

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

}

System.out.println();

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |
| TC3 |  |  |

**Observation**

* The program prints a right-angled triangle pattern using stars (\*).
* Uses nested loops: outer loop for rows, inner loop for stars per row.
* Demonstrates pattern building with incremental printing.

**4.**  **Problem Statement** :: Pyramid Pattern Challenge Input height. Print centered pyramid

**Pseudo code** **:**

Start

Prompt user to enter height of pyramid

Read height

For i from 1 to height

for space from 1 to (height - i)

print space

end for

for star from 1 to (2 \* i - 1)

print "\*"

end for

print new line

End for

Stop

**Algorithm: steps**

1. Start the program
2. Ask user to enter the height of the pyramid
3. Read the height
4. Loop from i = 1 to height:

* Print (height - i) spaces to center the stars
* Print (2 \* i - 1) stars to form the pyramid row
* Move to the next line

1. End the program

**Code** :

import java.util.Scanner;

public class PyramidPattern {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.print("Enter the height of the pyramid: ");

int height = scanner.nextInt();

for (int i = 1; i <= height; i++) {

for (int space = 1; space <= height - i; space++) {

System.out.print(" ");

}

for (int star = 1; star <= 2 \* i - 1; star++) {

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

}

System.out.println();

}

scanner.close();

}

}

| Test cases | Input | Expected Output |
| --- | --- | --- |
| TC1 |  |  |
| TC2 |  |  |

**Observation**

* The program prints a centered pyramid pattern using stars (\*).
* Uses nested loops to handle spaces and stars separately.
* Demonstrates control of alignment and symmetry using math in loops.