**Task 1 –** **countdown**

**Algorithm**

1. **Use a for loop to print numbers from 10 down to 1.**
2. **After the loop, print "Blastoff!".**
3. **Use System.out.println() for all outputs.**

**Psudocode**

START

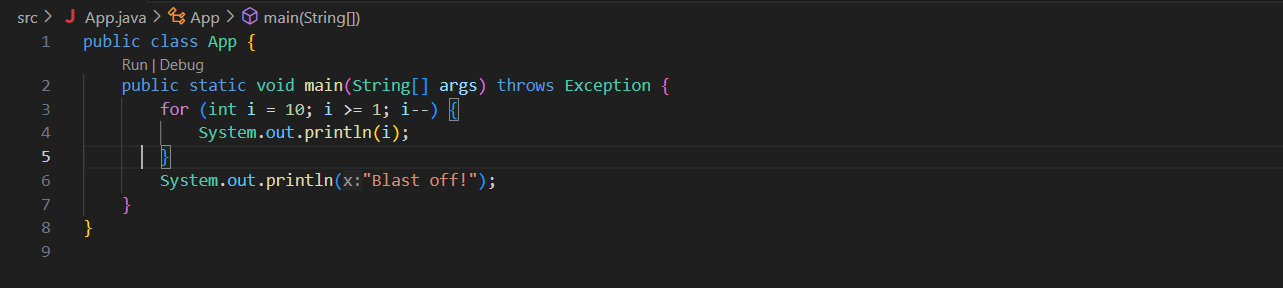
FOR i = 10 to 1

PRINT i

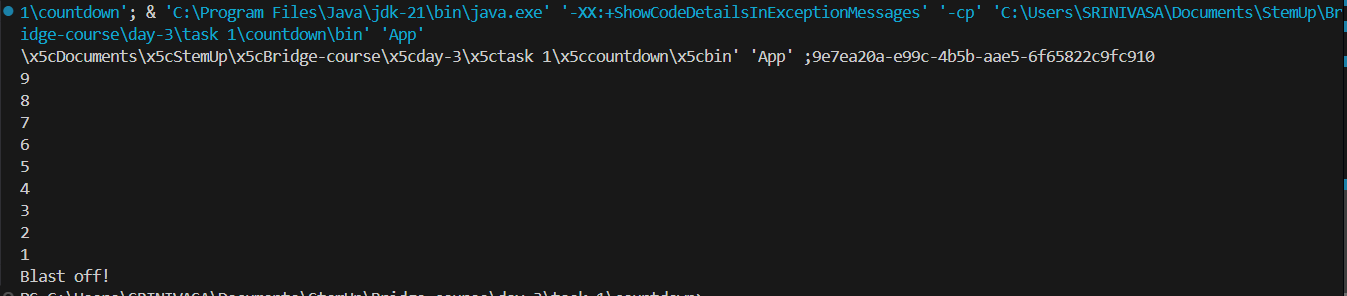
END FOR

PRINT "Blastoff!"

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

 Demonstrates use of a **descending for loop** in Java.

 Clearly prints each number, followed by **"Blastoff!"**.

 Reinforces control flow, loop structure, and output syntax in Java.

 Simple and effective for understanding countdown logic.

**Task 2 –** **Sum Until Zero**

**Algorithm**

 Import the Scanner class.

 Initialize sum = 0.

 Use a while loop to read numbers repeatedly.

 Stop when the user enters 0.

 Print the final sum.

**Psudocode**

START

IMPORT Scanner

INITIALIZE sum = 0

PROMPT for number

WHILE number != 0

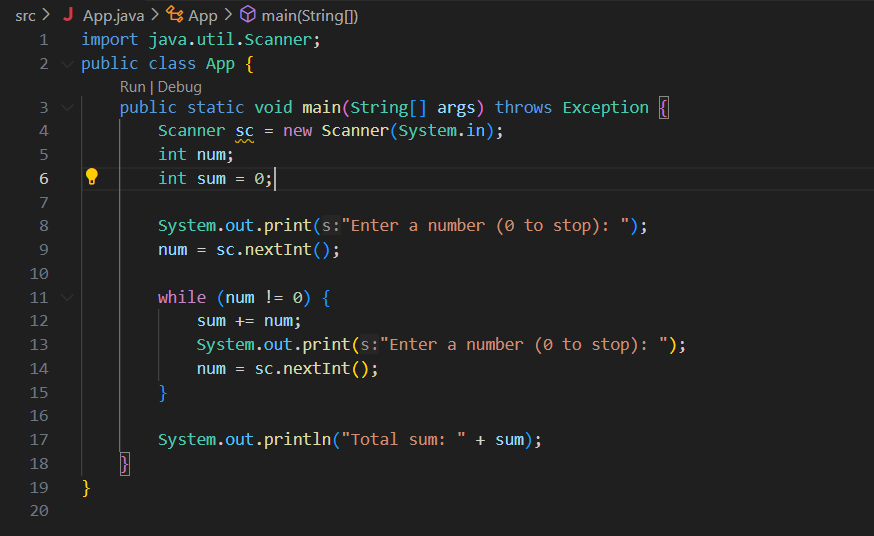
ADD number to sum

PROMPT for number

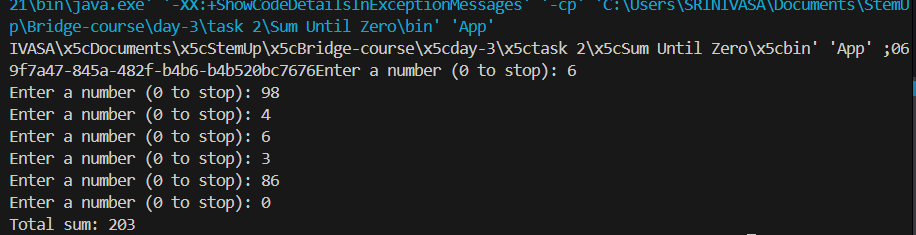
END WHILE

PRINT sum

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

 Uses a **while loop** to keep reading numbers until 0 is entered.

 Demonstrates accumulation logic (sum += num).

 A classic use case for sentinel-controlled loops.

 Helps build understanding of loop entry conditions and user-driven iteration.

**Task 3 –** **Guess the Number**

**Algorithm** Import Scanner and Random.

 Generate a random number between 1 and 10.

 Ask the user to guess the number.

 Use a while loop:

* If the guess is incorrect, prompt again.
* If correct, display success and exit.

**Psudocode**

START

IMPORT Scanner and Random

GENERATE random number between 1 and 10

PROMPT user to guess

WHILE guess != number

IF guess < number → PRINT "Too low"

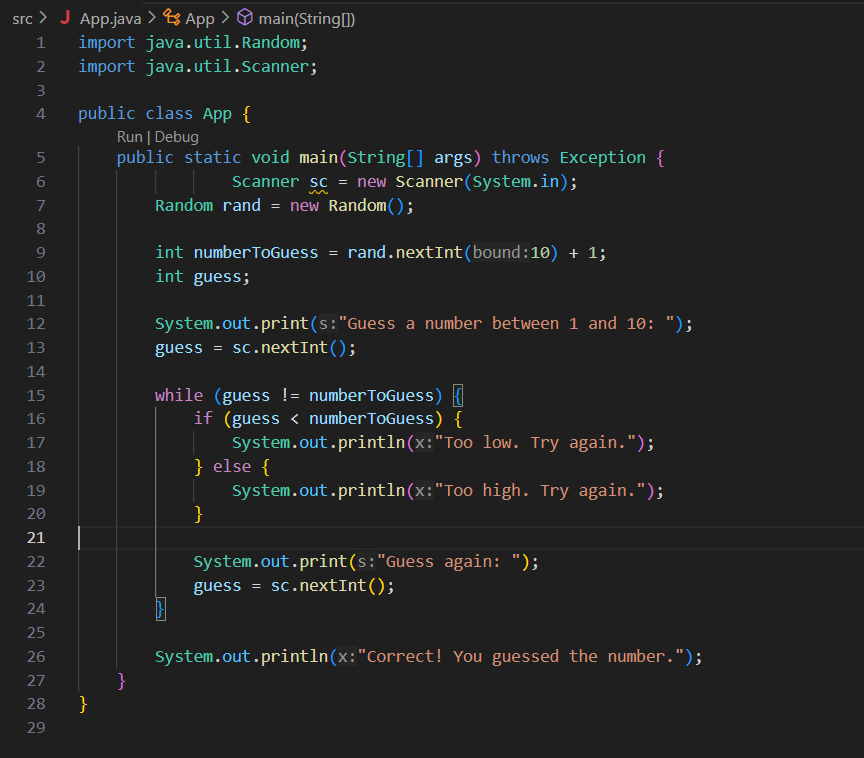
IF guess > number → PRINT "Too high"

PROMPT again

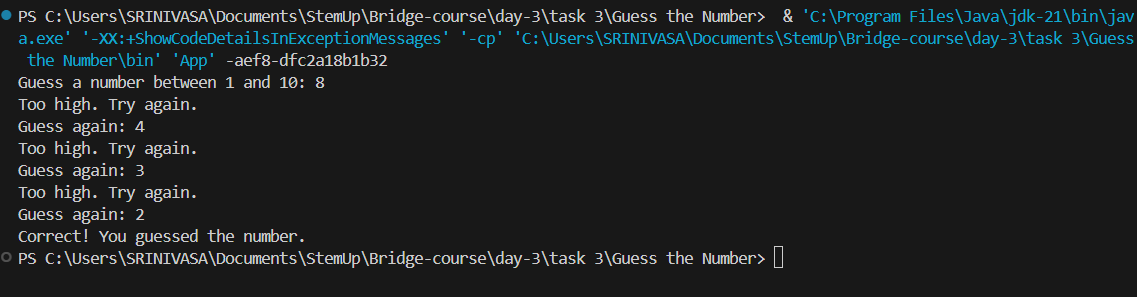
END WHILE

PRINT "Correct!"

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

 Uses Random to generate a number between 1–10 (rand.nextInt(10) + 1).

 Loops until the user **guesses correctly**.

 Gives real-time feedback — **"Too high" or "Too low"**.

 Reinforces loop control, conditionals, and random logic.

**Task 4 –** **Infinite Loop Debugging**

**Algorithm** Initialize counter = 0.

 Use while (counter < 5) to repeat 5 times.

 Print "Hello" each time.

 Increment counter by 1 on each loop iteration.

**Psudocode**

START

SET counter = 0

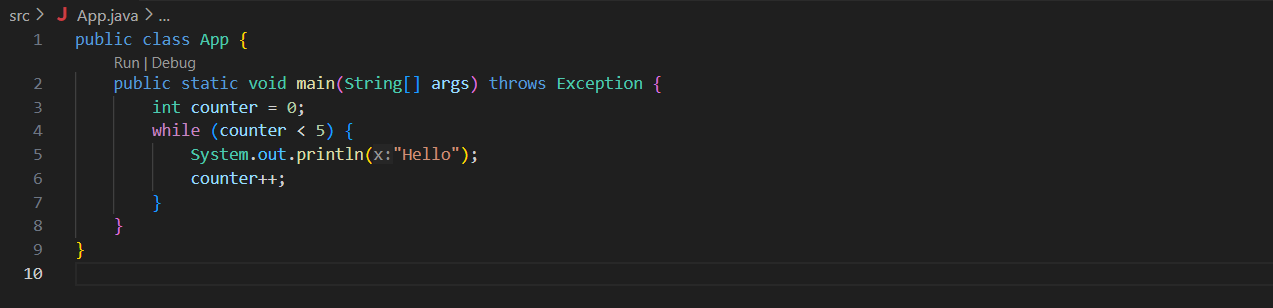
WHILE counter < 5

PRINT "Hello"

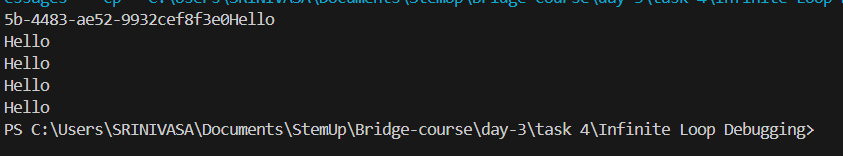
INCREMENT counter

END WHILE

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

* **Infinite loop bug** fixed by adding counter++ inside the loop body.
* Ensures that the loop condition becomes **false after 5 iterations**.
* Teaches the importance of **loop control variables** and **exit conditions** in while loops.

**Task 5 –** **Even Numbers**

**Algorithm** Use a for loop starting from 2 and ending at 20.

 Increment by 2 in each step.

 Print the current number in each iteration using System.out.println().

**Psudocode**

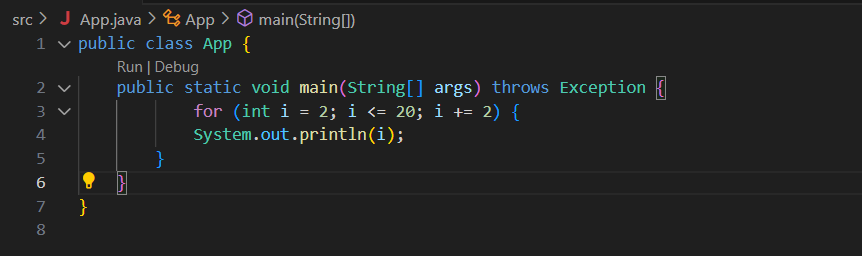
START

FOR i = 2 to 20 STEP 2

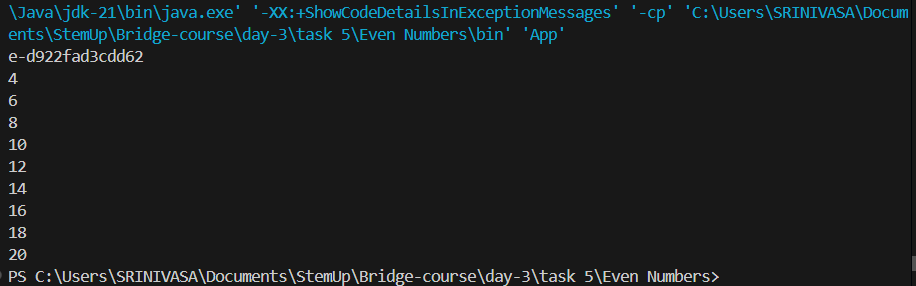
PRINT i

END FOR

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

 Loop uses i += 2 for printing only **even numbers**.

 Starts at 2 and stops at 20 (inclusive).

 Simple demonstration of for loop control and number patterns.

 Great for understanding step size in loop progression.

**Task 6 – Factorial Calculator**

**Algorithm** Import Scanner for input.

 Prompt the user to enter an integer n.

 If n == 0, return 1 (since 0! = 1).

 Otherwise, use a loop to calculate n!.

 Display the result using System.out.println().

**Psudocode**

START

IMPORT Scanner

PROMPT user to enter n

READ n

IF n == 0

factorial = 1

ELSE

SET factorial = 1

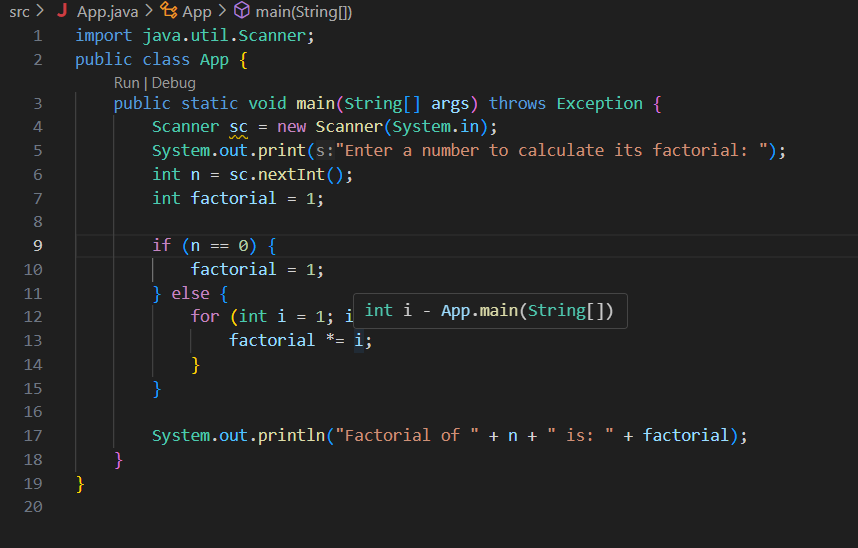
FOR i = 1 to n

factorial = factorial \* i

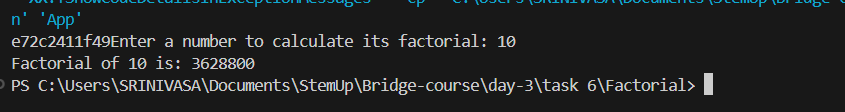
END IF

PRINT factorial

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

* Handles the **edge case** of 0! = 1 correctly.
* Uses a **for loop** for multiplication from 1 to n.
* Reinforces control structures and logic for accumulation patterns.
* Very common DSA question — foundational for recursion, loops, and math-based logic.

**Task 7 – Count ‘a’ in String**

**Algorithm**

1. **Import Scanner.**
2. **Prompt the user to enter a string.**
3. **Initialize a counter to 0.**
4. **Loop through each character of the string.**
5. **If character is 'a' or 'A', increment the counter.**
6. **Display the total count.**

**Psudocode**

START

IMPORT Scanner

PROMPT user for a string

READ input string

SET count = 0

FOR each character in string

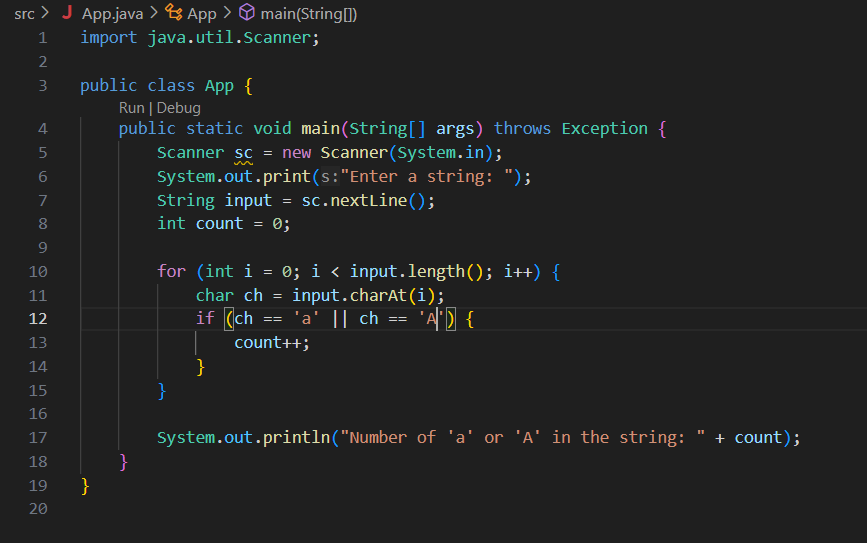
IF character == 'a' OR character == 'A'

INCREMENT count

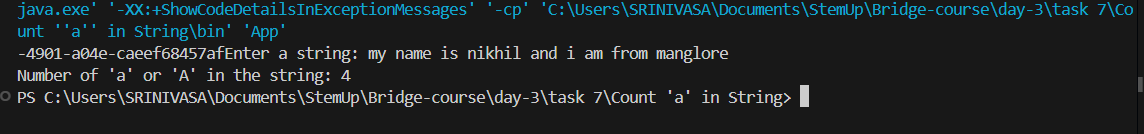
END FOR

PRINT count

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

 Uses a for loop and charAt() to access each character.

 Checks both lowercase and uppercase 'a'.

 A good example of **string traversal** and **character comparison** in Java.

 Strengthens understanding of working with strings and conditionals.

**Task 8 – Star print**

**Algorithm** Use a for loop that runs 5 times.

 Print a \* without a newline on each iteration using print().

 After the loop, print a newline.

**Psudocode**

START

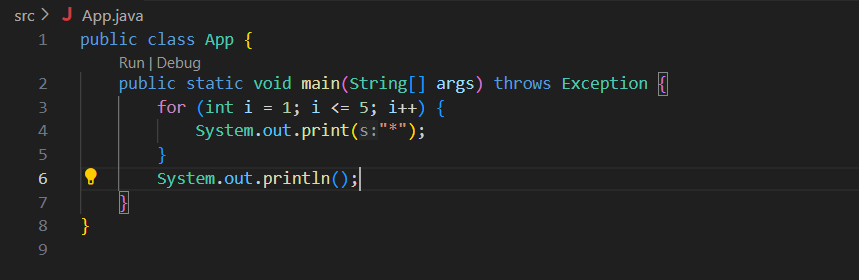
FOR i = 1 to 5

PRINT "\*"

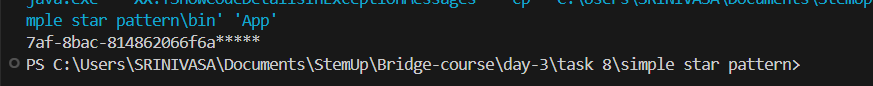
END FOR

PRINT newline

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

* Uses System.out.print() to print on the **same line**.
* System.out.println() after the loop ensures the cursor moves to the next line.
* A basic and clean intro to **pattern printing** using loops.

**Task 9 –** **Prime checker**

**Algorithm** Import Scanner for user input.

 Read an integer from the user.

 If the number is less than or equal to 1, it's not prime.

 Use a loop from 2 to n/2:

* If any number divides n, it’s **not prime** → break the loop.

 Use a boolean flag or break logic to print the result.

**Psudocode**

START

PROMPT user for number

IF number <= 1 → PRINT not prime

ELSE

FOR i = 2 to n/2

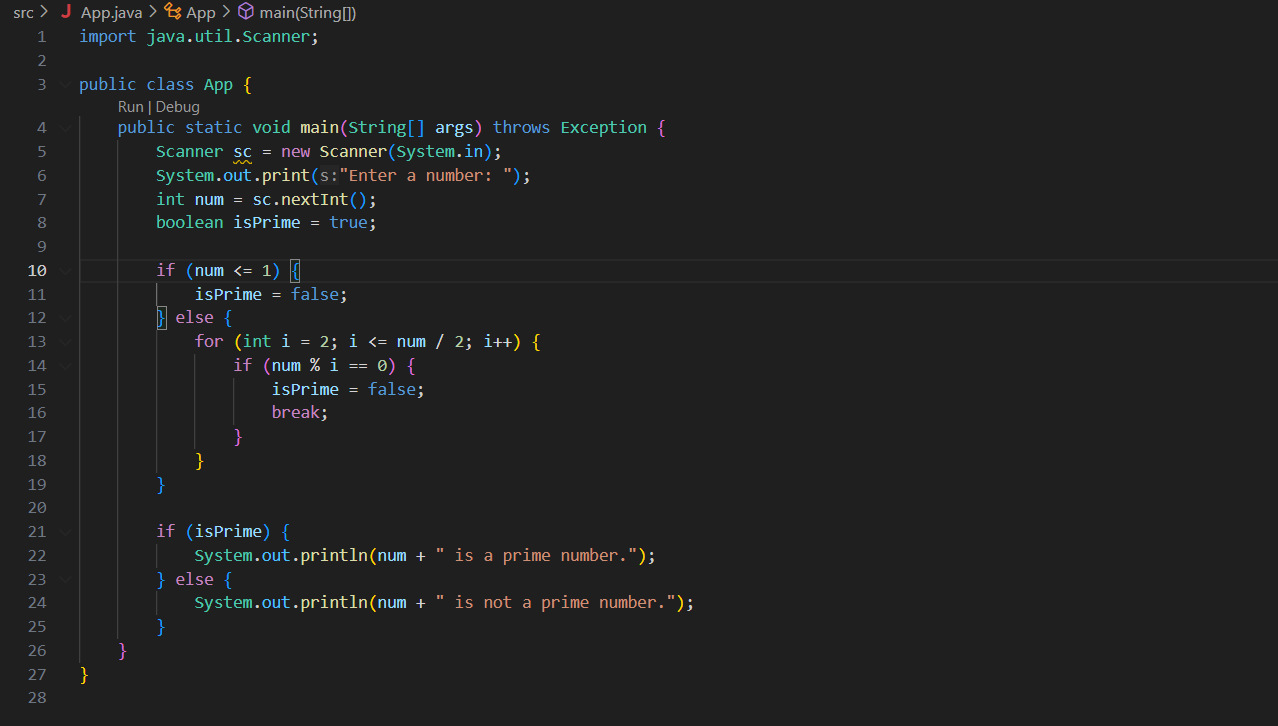
IF n % i == 0

PRINT not prime

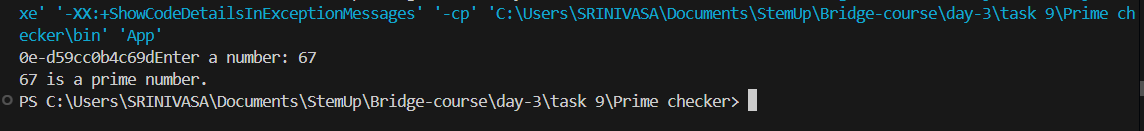
BREAK

IF loop completes → PRINT prime

END

**Code(javascript)  
**

**Output (test case 1)**



**Observations**

* Uses a **for loop** and **break** for early exit when a divisor is found.
* Efficiently checks divisibility up to num / 2 (can be optimized further to √n).
* Demonstrates control flow, conditional logic, and loop control.