Exercise 1

\*

Error

There is an extra semicolon (;) after the while condition, which terminates the loop prematurely.

The increment of i (++i) is outside the loop block.

Correction

Remove the semicolon after the while condition

Place the ++i statement inside the loop block to ensure the variable i is incremented during each iteration.

\*

Error:

The variable k is a floating-point number (like double), but the loop condition k != 1.0 can lead to floating-point precision issues.

Due to the way floating-point numbers are represented in computers, adding 0.1 repeatedly may not exactly result in 1.0, causing an infinite loop or unexpected behavior.

Correction:

Use a condition that avoids floating-point precision problems, such as using a for loop with a fixed number of iterations, or comparing the value to a threshold instead of directly checking k != 1.0.

\*

Error:

The switch statement is missing a break after case 1:, so it will "fall through" and execute the code for case 2: as well, even if n is 1.

The fall-through behavior in switch cases is a common mistake if you intend each case to be independent.

Correction:

Add a break; statement after the code in case 1: to prevent fall-through

\*

switch (n) {

case 1:

System.out.println("The number is 1");

break; // Added break here

case 2:

System.out.println("The number is 2");

break;

default:

System.out.println("The number is not 1 or 2");

break;

}

\*

int n = 1;

mientras (n <= 10)

System.out.println(n++);

\*

Control Variable Initialization – Set the starting value for the loop control variable.

Loop-Continuation Condition – Defines when the loop should continue.

Control Variable Update – Modifies the control variable to eventually make the condition false.

Loop Body – The set of statements that execute repeatedly while the condition is true.

\*

while loop:

Best used when the number of repetitions is not known beforehand

Initialization, condition, and update can be separated and placed anywhere.

for loop:

Ideal for counter-controlled repetition.

All three control elements (initialization, condition, update) are in one line, making it more compact.

\*

int number;

do {

number = input.nextInt();

} while (number <= 0);

\*

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

if (i == 3) continue; // skips printing 3

System.out.println(i);

}

\*

for (int i = 100; i >= 1; i--)

System.out.println(i);

\*

switch (value % 2) {

case 0:

System.out.println("Even integer");

break;

case 1:

System.out.println("Odd integer");

break;

}

\*

for (int i = 19; i >= 1; i -= 2)

System.out.println(i);

\*

int counter = 2;

do {

System.out.println(counter);

counter += 2;

} while (counter <= 100);

\*

This program prints a rectangle made of @ symbols.

Outer loop (i = 1 to 10) controls the number of rows.

Inner loop (j = 1 to 5) prints 5 '@' symbols per row.

System.out.println(); moves to the next line after each row.

\*

The formula for compound growth is:

𝐴

=

𝑃

×

(

1

+

𝑟

)

𝑡

A=P×(1+r)

t

Where:

𝐴

A is the future value (target number of users).

𝑃

P is the initial value (current number of users).

𝑟

r is the growth rate per period (monthly growth rate).

𝑡

t is the number of periods (months) required to reach the target.

Given:

Initial number of users,

𝑃

=

1

P=1 billion.

Monthly growth rate,

𝑟

=

4

%

=

0.04

r=4%=0.04.

We want to find the time it will take to reach

𝐴

=

1.5

A=1.5 billion and

𝐴

=

2

A=2 billion.

We need to solve for

𝑡

t, the number of months, in each case.

Step 1: Formula Rearrangement

We need to isolate

𝑡

t (the number of months) in the compound growth formula.

𝐴

=

𝑃

×

(

1

+

𝑟

)

𝑡

A=P×(1+r)

t

Dividing both sides by

𝑃

P:

𝐴

𝑃

=

(

1

+

𝑟

)

𝑡

P

A

​

=(1+r)

t

Taking the natural logarithm (

ln

⁡

ln) of both sides:

ln

⁡

(

𝐴

𝑃

)

=

𝑡

×

ln

⁡

(

1

+

𝑟

)

ln(

P

A

​

)=t×ln(1+r)

Now, solving for

𝑡

t:

𝑡

=

ln

⁡

(

𝐴

𝑃

)

ln

⁡

(

1

+

𝑟

)

t=

ln(1+r)

ln(

P

A

​

)

​

Step 2: Calculation for 1.5 Billion Users

For

𝐴

=

1.5

A=1.5 billion:

𝑡

=

ln

⁡

(

1.5

1

)

ln

⁡

(

1

+

0.04

)

t=

ln(1+0.04)

ln(

1

1.5

​

)

​

𝑡

=

ln

⁡

(

1.5

)

ln

⁡

(

1.04

)

t=

ln(1.04)

ln(1.5)

​

Now, calculating the logarithms:

ln

⁡

(

1.5

)

≈

0.4055

,

ln

⁡

(

1.04

)

≈

0.0392

ln(1.5)≈0.4055,ln(1.04)≈0.0392

𝑡

=

0.4055

0.0392

≈

10.34

months

t=

0.0392

0.4055

​

≈10.34months

So, it will take approximately 10.34 months for Facebook to grow from 1 billion users to 1.5 billion users.

Step 3: Calculation for 2 Billion Users

For

𝐴

=

2

A=2 billion:

𝑡

=

ln

⁡

(

2

1

)

ln

⁡

(

1

+

0.04

)

t=

ln(1+0.04)

ln(

1

2

​

)

​

𝑡

=

ln

⁡

(

2

)

ln

⁡

(

1.04

)

t=

ln(1.04)

ln(2)

​

Now, calculating the logarithms:

ln

⁡

(

2

)

≈

0.6931

,

ln

⁡

(

1.04

)

≈

0.0392

ln(2)≈0.6931,ln(1.04)≈0.0392

𝑡

=

0.6931

0.0392

≈

17.67

months

t=

0.0392

0.6931

​

≈17.67months