Assignment No – 3

Snippet 1

public class InfiniteForLoop {

public static void main(String[] args) {

for (int i = 0; i < 10; i--) {

System.out.println(i);

}

}

}

The for loop initializes i at 0 and has a condition i < 10, meaning it should terminate when i reaches 10.

However, the loop decrements i (i--) instead of incrementing it (i++).

Since i starts at 0 and decreases indefinitely (i = -1, -2, -3,...), it will never reach 10, causing an infinite loop.

To ensure the loop runs exactly 10 times, the control variable i should increment (i++) rather than decrement (i--):

Corrected Code

public class FixedForLoop {

public static void main(String[] args) {

for (int i = 0; i < 10; i++) {

System.out.println(i);

}

}

}

Output

0

1

2

3

4

5

6

7

8

9

------------------------------------------------------------------------------------------------------------------

Snippet 2

public class IncorrectWhileCondition {

public static void main(String[] args) {

int count = 5;

while (count = 0) {

System.out.println(count);

count--;

}

}

}

Error

IncorrectWhileCondition.java:5: error: incompatible types: int cannot be converted to boolean

while (count = 0) {

^

In Java, the condition inside a while loop must be a boolean expression (i.e., true or false).

However, count = 0 assigns 0 to count instead of comparing it.

Since 0 is an integer, not a boolean, Java throws an "incompatible types" error.

Use == for comparison instead of =:

while (count == 0)

However, since count starts at 5, while (count == 0) will never be true, meaning the loop won't run.

If the goal is to decrement count until it reaches 0, the correct condition is:

Corrected Code

public class CorrectWhileCondition {

public static void main(String[] args) {

int count = 5;

while (count > 0) {

System.out.println(count);

count--;

}

}

}

Output

5

4

3

2

1

---------------------------------------------------------------------------------------------------------------------Snippet 3

public class DoWhileIncorrectCondition {

public static void main(String[] args) {

int num = 0;

do {

System.out.println(num);

num++;

} while (num > 0);

}

}

The condition while (num > 0); causes an infinite loop because num keeps increasing indefinitely.

If the goal is to run only once,change the condition:

while (num < 0);

Corrected Code

public class DoWhileFixed {

public static void main(String[] args) {

int num = 0;

do {

System.out.println(num);

num++;

} while (num < 0);

}

}

Output

0

If the goal is to print numbers until a certain limit, modify the condition (e.g., num < 5 to print numbers from 0 to 4):

public class DoWhileFixed {

public static void main(String[] args) {

int num = 0;

do {

System.out.println(num);

num++;

} while (num < 5);

}

}

Output

0

1

2

3

4

---------------------------------------------------------------------------------------------------------------------Snippet 4

public class OffByOneErrorForLoop {

public static void main(String[] args) {

for (int i = 1; i <= 10; i++) {

System.out.println(i);

}

// Expected: 10 iterations with numbers 1 to 10

// Actual: Prints numbers 1 to 10, but the task expected only 1 to 9

}

}

//

The loop runs from 1 to 10 (i <= 10), meaning it prints numbers 1 to 10.

However, the task expected only 1 to 9 (indicating an off-by-one error).

Change the condition from i <= 10 to i < 10 so it stops at 9.

Corrected Code

public class OffByOneErrorForLoop {

public static void main(String[] args) {

for (int i = 1; i < 10; i++) {

System.out.println(i);

}

}

}

Output

1

2

3

4

5

6

7

8

9

---------------------------------------------------------------------------------------------------------------------Snippet 5

public class WrongInitializationForLoop {

public static void main(String[] args) {

for (int i = 10; i >= 0; i++) {

System.out.println(i);

}

}

}

The loop initializes i = 10 and runs while i >= 0.

However, i++ increases i instead of decreasing it.

Since i starts at 10 and keeps increasing (11, 12, 13...), the condition i >= 0 is always true, leading to an infinite loop.

Change i++ to i-- so i decreases instead of increasing.

Corrected Code

public class WrongInitializationForLoop {

public static void main(String[] args) {

for (int i = 10; i >= 0; i--) {

System.out.println(i);

}

}

}

Output

10

9

8

7

6

5

4

3

2

1

0

---------------------------------------------------------------------------------------------------------------------

Snippet 6

public class MisplacedForLoopBody {

public static void main(String[] args) {

for (int i = 0; i < 5; i++)

System.out.println(i);

System.out.println("Done");

}

}

The for loop only controls the first statement (System.out.println(i);) because it lacks curly braces {}.

As a result, System.out.println("Done"); is outside the loop and executes only once, after the loop finishes.

Enclose the loop body inside {} to include both statements.

Corrected Code

public class MisplacedForLoopBody {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) { // Added curly braces

System.out.println(i);

System.out.println("Done");

}

}

}

Output

0

Done

1

Done

2

Done

3

Done

4

Done

---------------------------------------------------------------------------------------------------------------------Snippet 7

public class UninitializedWhileLoop {

public static void main(String[] args) {

int count;

while (count < 10) {

System.out.println(count);

count++;

}

}

}

The variable count is declared but not initialized (int count;).

In Java, local variables must be initialized before use.

The condition while (count < 10) tries to use count, causing a compilation error.

Initialize count before using it in the loop.

Corrected Code

public class UninitializedWhileLoop {

public static void main(String[] args) {

int count = 0;

while (count < 10) {

System.out.println(count);

count++;

}

}

}

Output

0

1

2

3

4

5

6

7

8

9

---------------------------------------------------------------------------------------------------------------------Snippet 8

public class OffByOneDoWhileLoop {

public static void main(String[] args) {

int num = 1;

do {

System.out.println(num);

num--;

} while (num > 0);

}

}

The loop starts with num = 1 and decrements (num--) inside the do block.

After printing 1, num becomes 0.

The condition while (num > 0) fails immediately, causing the loop to terminate after just one iteration.

This is an off-by-one error, likely caused by decrementing instead of incrementing.

If the goal is to print numbers from 1 to 10, increment num instead of decrementing (num++).

Change while (num > 0) to while (num <= 10).

Corrected Code

public class OffByOneDoWhileLoop {

public static void main(String[] args) {

int num = 1;

do {

System.out.println(num);

num++;

} while (num <= 10);

}

}

Output

1

2

3

4

5

6

7

8

9

10

---------------------------------------------------------------------------------------------------------------------Snippet 9

public class InfiniteForLoopUpdate {

public static void main(String[] args) {

for (int i = 0; i < 5; i += 2) {

System.out.println(i);

}

}

}

The loop does not run infinitely but might not behave as expected due to its update step.

The update i += 2 increments i by 2 each time (0, 2, 4).

If the expectation was to print 0 to 4 sequentially (0, 1, 2, 3, 4), this is an off-by-one error.

If the goal is to print all numbers from 0 to 4, change i += 2 to i++:

for (int i = 0; i < 5; i++) // Changed i += 2 to i++

If the goal is to print even numbers only (0, 2, 4), the original code is correct.

Corrected Code (Sequential 0 to 4)

public class CorrectForLoop {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) {

System.out.println(i);

}

}

}

Output

0

1

2

3

4

---------------------------------------------------------------------------------------------------------------------Snippet 10

public class IncorrectWhileLoopControl {

public static void main(String[] args) {

int num = 10;

while (num = 10) {

System.out.println(num);

num--;

}

}

}

The condition while (num = 10) is incorrect because = is an assignment operator, not a comparison.

In Java, conditions in loops must be boolean expressions, but num = 10 assigns 10 to num, which is an integer, causing a compilation error.

Use == for comparison instead of =:

while (num == 10)

However, even with this fix, the loop would run only once because num-- makes num 9 after the first iteration.

Corrected Code (Counts Down from 10 to 1)

public class CorrectWhileLoop {

public static void main(String[] args) {

int num = 10;

while (num > 0) {

System.out.println(num);

num--;

}

}

}

Output

10

9

8

7

6

5

4

3

2

1

---------------------------------------------------------------------------------------------------------------------Snippet 11

public class IncorrectLoopUpdate {

public static void main(String[] args) {

int i = 0;

while (i < 5) {

System.out.println(i);

i += 2; // Error: This may cause unexpected results in output

}

}

}

The update i += 2 increments i by 2 in each iteration.

This means the loop prints only even numbers (0, 2, 4), skipping 1 and 3.

If the goal was to print all numbers from 0 to 4 sequentially, this is an off-by-one error.

Output of the Given Code

0

2

4

i = 0 → prints 0, then i += 2 → i = 2

i = 2 → prints 2, then i += 2 → i = 4

i = 4 → prints 4, then i += 2 → i = 6 (loop stops because i < 5 is false)

Fix

If the goal is to print all numbers from 0 to 4, change i += 2 to i++:

public class CorrectLoopUpdate {

public static void main(String[] args) {

int i = 0;

while (i < 5) {

System.out.println(i);

i++;

}

}

}

Expected Output

0

1

2

3

4

---------------------------------------------------------------------------------------------------------------------Snippet 12

public class LoopVariableScope {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) {

int x = i \* 2;

}

System.out.println(x); // Error: 'x' is not accessible here

}

}

The variable x is declared inside the for loop, meaning it only exists within that loop (block scope).

After the loop ends, x is out of scope, so System.out.println(x); causes a compilation error:

error: cannot find symbol

System.out.println(x);

^

Declare x outside the loop so it remains accessible after the loop ends.

Corrected Code (If x is Needed After the Loop)

public class LoopVariableScope {

public static void main(String[] args) {

int x = 0;

for (int i = 0; i < 5; i++) {

x = i \* 2;

}

System.out.println(x);

}

}

Output

8

Alternative (If x is Only Needed Inside the Loop)

public class LoopVariableScope {

public static void main(String[] args) {

for (int i = 0; i < 5; i++) {

int x = i \* 2; // x is valid only inside the loop

System.out.println(x); // Print inside the loop

}

}

}

Output

0

2

4

6

8