SECTION 1: Error-Driven Learning Assignment: Loop Errors

Instructions:

Analyze each code snippet for errors or unexpected behavior. For each snippet, determine:

1. Why does the error or unexpected behavior occur?

2. How can the code be corrected to achieve the intended behavior?

###################################################################

Snippet 1:

public class InfiniteForLoop {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

// Error to investigate: Why does this loop run infinitely? How should the loop control variable be adjusted?

**ANS**:=>The loop runs infinitely because here we initialized the variable ‘i’ as 0 and its decremented so it goes in minus that’s why it happen.

public class InfiniteForLoop {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

Snippet 2:

public class IncorrectWhileCondition {

public static void main(String[] args) {

int count = 5;

while (count = 0) {

System.out.println(count);

count--;

}

}

}

// Error to investigate: Why does the loop not execute as expected? What is the issue with the condition in the `while` loop?

**ANS**:=>while loop check the condition either true or false so it consider Boolean and here we assign the 0 int value to count variable, so it gives incompatible data types error.

The while loop should be declared as while(count!=0) so condition is true.

Snippet 3:

public class DoWhileIncorrectCondition {

public static void main(String[] args) {

int num = 0;

do {

System.out.println(num);

num++;

} while (num > 0);

}

}

// Error to investigate: Why does the loop only execute once? What is wrong with the loop condition in the `dowhile` loop?

**ANS**:=>loop does not execute once,it goes infinitely, here in while condition the num > 0 so always condition becomes true.

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

}

}

// Error to investigate: What is the issue with the loop boundaries? How should the loop be adjusted to meet the expected output?

**ANS**;=>here i<=10 that’s why it printed 1 to 10

If we declare it as i< 10 then it will print 1 to 9

Snippet 5:

public class WrongInitializationForLoop {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

// Error to investigate: Why does this loop not print numbers in the expected order? What is the problem with the initialization and update statements in the `for` loop?

**ANS**:=>the loop goes infinitely ,every time the condition becomes true,

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");

}

}

// Error to investigate: Why does "Done" print only once, outside the loop? How should the loop body be enclosed to include all statements within the loop?

**ANS**;=>because after for loop only first statement consider within for loop, second sop statement is outside the for loop that’s wht it printed only once.

The loop body should enclosed within { } bracket to include all statement within for loop.

Snippet 7:

public class UninitializedWhileLoop {

public static void main(String[] args) {

int count;

while (count < 10) {

System.out.println(count);

count++;

}

}

}

// Error to investigate: Why does this code produce a compilation error? What needs to be done to initialize the loop variable properly?

**ANS**:=>The count variable is not initialized that’s why it gives compilation errors,we need to assign some value to count variable.

Snippet 8:

public class OffByOneDoWhileLoop {

public static void main(String[] args) {

int num = 1;

do {

System.out.println(num);

num--;

} while (num > 0);

}

}

// Error to investigate: Why does this loop print unexpected numbers? What adjustments are needed to print the numbers from 1 to 5?

**ANS**:=>

public class OffByOneDoWhileLoop {

public static void main(String[] args) {

int num = 1;

do {

System.out.println(num);

num++;

} while (num <=5);

}

}

Snippet 9:

public class InfiniteForLoopUpdate {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

// Error to investigate: Why does the loop print unexpected results or run infinitely? How should the loop update expression be corrected?

**ANS**:=>The code gives 0 2 4 output.it not run infinitely.

public class InfiniteForLoopUpdate {

public static void main(String[] args) {

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

System.out.println(i);

}

}

}

Snippet 10:

public class IncorrectWhileLoopControl {

public static void main(String[] args) {

int num = 10;

while (num = 10) {

System.out.println(num);

num--;

}

}

}

// Error to investigate: Why does the loop execute indefinitely? What is wrong with the loop condition?

**ANS**:=>The loop does not execute indefinitely, it gives compilation error as incompatible types:int can not be converted to Boolean.

In while loop here 10 value assign to num variable.while loop checks wheather the condition is true or false.

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

}

}

}

// Error to investigate: What will be the output of this loop? How should the loop variable be updated to achieve the desired result?

ANS:=>The output is 0 2 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

}

}

// Error to investigate: Why does the variable 'x' cause a compilation error? How does scope

**ANS**:=>Here the scope of x is within the for loop only ,so we cant access the value of x outside the loop.