**SECTION 1: Error-Driven Learning Assignment: Loop Errors**

**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?** The loop condition is i < 10, meaning the loop continues running as long as i is less than 10. The update statement is i--, which decreases i in each iteration. Since i starts at 0 and is continuously decreasing, it will always be less than 10. As a result, the loop never terminates because i never reaches or exceeds 10.

**FIX:**

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?**

count = 0 is an assignment statement, not a comparison. Its consider as integer and integer not converted Boolean type . so it show incompatible error.

**FIX:**

public class IncorrectWhileCondition {

public static void main(String[] args) {

int count = 5;

while (count > 0) {

System.out.println(count);

count--;

} } }

**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

} }

**EXPLANATION for expected output:**

Just change condition means I <= 9 then the iteration goes 1 to 9

**FIX:**

public class OffByOneErrorForLoop {

public static void main(String[] args) {

for (int i = 1; i <= 9; 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

} }

**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?**

After the first iteration, i++ makes i = 11, which never satisfies i >= 0 as a stopping condition. As a result, i keeps increasing and never terminates, causing an infinite loop.

**FIX:**

public class WrongInitializationForLoop {

public static void main(String[] args) {

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

System.out.println(i);

} }

}

**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?**

System.out.println(i) is conside inside the loop bit System.out.println("Done") is not inside the loop hence it print only once if we want it print 5 times then add {} in for loop .

**FIX:**

public class MisplacedForLoopBody {

public static void main(String[] args) {

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

System.out.println(i);

System.out.println("Done");

}

}}

**Snippet 7:**

public class UninitializedWhileLoop {

public static void main(String[] args) {

int count ;

while (count < 10) {

System.out.println(count);

count++;

}

}}

**Error to investigate:** count is not initialize that is why loop can not work

**FIX:**

public class UninitializedWhileLoop {

public static void main(String[] args) {

int count =5 ;

while (count < 10) {

System.out.println(count);

count++;

}

}}

**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 the loop print unexpected results or run infinitely? How should the loop update expression be corrected?**

statement num-- decrements num to 0. The condition while (num > 0) becomes false, so the loop exits immediately.

**FIX:**

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);

}

}

}

**Why does the loop print unexpected results or run infinitely? How should the loop update expression be corrected**

No, this loop does not run infinitely. it prints even numbers up to 4 and then exits.

**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?**

Error is our loop condition we use Assignment (=) Instead of Comparison (==)

**FIX:**

class IncorrectWhileLoopControl {

public static void main(String[] args) {

int num = 10;

while (num == 10) {

System.out.println(num);

num--;

}

}

}

**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?**

The loop skips odd numbers because i increases by 2 in each iteration. The loop does not print 1, 3.

**FIX:**

public class IncorrectLoopUpdate {

public static void main(String[] args) {

int i = 0;

while (i < 5) {

System.out.println(i);

i++;

}

}

}

**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**

The variable x is declared inside the for loop block.In Java, variables declared inside a block {} are only accessible within that block. x is a local variable and is created and destroyed on each iteration of the loop. Outside the loop, x does not exist, so trying to access it in System.out.println(x); causes a compilation error.

**FIX:**

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); // Error: 'x' is not accessible here

}

}