Credit Name: Chapter 13

Assignment Name: ReverseList

Describe the errors you've encountered while working on this assignment. What caused the error and how do you overcome the error?

1. Infinite Input Loop

Error: The program continues to prompt the user for input even after entering 999.

Cause: The termination condition in the input loop was not properly implemented, and the loop does not exit when 999 is entered.

Error Code:

```
while (stack.size() < 10) {
    num = input.nextInt();
    if (num == 999) {
    }
    stack.push(num);
}
```

Fix: Added a break statement when 999 is entered to exit the loop.

```
while (stack.size() < 10) {
    num = input.nextInt();
    if (num == 999) {
        break;
    }
    stack.push(num);
}
```

2. Calling pop() on an Empty Stack

Error: EmptyStackException occurs when attempting to reverse an empty list.

Cause: The program does not check if the stack is empty before calling pop().

Error Code:

```
while (true) {
    System.out.print(stack.pop() + " ");
}
```

Fix: Added a condition to check if the stack is not empty before calling pop().

```
while (!stack.isEmpty()) {
    System.out.print(stack.pop() + " ");
}
```

3. Missing Input Validation

Error: The program throws an InputMismatchException when the user enters non-integer input.

Cause: The program assumes all input is valid and does not handle invalid entries.

Error Code:

```
int num = input.nextInt();
```

Fix: Used a try-catch block to handle invalid input and prompt the user again.

```
int num;
while (true) {
    try {
        num = input.nextInt();
        break;
    } catch (Exception e) {
        System.out.println("Invalid input. Please enter an integer.");
        input.next();
    }
}
```

4. Forgetting to Close the Scanner

Error: Resource leak warning appears when the Scanner object is not closed.

Cause: The program does not close the Scanner at the end, leading to potential resource leaks.

Error Code:

```
Scanner input = new Scanner(System.in);
```

Fix: Added input.close() at the end of the program to properly release resources.

```
input.close();
```

5. Stack Overflow Due to Too Many Items

Error: The stack overflows if more than 10 items are entered.

Cause: No limit was set on the number of items that could be added to the stack.

Error Code:

```
while (true) {
    stack.push(num);
}
```

Fix: Added a condition to ensure the stack size does not exceed 10.

```
while (stack.size() < 10) {
    num = input.nextInt();
    if (num == 999) {
        break;
    }
    stack.push(num);
}
```