Scenario-based problem statements to practice the use of `break` and `continue` statements in Java:

---

1. Early Exit from a Loop

Write a program to search for a specific number in an array. Use a `break` statement to exit the loop immediately once the number is found. Print the index of the number or a message if it's not found.

---

2. Skipping Specific Iterations

Create a program to print all numbers from 1 to 50, but skip numbers divisible by 7 using the `continue` statement.

---

3. Nested Loops and Break

Write a program to find the first pair of numbers `(i, j)` such that `i + j = 50`, where `i` ranges from 1 to 50 and `j` ranges from 1 to 50. Use `break` to exit both loops once the pair is found.

---

4. Skipping Invalid Input

Simulate a user input scenario where the user is asked to enter 10 positive integers. Use `continue` to skip processing if the user enters a negative number or zero.

---

5. Prime Number Finder

Write a program to print all prime numbers between 1 and 100. Use `continue` to skip non-prime numbers during the iteration.

---

6. First Repeated Character in a String

Write a program to find the first repeated character in a string. Use a `break` statement to exit the loop once the repeated character is found.

---

7. Skipping Specific Conditions in Nested Loops

Create a multiplication table for numbers 1 to 10, but skip printing the table for multiples of 5 using the `continue` statement in a nested loop.

---

8. Early Termination in a Game Simulation

Simulate a simple game where the user guesses a number between 1 and 100. Allow the user to make up to 10 guesses, but exit the loop early using `break` if the correct number is guessed.

---

9. Skipping Certain Characters in a String

Write a program to count the number of vowels in a string. Use `continue` to skip consonants and non-alphabetic characters during the iteration.

---

10. Breaking Out of Infinite Loops

Simulate a menu-driven program with an infinite loop where the user can choose options like "Add", "Delete", "Update", and "Exit". Use `break` to exit the loop when the user selects the "Exit" option.