Import java.util.\*;

Import java.util.function.Predicate;

Public class GenericCollectionCounterWithValidation {

// Generic method to count elements based on a condition (predicate)

Public static &lt;T&gt; int countByCondition(Collection&lt;T&gt; collection, Predicate&lt;T&gt;

predicate) {

Int count = 0;

For (T element : collection) {

If (predicate.test(element)) {

Count++;

}

}

Return count;

}

// Method to check if a number is even, with validation for positive numbers

Public static boolean isEven(Integer num) {

validateNumber(num);

return num % 2 == 0;

}

// Method to check if a number is odd, with validation for positive numbers

Public static boolean isOdd(Integer num) {

validateNumber(num);

return num % 2 != 0;

}

// Method to check if a number is prime, with validation for positive numbers

Public static boolean isPrime(Integer num) {

validateNumber(num);

if (num &lt;= 1) return false;

for (int I = 2; I &lt;= Math.sqrt(num); i++) {

if (num % I == 0) {

return false;

}

}

Return true;

}

// Method to check if a string is a palindrome, with validation for non-null and non-

empty strings

Public static boolean isPalindrome(String str) {

validateString(str);

String reverse = new StringBuilder(str).reverse().toString();

Return str.equalsIgnoreCase(reverse);

}

// Validate that the number is not negative

Private static void validateNumber(Integer num) {

If (num == null) {

Throw new IllegalArgumentException(“Number cannot be null!”);

}

If (num &lt; 0) {

Throw new IllegalArgumentException(“Number cannot be negative!”);

}

}

// Validate that the string is not null or empty

Private static void validateString(String str) {

If (str == null) {

Throw new IllegalArgumentException(“String cannot be null!”);

}

If (str.trim().isEmpty()) {

Throw new IllegalArgumentException(“String cannot be empty!”);

}

}

Public static void main(String[] args) {

// Create a list of numbers with a mix of positive and negative numbers

List&lt;Integer&gt; numbers = Arrays.asList(11, 2, 3, 5, 4, 6, 7, 13, 22, -10, 101, 121);

// Create a list of strings with some invalid (empty or null) strings

List&lt;String&gt; words = Arrays.asList(“racecar”, “”, “madam”, “level”, “world”, null,

“noon”);

Try {

// Count even numbers with validation

Int evenCount = countByCondition(numbers,

GenericCollectionCounterWithValidation::isEven);

System.out.println(“Number of even numbers: “ + evenCount);

} catch (IllegalArgumentException e) {

System.out.println(“Error: “ + e.getMessage());

}

Try {

// Count odd numbers with validation

Int oddCount = countByCondition(numbers,

GenericCollectionCounterWithValidation::isOdd);

System.out.println(“Number of odd numbers: “ + oddCount);

} catch (IllegalArgumentException e) {

System.out.println(“Error: “ + e.getMessage());

}

Try {

// Count prime numbers with validation

Int primeCount = countByCondition(numbers,

GenericCollectionCounterWithValidation::isPrime);

System.out.println(“Number of prime numbers: “ + primeCount);

} catch (IllegalArgumentException e) {

System.out.println(“Error: “ + e.getMessage());

}

Try {

// Count palindromes with validation

Int palindromeCount = countByCondition(words,

GenericCollectionCounterWithValidation::isPalindrome);

System.out.println(“Number of palindromes: “ + palindromeCount);

} catch (IllegalArgumentException e) {

System.out.println(“Error: “ + e.getMessage());

}

}

}