# Exercises: Unit Testing Exceptions

Submit your solutions here: [https://judge.softuni.org/Contests/4493](https://judge.softuni.org/Contests/4493/Exceptions-and-Error-Handling-Unit-Testing-Exercise)

# Unit Test: Reverse – Argument Null Exception

Look at the **provided skeleton** and examine the Exceptions.cs class that you will test:

A screenshot of a computer

Description automatically generated

The class has **multiple methods each showing the use of different exceptions**.

The first method, ArgumentNullReverse(), takes in a **string** **and reverses it**. If the string is **null** an ArgumentNullException is thrown:

A computer code on a white background

Description automatically generated

Then, look at the tests inside the ExceptionTests.cs class:

A screenshot of a computer

Description automatically generated

Notice the use of a **setup** **method**, so each test has a brand new **exception instance** to use.

A screen shot of a computer program

Description automatically generated

The two tests are **partially** **finished**, and your task is to finish them. The tests should run when you're finished:

A close up of words

Description automatically generated

# Unit Test: Calculate Discount – Argument Exception

The ArgumentCalculateDiscount() **method** takes in a **total price decimal**, and **discount decimal**. It calculates and returns the discounted price. If the discount is **lower than 0 or higher than 100** an ArgumentException is thrown:

A screen shot of a computer code

Description automatically generated

Now, look at the tests:

A screen shot of a computer

Description automatically generated

You are given **two** **partially** **finished** test, the rest are **empty,** and your task is to finish them. The tests should run when you're finished:

A screenshot of a computer screen

Description automatically generated

# Unit Test: Get Element – Index out of Range Exception

The IndexOutOfRangeGetElement() **method** takes in an **array of integers**, and an **index**. It **retrieves** the element from the **array** at the **given index**. If the index is **lower than 0 or higher / equal** to the **length** an IndexOutOfRangeException is thrown:

A screen shot of a computer code

Description automatically generated

Now, look at the tests:

A screenshot of a computer program

Description automatically generated

You are given **one** **partially** **finished** test, the rest are **empty,** and your task is to finish them. The tests should run when you're finished:

A text on a white background

Description automatically generated

# Unit Test: Perform Secure Operation – Invalid Operation Exception

The InvalidOperationPerformSecureOperation() **method** takes in a **Boolean indicating if the user is logged in**. If the user is not logged in an InvalidOperationException is thrown:

A computer screen shot of text

Description automatically generated

Now, look at the tests:

A close-up of a computer screen

Description automatically generated

You are given **two** **empty** tests, and your task is to finish them. The tests should run when you're finished:

A close up of words

Description automatically generated

# Unit Test: Parse Int - Format Exception

The FormatExceptionParseInt() **method** takes in a **string as input** and tries to **parse** it into an **integer**. If the string is not a valid number a FormatException is thrown:

A computer screen shot of a program

Description automatically generated

Now, look at the tests:

A computer code with black text

Description automatically generated with medium confidence

You are given **two** **empty** tests, and your task is to finish them. The tests should run when you're finished:

A close-up of a computer

Description automatically generated

# Unit Test: Find Value by Key – Key Not Found Exception

The KeyNotFoundFindValueByKey() **method** takes in a **dictionary**, and a **string representing a key from the dictionary**. If the key does not exist in the dictionary a KeyNotFoundException is thrown:

A computer screen shot of text

Description automatically generated

Now, look at the tests:

A close-up of a computer screen

Description automatically generated

You are given **two** **empty** tests, and your task is to finish them. The tests should run when you're finished:



# Unit Test: Add Numbers – Overflow Exception

The OverflowAddNumbers() **method** takes in **two numbers to be summed together**. If summing the numbers **overflows the integer type** aOverflowException is thrown:

A computer screen shot of a computer code

Description automatically generated

Now, look at the tests:

A screenshot of a computer code

Description automatically generated

You are given **three** **empty** tests, and your task is to finish them. The tests should run when you're finished:

A close-up of words

Description automatically generated

# Unit Test: Divide Numbers – Divide by Zero Exception

The DivideByZeroDivideNumbers() **method** takes in a **two numbers to be divided**. If the **divisor is 0** a DivideByZeroException is thrown:

A computer code with text

Description automatically generated with medium confidence

Now, look at the tests:

A white background with black text

Description automatically generated

You are given **two** **empty** tests, and your task is to finish them. The tests should run when you're finished:

A black text on a white background

Description automatically generated

# Unit Test: Sum Collection Elements

The SumCollectionElements() **method** takes in an **array of integers**, and an **index**. If the collection is **null** an ArgumentNullException is thrown, or if the **index is out of bounds** an IndexOutOfRangeException is thrown:

A screenshot of a computer code

Description automatically generated

Now, look at the tests:

A screenshot of a computer code

Description automatically generated

You are given **three** **empty** tests, and your task is to finish them. The tests should run when you're finished:

A screenshot of a computer

Description automatically generated

# Unit Test: Get Element as Number

The GetElementAsNumber() **method** takes in a **dictionary**, and a **string representing a key from the dictionary**. If the key does not exist in the dictionary a KeyNotFoundException is thrown, if the value **cannot be converted to integer** a FormatException is thrown:

A screen shot of a computer code

Description automatically generated

Now, look at the tests:

A screenshot of a computer code

Description automatically generated

You are given **two** **empty** tests, and your task is to finish them. The tests should run when you're finished:

A close up of words

Description automatically generated

At the end make sure all your tests run:

A screenshot of a computer

Description automatically generated