4.1 Exception handling report

NullReferenceException

This exception is thrown when you try to use something as reference, but it is not yet initialized.

This exception should be thrown by the Runtime system not by me or any programmer, as it reflects the programmer’s error rather than the user-side error. This exception should be aware and avoided by programmer.

As this exception reflects the programmer’s error, it should include the variable that is being null, the message should be like this: “There is a null value that is being used as reference!”. No parameter because the parameter that have this exception is being null and the debug console has provided the line that created this exception.

This exception can be easily caught by programmer and therefore can be handled.

This exception should be dealt by programmer as it occurs by programmer’s logic and should be fix by programmer.

Programmer should avoid this exception by finding the reason why the parameter is being null and fix it. Usually the parameter is set to null before any reference, just set it to something else that suitable to the working project.

IndexOutOfRangeException

This exception is thrown when you are trying to access an item in an array using an invalid index, usually the index is out of array’s range. For example, the array has length of 4 (0 to 3) but you are trying to access the 6th position.

This exception is sometime caused by user and sometime by programmer, therefore the programmer should throw this exception.

The message that comes along with the exception is: $“You are trying to access the position {postion} of the array that have the length of {array.Length}” The parameters that included are the index position that is being access to and the length of the array.

This exception can be generally caught as it can be caused by both user and programmer; and can be handle by programmer.

If this exception occurs, it should display as message because it is usually vital to both user and programmer.

This exception should be avoided by user by using a loop to set the limit to user access to the array’s range. If the exception is caused by the programmer, then it should be related to the logic of the program. Such as the array has the length of 4, then the maximum index should be 3. If programmer use a loop to work with this array, the condition should be “i<array.Length” or “i<=array.Length-1”, not “i<=array.Length”.

StackOverflowException

This exception is thrown when there is an infinite loop or an infinite recursion

The system is in charge of throwing this exception to programmer as it usually causes of programmer’s error.

If this exception occurs, it should be displayed as message as this exception is vital to the programmer’s program usability. The message should be the exception’s message itself. As the system throws this exception, not programmer.

This exception can not be catch since .NET Framework 2.0, therefore the programmers must handle it themselves, by debugging for any unwanted infinite loop or infinite recursion.

OutOfMemoryException

This exception is thrown when the memory of the device is not enough to run the program

The system is in charge of throwing this exception because it is not programmer’s fault.

If this exception occurs, the system should display the message, without any parameter because the programmer can hardly handle it.

This exception can not easily get caught because devices these days usually has enough memory to handle programs.

This exception is not the kind that you want to avoid because it’s related to hardware, the only way to fix it would be optimize the program to use less resources or run the program with a device that have enough memory.

InvalidCastException

This exception is thrown when the conversion of 2 types of data is not supported, for example: conversion from string to date and time.

The runtime system is in charge of throwing this exception because it reflects programmer’s fault.

If this exception is required to be thrown