Object Oriented Development: Exceptions Report
Lyndon Purcell: 221368813

a) NullReferenceException

– **What is a possible situation that leads to the exception:** An attempt to access an attribute or operation from an object is made, but the reference is set to null.

– **Who is in charge of throwing the exception:** The runtime. This exception is usually due to a programming error (such as failing to check if a reference isn't set to null before referring to it). As such, the runtime must throw the exception to highlight the concern.

– **If you need to throw the exception, what parameters would you include in the message:** Which object, or attribute of an object, was set to null.

– **Can the exception be generally caught (and therefore handled):** Yes, if caught, you can then execute code that instantiates the object.

– **If the exception occurs, should you generally catch this exception type or is it better to pass such exception to the user (caller):** Catch it and execute code that instantiates the object if it is being referenced directly. Possibly if the reference is being used as input for another method, then the message should possibly be passed on.

– **Is the exception a case when you want to avoid this exception to occur in your application in general? If so, what would be your actions as a programmer to avoid it:** Yes. You can avoid it by instantiating objects, declaring the number of elements in an array before initialising it, and checking methods return value isn't null.

b) IndexOutOfRangeException

– **What is a possible situation that leads to the exception:** The index value for that particular array does not exist.

– **Who is in charge of throwing the exception:** The runtime. This exception is usually due to a programming error (such as trying to iterate up to the number of elements in an array, as opposed to one less than, due to zero-indexing). As such, the runtime must throw the exception to highlight the concern.

– **If you need to throw the exception, what parameters would you include in the message:** Which array you're trying to incorrectly index into.

– **Can the exception be generally caught (and therefore handled):** In most cases it should not be handled and the cause of the error should be determined and fixed.

– **If the exception occurs, should you generally catch this exception type or is it better to**

pass such exception to the user (caller): Pass to caller. They should be notified that the underlying logic is unreliable.

– **Is the exception a case when you want to avoid this exception to occur in your application in general? If so, what would be your actions as a programmer to avoid it:** Ensure loops are used correctly when iterating over the elements of an array or list, or verify returned values are within the appropriate range first if using them to index into elements.

c) StackOverflowException

– **What is a possible situation that leads to the exception:** The call stack has grown too large and cannot take on anymore tasks.

– **Who is in charge of throwing the exception:** The runtime. This exception is usually due to a programming error (such an infinite loop or unbounded recursion). As such, the runtime must throw the exception to highlight the concern.

– **If you need to throw the exception, what parameters would you include in the message:** The series of function call path that led to the overflow.

– **Can the exception be generally caught (and therefore handled):** It can be caught, and you should program in a way to catch it if it occurs. When caught, the handler will terminate the offending process.

– **If the exception occurs, should you generally catch this exception type or is it better to pass such exception to the user (caller):** Catch and handle. The stack is overflowing, by definition, and that issue should be addressed immediately.

– **Is the exception a case when you want to avoid this exception to occur in your application in general? If so, what would be your actions as a programmer to avoid it:** Yes, you would want to avoid this issue. You can help to prevent this from occurring by ensuring any recursive functions have established base-cases.

d) OutOfMemoryException

– **What is a possible situation that leads to the exception:** Insufficient memory is available for allocation, either totally or contiguously.

– **Who is in charge of throwing the exception:** The runtime.

– **If you need to throw the exception, what parameters would you include in the message:** What process specifically had insufficient memory to proceed.

– **Can the exception be generally caught (and therefore handled):** It can be, but the

documentation suggests that if it is handled, you still opt to have the application log the exception and 'FailFast.'

– **If the exception occurs, should you generally catch this exception type or is it better to pass such exception to the user (caller):** It should be caught and dealt with immediately (such as logging and terminating the app). A lack of memory resources is a serious vulnerability, and this state should not be operated in for a prolonged period.

– **Is the exception a case when you want to avoid this exception to occur in your application in general? If so, what would be your actions as a programmer to avoid it:** Yes, it should be avoided at all costs. There are a lot of things you can do to avoid this, being cautious with memory management when using StringBuilders or other large objects in memory would be a good start.

e) DivideByZeroException

– **What is a possible situation that leads to the exception:** When an attempt to a divide a decimal or integer value by zero is made.

– **Who is in charge of throwing the exception:** The runtime.

– **If you need to throw the exception, what parameters would you include in the message:** What is the value that is being attempted to be divided by zero.

f) ArgumentNullException

– **What is a possible situation that leads to the exception:** When an object, whose reference is set to null, is passed to a method that does not accept it as an argument.

– **Who is in charge of throwing the exception:** The developer.

– **If you need to throw the exception, what parameters would you include in the message:** Which method had a null argument passed to it, as well as the arguments it received.

– **Can the exception be generally caught (and therefore handled):** It is best to remove the cause of this error, rather than catching it. Returned values or object attributes should be verified before using them as arguments.

– **If the exception occurs, should you generally catch this exception type or is it better to pass such exception to the user (caller):** Pass to caller.

– **Is the exception a case when you want to avoid this exception to occur in your application in general? If so, what would be your actions as a programmer to avoid it:** Yes. You can avoid it by instantiating objects and ensuring passed values are not null before using them as arguments.

g) ArgumentOutOfRangeException

– **What is a possible situation that leads to the exception:** When an object, whose reference is *not* set to null, is passed to a method that does not accept that value as an argument.

– **Who is in charge of throwing the exception:** The developer.

– **If you need to throw the exception, what parameters would you include in the message:** Which method had an out of range argument passed to it, as well as the arguments it received.

– **Can the exception be generally caught (and therefore handled):** It is best to remove the cause of this error, rather than catching it. Passed values or user input should be verified as appropriate before using them as arguments.

– **If the exception occurs, should you generally catch this exception type or is it better to pass such exception to the user (caller):** Pass to caller.

– **Is the exception a case when you want to avoid this exception to occur in your application in general? If so, what would be your actions as a programmer to avoid it:** Yes. You can avoid it by ensuring returned values or user input are appropriate before using them as arguments.

h) FormatException

– **What is a possible situation that leads to the exception:** The format of an argument is invalid, such as in a format string or passing data to the incorrect type.

– **Who is in charge of throwing the exception:** The runtime.

– **If you need to throw the exception, what parameters would you include in the message:** What argument, of the incorrect format, was passed and to what.

– **Can the exception be generally caught (and therefore handled):** It can be caught and handled in some cases, such as by prompting the user to enter input of the correct type.

– **If the exception occurs, should you generally catch this exception type or is it better to pass such exception to the user (caller):** Catch and handle it.

– **Is the exception a case when you want to avoid this exception to occur in your**

application in general? If so, what would be your actions as a programmer to avoid it: Utilising try-catch-finally blocks when parsing and converting user input.

i) SystemException

– What is a possible situation that leads to the exception: This exception, specifically, shouldn't be used. This is the namespace class of all other exceptions. The derived classes provide additional information that this abstract class does not.