Exceptions

Errors in programs

Errors could be different:

- Syntactic

```
string s = 12;
```

- Run-Time

```
int i = int.Parse(Console.ReadLine());
```

- Logical
  - the program works, but it gives something wrong

Exceptions and Exception Handling

- The C# language's exception handling features help you deal with any unexpected or exceptional situations that occur when a program is running.
- Exception handling uses the **try**, **catch**, and **finally** keywords to try actions that may not succeed, to handle failures when you decide that it's reasonable to do so, and to clean up resources afterward.
- Exceptions can be generated by the common language runtime (CLR), by .NET or third-party libraries, or by application code.
- Exceptions are created by using the **throw** keyword.
- In many cases, an exception may be thrown not by a method that your code has called directly, but by another method further down in the call stack.
  - When an exception is thrown, the CLR will unwind the stack, looking for a method with a catch block for the specific exception type, and it will execute the first such catch block that it finds.
  - If it finds no appropriate catch block anywhere in the call stack, it will terminate the process and display a message to the user.

Exception Handling

```
try
{
    int i = int.Parse(Console.ReadLine());
}
catch // Exception Handling
{
    Console.WriteLine(“It is not a number...");
}
finally // this block is optional
{
    Console.WriteLine(“Finally!"); // but always works
}
```

catch () {…}

Separation of exception types:

```
string s = Console.ReadLine();
try
{
    int i = int.Parse(s);
}
catch(FormatException ex)
{
    Console.WriteLine($“Exception {ex.Message}");
}
catch
{ … }
```

Exception filters

```
string s = Console.ReadLine();
try
{
    int i = int.Parse(s);
}
catch(FormatException ex) when (s == "Hello world!")
{
    Console.WriteLine("Hello to you too!");
}
catch(FormatException ex)
{
    Console.WriteLine($"Exception {ex.Message}");
}
```

Exception class

- Base class for all exceptions

```
catch (Exception ex)
{
    Console.WriteLine($“Exception message: {ex.Message}");
    Console.WriteLine($“Method: {ex.TargetSite}");
    Console.WriteLine($“Stack trace: {ex.StackTrace}");
}
```

- You can inherit and create your own exception type

```
class MyException : Exception { … }
```

throw

```
Console.WriteLine(“Enter your name: ");
string name = Console.ReadLine();
if (name.Length < 2)
{
    throw new MyException("Name must be at least 2 characters long");
}
```

- This exception can be catched as usual!

Exceptions Overwiew

- Once an exception occurs in the try block, the flow of control jumps to the first associated exception handler that is present anywhere in the call stack. In C#, the catch keyword is used to define an exception handler.
- If no exception handler for a given exception is present, the program stops executing with an error message.
- Don't catch an exception unless you can handle it and leave the application in a known state. If you catch `System.Exception`, **rethrow** it using the throw keyword at the end of the catch block.
- If a catch block defines an exception variable, you can use it to obtain more information about the type of exception that occurred.
- Exceptions can be explicitly generated by a program by using the throw keyword.
- Exception objects contain detailed information about the error, such as the state of the call stack and a text description of the error.
- Code in a finally block is executed regardless of if an exception is thrown. Use a finally block to release resources, for example to close any streams or files that were opened in the try block.

Exceptions. Summary

- An exception is one in which a program does not run as expected.
- Exceptions are very helpful in locating the error in the future, they allow you to get rid of the if ... else sheet, checks for null, false, etc.
- But it’s worth using them carefully, after all, it’s worth separating the exceptional situation and the completely normal, standard execution branch.

- **Do not hang application logic on exceptions!**