Exception

HCL

Types of Errors

- Compile time error
  - Detected at the compile time by the compiler
- Runtime errors
  - Detected by the runtime system
  - Who handles them?
  - How can your program handle them?

HCL

Wonder?

```
class Div{
public void Main(){
int k=10;
k=k/0;
System.Console.WriteLine("hello");
}
}
```

What error do you expect to get?

HCL

Runtime error

```
class Div{

public static void Main(){

int k=10, j=0;

k=k/j;

System.Console.WriteLine("hello");
```

```
Unhandled Exception:
System.DivideByZeroException: Attempted to
divide by zero.
at Div.Main()
```

Runtime error automatically handled by the runtime system

HCL

Exception Handling

- Handling exception that occur at runtime in our application is exception handling.
- Handler

```
try{
// code that may throw exception
}catch(Exception e){
// handler
}
```

HCL

Example

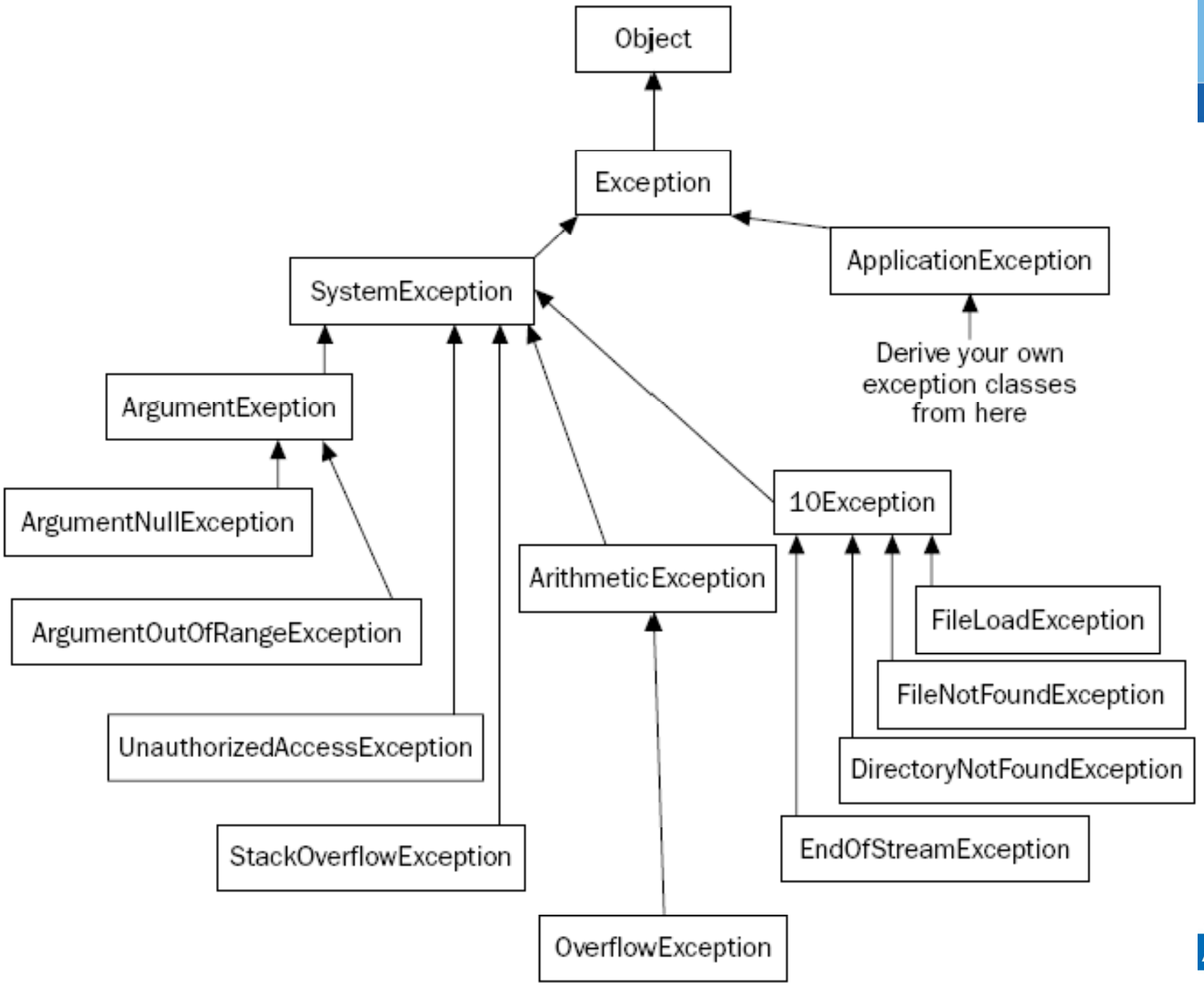
```
using System;
class Div{
public static void Main(){
int k=10, j=0;
try{
k=k/j;
Console.WriteLine("hello");
}catch(Exception e){
Console.WriteLine("you are attempting
  to divide by 0");
}}}
```

HCL

try-catch blocks

- A try block is a block where error are expected to occur.
- The errors are thrown in the form of Exception object in C#.
- A catch block will have the code to handle the error.
- A try block can have one or more catch block where each catch block can handle different types of exceptions.

HCL


Object
Exception
SystemException
ApplicationException
Derive your own exception classes from here
ArgumentExeption
ArgumentNullException
ArgumentOutOfRangeException
UnauthorizedAccessException
StackOverflowException
ArithmeticException
OverflowException
10Exception
FileLoadException
FileNotFoundException
DirectoryNotFoundException
EndOfStreamException

Exception class members

- **`Message`**
  - Gets a message that describes the current exception.
- **`Source`**
  - Gets or sets the name of the application or the object that causes the error.
- **`StackTrace`**
  - Gets a string representation of the frames on the call stack at the time the current exception was thrown
- **`TargetSite`**
  - Gets the method that throws the current exception.

HCL

Multiple Exception

```
using System;
class Div{
public static void Main(string[] s){
try{
int j=10;
j=j/s.Length;
int k=Int32.Parse(s[0]);
j=j/k;
Console.WriteLine("j"+j);
}catch(DivideByZeroException d){
Console.WriteLine("divide by zero "+d);
}
```

HCL

```
catch(FormatException d){
Console.WriteLine(" not a number "+d);}

catch(Exception d){
Console.WriteLine(" general error
"+d);}
catch{
Console.WriteLine("error due to code
outside the system");}
}}
```

for objects which thrown are not of Exception type → In cases where you invoke a method written in a language that can throws object others than the Exception type.

The catch handler sequencing is important. The subclasses objects must be caught before the super class.

HCL

finally

- try block can have finally block as well apart from the catch block.
- finally block will execute whether or not an exception occurs.
- It is provided so that clean up code could be written in all cases whether an error occurs or not like closing of a file, database connection etc.
- In C#, a try block must be followed by either a catch or finally block .

HCL

```
using System;
class Div{
public static void Main(string[] s){
try{
int j=10;
j=j/s.Length;
int k=Int32.Parse(s[0]);
j=j/k;
Console.WriteLine("j"+j);
}catch(DivideByZeroException d){
Console.WriteLine("divide by zero "+d);}
catch(FormatException d){
Console.WriteLine(" not a number "+d);}
catch(Exception d){
Console.WriteLine(" general error "+d);}
finally { Console.WriteLine("Finally Block"); }
 Console.WriteLine("Bye");
}}
```

HCL

What do you notice?

- Executing
  - **F:\..\Slide Examples\6. Exception>Arg**

```
divide by zero
System.DivideByZeroException:
Attempted to divide by zero.
at Div.Main(String[] s)
Finally Block
Bye
```

  - **F:\Materials\My version\Dot Net Material\C# Material\Slide Examples\6. Exception>Arg 78**

```
j0
Finally Block
Bye
```

Note that both Finally Block and Bye are printed in both the cases

HCL

Throwing an exception

- it is possible to throw an exception explicitly from code.
- **`throw excepobject;`**
- ***`Or`***
- **`throw new ArgumentException("wrong arguments");`**

HCL

Types of exception

- Standard Exception
  - Exception thrown by the CLR
  - CLR throws objects of type **`SystemException`**
- Application Exception
  - thrown by a user program rather than the runtime.
  - Inherits from **`ApplicationException`**

HCL

Application exception

```
using System;
class AgeException :
  ApplicationException{
string s;
 public AgeException(string str) {
   s=str +" is invalid age. Should be
   between 1 and 100";
  }
public override string ToString(){
return s;
}
}
```

HCL

```
class Test{
 public static void Main() {
   try
   {
    Console.WriteLine("enter age");
    string s=Console.ReadLine();
    int num = Int32.Parse(s);
      if(num<1 || num>100)
      throw new AgeException(s);
   }
   catch(AgeException  e) {
    Console.WriteLine (e);
   }
   }
}
```

What will happen if enter a alphabets instead of number for age?

HCL