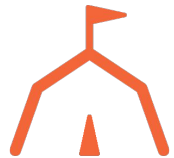
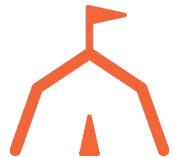
Exceptions And Error Handling



Types Of Errors

- Syntax errors violate the rules for how C# statements must be written.
- Runtime errors don’t violate syntax rules, but cause exceptions to be thrown which stop the execution of the application.
- Logic errors don’t cause syntax or runtime errors, but produce incorrect results.



Common Syntax Errors

- Misspelling keywords
- Forgetting to declare a data type for a variable
- Forgetting an opening or closing parenthesis, bracket, quotation mark, or comment character
- Forgetting to code a semicolon at the end of a statement

Handling Exceptions

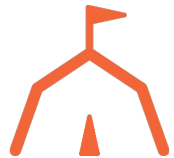
How Exceptions Work

An exception is thrown when the application can’t perform an operation. An exception is an object created from one of several different Exception classes.

Handling Exceptions

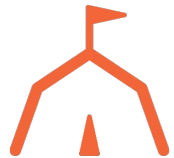
How Exceptions Work

An application is said to catch any thrown exceptions and handle them, which may involve simply informing the user they need to provide valid input or more complicated action.



Handling Exceptions

In order to catch an exception, we use blocks of statements, called the try statement (which contains code that may throw an exception) and exception handler (which details the appropriate response to an exception).



Syntax

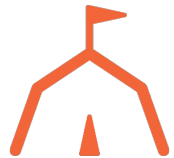
```
try
{ statements
}
catch (ExceptionClass exceptionName)
{
  statements
}
```

Example

```
double subtotal = 0.0;
    try{
    Console.WriteLine("Enter subtotal:");
    subtotal = Double.Parse(Console.ReadLine());
    }
    catch (FormatException e)
    {
        Console.WriteLine("Error! Invalid number.");
    }
```

Having Multiple Catches

```
try{ // statements causing exception
}
catch( ExceptionName e1 )
{   // error handling code
}
catch( ExceptionName e2 )
{   // error handling code
}
catch( ExceptionName e3 )
{   // error handling code
}
```



Throw Statement

Gives the programmer the ability to generate an exception

```
throw new Exception("Interst rate must be > 0.");
```