

ShapeParser

Generated by Doxygen 1.9.7

1 Đ ÁN CUI KÌ MÔN LP TRÌNH HNG ĐI TNG	1
2 shape	3
3 Namespace Index	5
3.1 Namespace List	5
4 Hierarchical Index	7
4.1 Class Hierarchy	7
5 Class Index	9
5.1 Class List	9
6 File Index	13
6.1 File List	13
7 Namespace Documentation	17
7.1 myCircle Namespace Reference	17
7.2 myEllipse Namespace Reference	17
7.3 myIsoscelesTrapezoid Namespace Reference	17
7.4 myParallelogram Namespace Reference	17
7.5 myRectangle Namespace Reference	18
7.6 myRhombus Namespace Reference	18
7.7 mySquare Namespace Reference	18
7.8 myTriangle Namespace Reference	18
8 Class Documentation	19
8.1 myCircle::Circle Class Reference	19
8.1.1 Detailed Description	20
8.1.2 Constructor & Destructor Documentation	20
8.1.2.1 Circle()	20
8.1.3 Member Function Documentation	20
8.1.3.1 area()	20
8.1.3.2 perimeter()	20
8.1.3.3 radius()	21
8.1.3.4 toString()	21
8.1.4 Member Data Documentation	21
8.1.4.1 _radius	21
8.2 CircleParser Class Reference	21
8.2.1 Detailed Description	22
8.2.2 Constructor & Destructor Documentation	22
8.2.2.1 CircleParser() [1/2]	22
8.2.2.2 ~CircleParser()	23
8.2.2.3 CircleParser() [2/2]	23
8.2.3 Member Function Documentation	23

8.2.3.1 getInstance()	23
8.2.3.2 operator=()	23
8.2.3.3 parse()	23
8.2.3.4 toString()	24
8.2.4 Member Data Documentation	24
8.2.4.1 _instance	24
8.3 CircleToStringConverter Class Reference	24
8.3.1 Detailed Description	25
8.3.2 Member Function Documentation	25
8.3.2.1 convert()	25
8.3.2.2 toString()	25
8.4 ConverterFactory Class Reference	26
8.4.1 Detailed Description	26
8.4.2 Member Function Documentation	26
8.4.2.1 registerWith()	26
8.4.2.2 select()	27
8.4.2.3 toString()	27
8.4.3 Member Data Documentation	27
8.4.3.1 _prototypes	27
8.5 myEllipse::Ellipse Class Reference	27
8.5.1 Detailed Description	28
8.5.2 Constructor & Destructor Documentation	28
8.5.2.1 Ellipse()	28
8.5.3 Member Function Documentation	29
8.5.3.1 area()	29
8.5.3.2 perimeter()	29
8.5.3.3 semi_major_axis()	29
8.5.3.4 semi_minor_axis()	29
8.5.3.5 toString()	30
8.5.4 Member Data Documentation	30
8.5.4.1 _semi_major_axis	30
8.5.4.2 _semi_minor_axis	30
8.6 EllipseParser Class Reference	30
8.6.1 Detailed Description	31
8.6.2 Constructor & Destructor Documentation	31
8.6.2.1 EllipseParser() [1/2]	31
8.6.2.2 ~EllipseParser()	32
8.6.2.3 EllipseParser() [2/2]	32
8.6.3 Member Function Documentation	32
8.6.3.1 getInstance()	32
8.6.3.2 operator=()	32
8.6.3.3 parse()	32

8.6.3.4 toString()	33
8.6.4 Member Data Documentation	33
8.6.4.1 _instance	33
8.7 EllipseToStringConverter Class Reference	33
8.7.1 Detailed Description	34
8.7.2 Member Function Documentation	34
8.7.2.1 convert()	34
8.7.2.2 toString()	34
8.8 IParser Class Reference	35
8.8.1 Detailed Description	35
8.8.2 Member Function Documentation	35
8.8.2.1 parse()	35
8.9 IShape Class Reference	36
8.9.1 Detailed Description	37
8.9.2 Member Function Documentation	37
8.9.2.1 area()	37
8.9.2.2 perimeter()	37
8.10 IShapeToStringConverter Class Reference	38
8.10.1 Detailed Description	38
8.10.2 Member Function Documentation	38
8.10.2.1 convert()	38
8.11 IShowDataBehavior Class Reference	39
8.11.1 Detailed Description	39
8.11.2 Member Function Documentation	39
8.11.2.1 showData()	39
8.12 IShowTableBehavior Class Reference	40
8.12.1 Detailed Description	40
8.12.2 Member Function Documentation	40
8.12.2.1 showTable()	40
8.13 myIsoscelesTrapezoid::IsoscelesTrapezoid Class Reference	41
8.13.1 Detailed Description	42
8.13.2 Constructor & Destructor Documentation	42
8.13.2.1 IsoscelesTrapezoid()	42
8.13.3 Member Function Documentation	42
8.13.3.1 area()	42
8.13.3.2 base()	43
8.13.3.3 height()	43
8.13.3.4 perimeter()	43
8.13.3.5 top()	43
8.13.3.6 toString()	44
8.13.4 Member Data Documentation	44
8.13.4.1 _base	44

8.13.4.2	_height	44
8.13.4.3	_top	44
8.14	IsoscelesTrapezoidParser Class Reference	44
8.14.1	Detailed Description	45
8.14.2	Constructor & Destructor Documentation	45
8.14.2.1	IsoscelesTrapezoidParser() [1/2]	45
8.14.2.2	~IsoscelesTrapezoidParser()	46
8.14.2.3	IsoscelesTrapezoidParser() [2/2]	46
8.14.3	Member Function Documentation	46
8.14.3.1	getInstance()	46
8.14.3.2	operator=()	46
8.14.3.3	parse()	46
8.14.3.4	toString()	47
8.14.4	Member Data Documentation	47
8.14.4.1	_instance	47
8.15	IsoscelesTrapezoidToStringConverter Class Reference	47
8.15.1	Detailed Description	48
8.15.2	Member Function Documentation	48
8.15.2.1	convert()	48
8.15.2.2	toString()	48
8.16	Object Class Reference	49
8.16.1	Detailed Description	49
8.16.2	Member Function Documentation	49
8.16.2.1	toString()	49
8.17	myParallelogram::Parallelogram Class Reference	50
8.17.1	Detailed Description	51
8.17.2	Constructor & Destructor Documentation	51
8.17.2.1	Parallelogram()	51
8.17.3	Member Function Documentation	51
8.17.3.1	area()	51
8.17.3.2	base()	52
8.17.3.3	height()	52
8.17.3.4	perimeter()	52
8.17.3.5	side()	52
8.17.3.6	toString()	53
8.17.4	Member Data Documentation	53
8.17.4.1	_base	53
8.17.4.2	_height	53
8.17.4.3	_side	53
8.18	ParallelogramParser Class Reference	53
8.18.1	Detailed Description	54
8.18.2	Constructor & Destructor Documentation	54

8.18.2.1 ParallelogramParser() [1/2]	54
8.18.2.2 ~ParallelogramParser()	55
8.18.2.3 ParallelogramParser() [2/2]	55
8.18.3 Member Function Documentation	55
8.18.3.1 getInstance()	55
8.18.3.2 operator=()	55
8.18.3.3 parse()	55
8.18.3.4 toString()	56
8.18.4 Member Data Documentation	56
8.18.4.1 _instance	56
8.19 ParallelogramToStringConverter Class Reference	56
8.19.1 Detailed Description	57
8.19.2 Member Function Documentation	57
8.19.2.1 convert()	57
8.19.2.2 toString()	57
8.20 ParserFactory Class Reference	58
8.20.1 Detailed Description	58
8.20.2 Member Function Documentation	58
8.20.2.1 registerWith()	58
8.20.2.2 select()	59
8.20.2.3 toString()	59
8.20.3 Member Data Documentation	59
8.20.3.1 _prototypes	59
8.21 myRectangle::Rectangle Class Reference	59
8.21.1 Detailed Description	60
8.21.2 Constructor & Destructor Documentation	60
8.21.2.1 Rectangle()	60
8.21.3 Member Function Documentation	61
8.21.3.1 area()	61
8.21.3.2 height()	61
8.21.3.3 perimeter()	61
8.21.3.4 toString()	61
8.21.3.5 width()	62
8.21.4 Member Data Documentation	62
8.21.4.1 _height	62
8.21.4.2 _width	62
8.22 RectangleParser Class Reference	62
8.22.1 Detailed Description	63
8.22.2 Constructor & Destructor Documentation	63
8.22.2.1 RectangleParser() [1/2]	63
8.22.2.2 ~RectangleParser()	64
8.22.2.3 RectangleParser() [2/2]	64

8.22.3 Member Function Documentation	64
8.22.3.1 getInstance()	64
8.22.3.2 operator=()	64
8.22.3.3 parse()	64
8.22.3.4 toString()	65
8.22.4 Member Data Documentation	65
8.22.4.1 _instance	65
8.23 RectangleToStringConverter Class Reference	65
8.23.1 Detailed Description	66
8.23.2 Member Function Documentation	66
8.23.2.1 convert()	66
8.23.2.2 toString()	66
8.24 myRhombus::Rhombus Class Reference	67
8.24.1 Detailed Description	67
8.24.2 Constructor & Destructor Documentation	68
8.24.2.1 Rhombus()	68
8.24.3 Member Function Documentation	68
8.24.3.1 area()	68
8.24.3.2 long_diagonal()	68
8.24.3.3 perimeter()	68
8.24.3.4 short_diagonal()	69
8.24.3.5 toString()	69
8.24.4 Member Data Documentation	69
8.24.4.1 _long_diagonal	69
8.24.4.2 _short_diagonal	69
8.25 RhombusParser Class Reference	70
8.25.1 Detailed Description	71
8.25.2 Constructor & Destructor Documentation	71
8.25.2.1 RhombusParser() [1/2]	71
8.25.2.2 ~RhombusParser()	71
8.25.2.3 RhombusParser() [2/2]	71
8.25.3 Member Function Documentation	71
8.25.3.1 getInstance()	71
8.25.3.2 operator=()	71
8.25.3.3 parse()	71
8.25.3.4 toString()	72
8.25.4 Member Data Documentation	72
8.25.4.1 _instance	72
8.26 RhombusToStringConverter Class Reference	72
8.26.1 Detailed Description	73
8.26.2 Member Function Documentation	73
8.26.2.1 convert()	73

8.26.2.2 toString()	73
8.27 ShapesPrinter Class Reference	74
8.27.1 Detailed Description	75
8.27.2 Constructor & Destructor Documentation	75
8.27.2.1 ShapesPrinter()	75
8.27.3 Member Function Documentation	75
8.27.3.1 clear()	75
8.27.3.2 getData()	75
8.27.3.3 performShowDataBehavior()	75
8.27.3.4 performShowTableBehavior()	76
8.27.3.5 push()	76
8.27.3.6 setShowDataBehavior()	76
8.27.3.7 setShowTableBehavior()	76
8.27.3.8 toString()	77
8.27.4 Member Data Documentation	77
8.27.4.1 _data	77
8.27.4.2 _showDataBehavior	77
8.27.4.3 _showTableBehavior	77
8.28 ShowDataCustom Class Reference	77
8.28.1 Detailed Description	78
8.28.2 Member Function Documentation	78
8.28.2.1 showData()	78
8.28.2.2 toString()	78
8.29 ShowDataDefault Class Reference	79
8.29.1 Detailed Description	79
8.29.2 Member Function Documentation	79
8.29.2.1 showData()	79
8.29.2.2 toString()	80
8.30 ShowTableCustom Class Reference	80
8.30.1 Detailed Description	81
8.30.2 Member Function Documentation	81
8.30.2.1 showTable()	81
8.30.2.2 toString()	81
8.31 ShowTableDefault Class Reference	81
8.31.1 Detailed Description	82
8.31.2 Member Function Documentation	82
8.31.2.1 showTable()	82
8.31.2.2 toString()	82
8.32 mySquare::Square Class Reference	83
8.32.1 Detailed Description	83
8.32.2 Constructor & Destructor Documentation	83
8.32.2.1 Square()	83

8.32.3 Member Function Documentation	84
8.32.3.1 area()	84
8.32.3.2 length()	84
8.32.3.3 perimeter()	84
8.32.3.4 toString()	84
8.32.4 Member Data Documentation	85
8.32.4.1 _length	85
8.33 SquareParser Class Reference	85
8.33.1 Detailed Description	86
8.33.2 Constructor & Destructor Documentation	86
8.33.2.1 SquareParser() [1/2]	86
8.33.2.2 ~SquareParser()	86
8.33.2.3 SquareParser() [2/2]	86
8.33.3 Member Function Documentation	87
8.33.3.1 getInstance()	87
8.33.3.2 operator=()	87
8.33.3.3 parse()	87
8.33.3.4 toString()	87
8.33.4 Member Data Documentation	88
8.33.4.1 _instance	88
8.34 SquareToStringConverter Class Reference	88
8.34.1 Detailed Description	89
8.34.2 Member Function Documentation	89
8.34.2.1 convert()	89
8.34.2.2 toString()	89
8.35 myTriangle::Triangle Class Reference	89
8.35.1 Detailed Description	90
8.35.2 Constructor & Destructor Documentation	90
8.35.2.1 Triangle()	90
8.35.3 Member Function Documentation	91
8.35.3.1 area()	91
8.35.3.2 first_edge()	91
8.35.3.3 perimeter()	91
8.35.3.4 second_edge()	92
8.35.3.5 third_edge()	92
8.35.3.6 toString()	92
8.35.4 Member Data Documentation	92
8.35.4.1 _first_edge	92
8.35.4.2 _second_edge	92
8.35.4.3 _third_edge	93
8.36 TriangleParser Class Reference	93
8.36.1 Detailed Description	94

8.36.2 Constructor & Destructor Documentation	94
8.36.2.1 TriangleParser() [1/2]	94
8.36.2.2 ~TriangleParser()	94
8.36.2.3 TriangleParser() [2/2]	94
8.36.3 Member Function Documentation	95
8.36.3.1 getInstance()	95
8.36.3.2 operator=()	95
8.36.3.3 parse()	95
8.36.3.4 toString()	95
8.36.4 Member Data Documentation	96
8.36.4.1 _instance	96
8.37 TriangleToStringConverter Class Reference	96
8.37.1 Detailed Description	97
8.37.2 Member Function Documentation	97
8.37.2.1 convert()	97
8.37.2.2 toString()	97
9 File Documentation	99
9.1 Circle/Circle.cpp File Reference	99
9.2 Circle/Circle.h File Reference	99
9.3 Circle.h	99
9.4 Circle/CircleParser.cpp File Reference	100
9.5 Circle/CircleParser.h File Reference	100
9.6 CircleParser.h	100
9.7 Circle/CircleToStringConverter.cpp File Reference	100
9.8 Circle/CircleToStringConverter.h File Reference	101
9.9 CircleToStringConverter.h	101
9.10 Circle/dllmain.cpp File Reference	101
9.10.1 Function Documentation	101
9.10.1.1 __declspec()	101
9.11 Ellipse/dllmain.cpp File Reference	102
9.11.1 Function Documentation	102
9.11.1.1 __declspec()	102
9.12 IsoscelesTrapezoid/dllmain.cpp File Reference	102
9.12.1 Function Documentation	102
9.12.1.1 __declspec()	102
9.13 Parallelogram/dllmain.cpp File Reference	103
9.13.1 Function Documentation	103
9.13.1.1 __declspec()	103
9.14 Rectangle/dllmain.cpp File Reference	103
9.14.1 Function Documentation	103
9.14.1.1 __declspec()	103

9.15 Rhombus/dllmain.cpp File Reference	104
9.15.1 Function Documentation	104
9.15.1.1 __declspec()	104
9.16 Square/dllmain.cpp File Reference	104
9.16.1 Function Documentation	104
9.16.1.1 __declspec()	104
9.17 Triangle/dllmain.cpp File Reference	105
9.17.1 Function Documentation	105
9.17.1.1 __declspec()	105
9.18 Circle/framework.h File Reference	105
9.18.1 Macro Definition Documentation	105
9.18.1.1 WIN32_LEAN_AND_MEAN	105
9.19 framework.h	105
9.20 Ellipse/framework.h File Reference	106
9.20.1 Macro Definition Documentation	106
9.20.1.1 WIN32_LEAN_AND_MEAN	106
9.21 framework.h	106
9.22 IsoscelesTrapezoid/framework.h File Reference	106
9.22.1 Macro Definition Documentation	106
9.22.1.1 WIN32_LEAN_AND_MEAN	106
9.23 framework.h	106
9.24 Parallelogram/framework.h File Reference	107
9.24.1 Macro Definition Documentation	107
9.24.1.1 WIN32_LEAN_AND_MEAN	107
9.25 framework.h	107
9.26 Rectangle/framework.h File Reference	107
9.26.1 Macro Definition Documentation	107
9.26.1.1 WIN32_LEAN_AND_MEAN	107
9.27 framework.h	107
9.28 Rhombus/framework.h File Reference	108
9.28.1 Macro Definition Documentation	108
9.28.1.1 WIN32_LEAN_AND_MEAN	108
9.29 framework.h	108
9.30 Square/framework.h File Reference	108
9.30.1 Macro Definition Documentation	108
9.30.1.1 WIN32_LEAN_AND_MEAN	108
9.31 framework.h	108
9.32 Triangle/framework.h File Reference	109
9.32.1 Macro Definition Documentation	109
9.32.1.1 WIN32_LEAN_AND_MEAN	109
9.33 framework.h	109
9.34 utils/framework.h File Reference	109

9.34.1 Macro Definition Documentation	109
9.34.1.1 WIN32_LEAN_AND_MEAN	109
9.35 framework.h	109
9.36 Circle/pch.cpp File Reference	110
9.37 Ellipse/pch.cpp File Reference	110
9.38 IsoscelesTrapezoid/pch.cpp File Reference	110
9.39 Parallelogram/pch.cpp File Reference	110
9.40 Rectangle/pch.cpp File Reference	110
9.41 Rhombus/pch.cpp File Reference	110
9.42 Square/pch.cpp File Reference	110
9.43 Triangle/pch.cpp File Reference	110
9.44 utils/pch.cpp File Reference	111
9.45 Circle/pch.h File Reference	111
9.46 pch.h	111
9.47 Ellipse/pch.h File Reference	111
9.48 pch.h	111
9.49 IsoscelesTrapezoid/pch.h File Reference	112
9.50 pch.h	112
9.51 Parallelogram/pch.h File Reference	112
9.52 pch.h	113
9.53 Rectangle/pch.h File Reference	113
9.54 pch.h	113
9.55 Rhombus/pch.h File Reference	113
9.56 pch.h	114
9.57 Square/pch.h File Reference	114
9.58 pch.h	114
9.59 Triangle/pch.h File Reference	114
9.60 pch.h	115
9.61 utils/pch.h File Reference	115
9.62 pch.h	115
9.63 Ellipse/Ellipse.cpp File Reference	115
9.64 Ellipse/Ellipse.h File Reference	115
9.65 Ellipse.h	116
9.66 Ellipse/EllipseParser.cpp File Reference	116
9.67 Ellipse/EllipseParser.h File Reference	116
9.68 EllipseParser.h	117
9.69 Ellipse/EllipseToStringConverter.cpp File Reference	117
9.70 Ellipse/EllipseToStringConverter.h File Reference	117
9.71 EllipseToStringConverter.h	117
9.72 IsoscelesTrapezoid/IsoscelesTrapezoid.cpp File Reference	118
9.73 IsoscelesTrapezoid/IsoscelesTrapezoid.h File Reference	118
9.74 IsoscelesTrapezoid.h	118

9.75 IsoscelesTrapezoid/IsoscelesTrapezoidParser.cpp File Reference	119
9.76 IsoscelesTrapezoid/IsoscelesTrapezoidParser.h File Reference	119
9.77 IsoscelesTrapezoidParser.h	119
9.78 IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.cpp File Reference	119
9.79 IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.h File Reference	120
9.80 IsoscelesTrapezoidToStringConverter.h	120
9.81 Parallelogram/Parallelogram.cpp File Reference	120
9.82 Parallelogram/Parallelogram.h File Reference	120
9.83 Parallelogram.h	121
9.84 Parallelogram/ParallelogramParser.cpp File Reference	121
9.85 Parallelogram/ParallelogramParser.h File Reference	121
9.86 ParallelogramParser.h	122
9.87 Parallelogram/ParallelogramToStringConverter.cpp File Reference	122
9.88 Parallelogram/ParallelogramToStringConverter.h File Reference	122
9.89 ParallelogramToStringConverter.h	122
9.90 README.md File Reference	123
9.91 Rectangle/Rectangle.cpp File Reference	123
9.92 Rectangle/Rectangle.h File Reference	123
9.93 Rectangle.h	123
9.94 Rectangle/RectangleParser.cpp File Reference	124
9.95 Rectangle/RectangleParser.h File Reference	124
9.96 RectangleParser.h	124
9.97 Rectangle/RectangleToStringConverter.cpp File Reference	124
9.98 Rectangle/RectangleToStringConverter.h File Reference	125
9.99 RectangleToStringConverter.h	125
9.100 Rhombus/Rhombus.cpp File Reference	125
9.101 Rhombus/Rhombus.h File Reference	125
9.102 Rhombus.h	126
9.103 Rhombus/RhombusParser.cpp File Reference	126
9.104 Rhombus/RhombusParser.h File Reference	126
9.105 RhombusParser.h	126
9.106 Rhombus/RhombusToStringConverter.cpp File Reference	127
9.107 Rhombus/RhombusToStringConverter.h File Reference	127
9.108 RhombusToStringConverter.h	127
9.109 ShapesParser/ConverterFactory.cpp File Reference	128
9.110 ShapesParser/ConverterFactory.h File Reference	128
9.111 ConverterFactory.h	128
9.112 ShapesParser/IParser.cpp File Reference	128
9.113 ShapesParser/IParser.h File Reference	128
9.114 IParser.h	129
9.115 ShapesParser/IShape.cpp File Reference	129
9.116 ShapesParser/IShape.h File Reference	129

9.117 IShape.h	129
9.118 ShapesParser/IShapeToStringConverter.cpp File Reference	130
9.119 ShapesParser/IShapeToStringConverter.h File Reference	130
9.120 IShapeToStringConverter.h	130
9.121 ShapesParser/IShowDataBehavior.cpp File Reference	130
9.122 ShapesParser/IShowDataBehavior.h File Reference	130
9.123 IShowDataBehavior.h	131
9.124 ShapesParser/IShowTableBehavior.cpp File Reference	131
9.125 ShapesParser/IShowTableBehavior.h File Reference	131
9.126 IShowTableBehavior.h	131
9.127 ShapesParser/main.cpp File Reference	131
9.127.1 Function Documentation	132
9.127.1.1 loadShapesToPrinter()	132
9.127.1.2 main()	132
9.127.1.3 printToScreen()	132
9.127.1.4 readFile()	133
9.127.1.5 setCustomPrinter()	133
9.127.1.6 setMode()	133
9.127.1.7 sortWithLambdaExpression()	134
9.128 ShapesParser/Object.cpp File Reference	134
9.129 ShapesParser/Object.h File Reference	134
9.130 Object.h	134
9.131 ShapesParser/ParserFactory.cpp File Reference	134
9.132 ShapesParser/ParserFactory.h File Reference	135
9.133 ParserFactory.h	135
9.134 ShapesParser/ShapesPrinter.cpp File Reference	135
9.135 ShapesParser/ShapesPrinter.h File Reference	135
9.136 ShapesPrinter.h	136
9.137 ShapesParser/ShowDataCustom.cpp File Reference	136
9.138 ShapesParser/ShowDataCustom.h File Reference	136
9.139 ShowDataCustom.h	137
9.140 ShapesParser/ShowDataDefault.cpp File Reference	137
9.141 ShapesParser/ShowDataDefault.h File Reference	137
9.142 ShowDataDefault.h	137
9.143 ShapesParser/ShowTableCustom.cpp File Reference	137
9.144 ShapesParser/ShowTableCustom.h File Reference	138
9.145 ShowTableCustom.h	138
9.146 ShapesParser/ShowTableDefault.cpp File Reference	138
9.147 ShapesParser/ShowTableDefault.h File Reference	138
9.148 ShowTableDefault.h	139
9.149 ShapesParser/Strategy.cpp File Reference	139
9.150 Square/Square.cpp File Reference	139

9.151 Square/Square.h File Reference	139
9.152 Square.h	140
9.153 Square/SquareParser.cpp File Reference	140
9.154 Square/SquareParser.h File Reference	140
9.155 SquareParser.h	140
9.156 Square/SquareToStringConverter.cpp File Reference	141
9.157 Square/SquareToStringConverter.h File Reference	141
9.158 SquareToStringConverter.h	141
9.159 Triangle/Triangle.cpp File Reference	141
9.160 Triangle/Triangle.h File Reference	142
9.161 Triangle.h	142
9.162 Triangle/TriangleParser.cpp File Reference	142
9.163 Triangle/TriangleParser.h File Reference	142
9.164 TriangleParser.h	143
9.165 Triangle/TriangleToStringConverter.cpp File Reference	143
9.166 Triangle/TriangleToStringConverter.h File Reference	143
9.167 TriangleToStringConverter.h	144
9.168 utils/utils.cpp File Reference	144
9.169 utils/utils.h File Reference	144
9.169.1 Typedef Documentation	145
9.169.1.1 SHAPE_DATA	145
9.169.2 Function Documentation	145
9.169.2.1 DOUBLE_PATTERN()	145
9.169.3 Variable Documentation	145
9.169.3.1 PI	145
9.170 utils.h	146
Index	147

Chapter 1

Đ ÁN CUI KÌ MÔN LP TRÌNH HNG ĐI TNG

21120353: Vi Lý Duy Trng

21120432: Vũ Tin Đt

Chapter 2

shape

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

myCircle	17
myEllipse	17
myIsoscelesTrapezoid	17
myParallelogram	17
myRectangle	18
myRhombus	18
mySquare	18
myTriangle	18

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Object	49
ConverterFactory	26
IParser	35
CircleParser	21
EllipseParser	30
IsoscelesTrapezoidParser	44
ParallelogramParser	53
RectangleParser	62
RhombusParser	70
SquareParser	85
TriangleParser	93
IShape	36
myCircle::Circle	19
myEllipse::Ellipse	27
myIsoscelesTrapezoid::IsoscelesTrapezoid	41
myParallelogram::Parallelogram	50
myRectangle::Rectangle	59
myRhombus::Rhombus	67
mySquare::Square	83
myTriangle::Triangle	89
IShapeToStringConverter	38
CircleToStringConverter	24
EllipseToStringConverter	33
IsoscelesTrapezoidToStringConverter	47
ParallelogramToStringConverter	56
RectangleToStringConverter	65
RhombusToStringConverter	72
SquareToStringConverter	88
TriangleToStringConverter	96
IShowDataBehavior	39
ShowDataCustom	77
ShowDataDefault	79
IShowTableBehavior	40
ShowTableCustom	80
ShowTableDefault	81
ParserFactory	58
ShapesPrinter	74

Chapter 5

Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

myCircle::Circle	
Circle class, which inherits from the IShape interface and stores information about a circle shape	19
CircleParser	
CircleParser class, which inherits from the IParser interface and performs the task of parsing circle shapes	21
CircleToStringConverter	
CircleToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting circle shape information to data set	24
ConverterFactory	
Class to manage a list of prototypes for IShapeToStringConverter objects	26
myEllipse::Ellipse	
Ellipse class, which inherits from the IShape interface and stores information about an ellipse shape	27
EllipseParser	
EllipseParser class, which inherits from the IParser interface and performs the task of parsing ellipse shapes	30
EllipseToStringConverter	
EllipseToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting ellipse shape information to data set	33
IParser	
IParser interface is used for declare methods for subclasses to implement	35
IShape	
IShape interface is used for declare methods for subclasses to implement	36
IShapeToStringConverter	
IShapeToStringConverter interface is used for declare methods for subclasses to implement	38
IShowDataBehavior	
IShowDataBehavior interface is used for declare methods for subclasses to implement	39
IShowTableBehavior	
IShowTableBehavior interface is used for declare methods for subclasses to implement	40
myIsoscelesTrapezoid::IsoscelesTrapezoid	
IsoscelesTrapezoid class, which inherits from the IShape interface and stores information about an isosceles trapezoid shape	41
IsoscelesTrapezoidParser	
IsoscelesTrapezoidParser class, which inherits from the IParser interface and performs the task of parsing isosceles trapezoid shapes	44

IsoscelesTrapezoidToStringConverter	
IsoscelesTrapezoidToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting isosceles trapezoid shape information to data set	47
Object	
Object class is the largest superclass of all classes in the program	49
myParallelogram::Parallelogram	
Parallelogram class, which inherits from the IShape interface and stores information about a parallelogram shape	50
ParallelogramParser	
ParallelogramParser class, which inherits from the IParser interface and performs the task of parsing parallelogram shapes	53
ParallelogramToStringConverter	
ParallelogramToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting parallelogram shape information to data set	56
ParserFactory	
Class to manage a list of prototypes for IParser objects	58
myRectangle::Rectangle	
Rectangle class, which inherits from the IShape interface and stores information about a rectangle shape	59
RectangleParser	
RectangleParser class, which inherits from the IParser interface and performs the task of parsing rectangle shapes	62
RectangleToStringConverter	
RectangleToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting rectangle shape information to data set	65
myRhombus::Rhombus	
Rhombus class, which inherits from the IShape interface and stores information about a rhombus shape	67
RhombusParser	
RhombusParser class, which inherits from the IParser interface and performs the task of parsing rhombus shapes	70
RhombusToStringConverter	
RhombusToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting rhombus shape information to data set	72
ShapesPrinter	
ShapesPrinter class, responsible for printing shapes to the screen	74
ShowDataCustom	
Custom implementation of IShowDataBehavior , responsible for displaying shape data in a customized format	77
ShowDataDefault	
Default implementation of IShowDataBehavior , responsible for displaying shape data in a default format	79
ShowTableCustom	
Custom implementation of IShowTableBehavior , responsible for displaying shape data in a customized table format	80
ShowTableDefault	
Default implementation of IShowTableBehavior , responsible for displaying shape data in a default table format	81
mySquare::Square	
Square class, which inherits from the IShape interface and stores information about a square shape	83
SquareParser	
SquareParser class, which inherits from the IParser interface and performs the task of parsing square shapes	85
SquareToStringConverter	
SquareToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting square shape information to data set	88

myTriangle::Triangle	
Triangle class, which inherits from the IShape interface and stores information about a triangle shape	89
TriangleParser	
TriangleParser class, which inherits from the IParser interface and performs the task of parsing triangle shapes	93
TriangleToStringConverter	
TriangleToStringConverter class, which inherits from the IShapeToStringConverter interface and performs the task of converting triangle shape information to data set	96

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

Circle/ Circle.cpp	99
Circle/ Circle.h	99
Circle/ CircleParser.cpp	100
Circle/ CircleParser.h	100
Circle/ CircleToStringConverter.cpp	100
Circle/ CircleToStringConverter.h	101
Circle/ dllmain.cpp	101
Circle/ framework.h	105
Circle/ pch.cpp	110
Circle/ pch.h	111
Ellipse/ dllmain.cpp	102
Ellipse/ Ellipse.cpp	115
Ellipse/ Ellipse.h	115
Ellipse/ EllipseParser.cpp	116
Ellipse/ EllipseParser.h	116
Ellipse/ EllipseToStringConverter.cpp	117
Ellipse/ EllipseToStringConverter.h	117
Ellipse/ framework.h	106
Ellipse/ pch.cpp	110
Ellipse/ pch.h	111
IsoscelesTrapezoid/ dllmain.cpp	102
IsoscelesTrapezoid/ framework.h	106
IsoscelesTrapezoid/ IsoscelesTrapezoid.cpp	118
IsoscelesTrapezoid/ IsoscelesTrapezoid.h	118
IsoscelesTrapezoid/ IsoscelesTrapezoidParser.cpp	119
IsoscelesTrapezoid/ IsoscelesTrapezoidParser.h	119
IsoscelesTrapezoid/ IsoscelesTrapezoidToStringConverter.cpp	119
IsoscelesTrapezoid/ IsoscelesTrapezoidToStringConverter.h	120
IsoscelesTrapezoid/ pch.cpp	110
IsoscelesTrapezoid/ pch.h	112
Parallelogram/ dllmain.cpp	103
Parallelogram/ framework.h	107
Parallelogram/ Parallelogram.cpp	120
Parallelogram/ Parallelogram.h	120
Parallelogram/ ParallelogramParser.cpp	121

Parallelogram/ParallelogramParser.h	121
Parallelogram/ParallelogramToStringConverter.cpp	122
Parallelogram/ParallelogramToStringConverter.h	122
Parallelogram/pch.cpp	110
Parallelogram/pch.h	112
Rectangle/dllmain.cpp	103
Rectangle/framework.h	107
Rectangle/pch.cpp	110
Rectangle/pch.h	113
Rectangle/Rectangle.cpp	123
Rectangle/Rectangle.h	123
Rectangle/RectangleParser.cpp	124
Rectangle/RectangleParser.h	124
Rectangle/RectangleToStringConverter.cpp	124
Rectangle/RectangleToStringConverter.h	125
Rhombus/dllmain.cpp	104
Rhombus/framework.h	108
Rhombus/pch.cpp	110
Rhombus/pch.h	113
Rhombus/Rhombus.cpp	125
Rhombus/Rhombus.h	125
Rhombus/RhombusParser.cpp	126
Rhombus/RhombusParser.h	126
Rhombus/RhombusToStringConverter.cpp	127
Rhombus/RhombusToStringConverter.h	127
ShapesParser/ConverterFactory.cpp	128
ShapesParser/ConverterFactory.h	128
ShapesParser/IParser.cpp	128
ShapesParser/IParser.h	128
ShapesParser/IShape.cpp	129
ShapesParser/IShape.h	129
ShapesParser/IShapeToStringConverter.cpp	130
ShapesParser/IShapeToStringConverter.h	130
ShapesParser/IShowDataBehavior.cpp	130
ShapesParser/IShowDataBehavior.h	130
ShapesParser/IShowTableBehavior.cpp	131
ShapesParser/IShowTableBehavior.h	131
ShapesParser/main.cpp	131
ShapesParser/Object.cpp	134
ShapesParser/Object.h	134
ShapesParser/ParserFactory.cpp	134
ShapesParser/ParserFactory.h	135
ShapesParser/ShapesPrinter.cpp	135
ShapesParser/ShapesPrinter.h	135
ShapesParser/ShowDataCustom.cpp	136
ShapesParser/ShowDataCustom.h	136
ShapesParser/ShowDataDefault.cpp	137
ShapesParser/ShowDataDefault.h	137
ShapesParser/ShowTableCustom.cpp	137
ShapesParser/ShowTableCustom.h	138
ShapesParser/ShowTableDefault.cpp	138
ShapesParser/ShowTableDefault.h	138
ShapesParser/Strategy.cpp	139
Square/dllmain.cpp	104
Square/framework.h	108
Square/pch.cpp	110
Square/pch.h	114
Square/Square.cpp	139

Square/Square.h	139
Square/SquareParser.cpp	140
Square/SquareParser.h	140
Square/SquareToStringConverter.cpp	141
Square/SquareToStringConverter.h	141
Triangle/dllmain.cpp	105
Triangle/framework.h	109
Triangle/pch.cpp	110
Triangle/pch.h	114
Triangle/Triangle.cpp	141
Triangle/Triangle.h	142
Triangle/TriangleParser.cpp	142
Triangle/TriangleParser.h	142
Triangle/TriangleToStringConverter.cpp	143
Triangle/TriangleToStringConverter.h	143
utils/framework.h	109
utils/pch.cpp	111
utils/pch.h	115
utils/utils.cpp	144
utils/utils.h	144

Chapter 7

Namespace Documentation

7.1 myCircle Namespace Reference

Classes

- class [Circle](#)
[Circle](#) class, which inherits from the [IShape](#) interface and stores information about a circle shape.

7.2 myEllipse Namespace Reference

Classes

- class [Ellipse](#)
[Ellipse](#) class, which inherits from the [IShape](#) interface and stores information about an ellipse shape.

7.3 myIsoscelesTrapezoid Namespace Reference

Classes

- class [IsoscelesTrapezoid](#)
[IsoscelesTrapezoid](#) class, which inherits from the [IShape](#) interface and stores information about an isosceles trapezoid shape.

7.4 myParallelogram Namespace Reference

Classes

- class [Parallelogram](#)
[Parallelogram](#) class, which inherits from the [IShape](#) interface and stores information about a parallelogram shape.

7.5 myRectangle Namespace Reference

Classes

- class [Rectangle](#)
[Rectangle](#) class, which inherits from the [IShape](#) interface and stores information about a rectangle shape.

7.6 myRhombus Namespace Reference

Classes

- class [Rhombus](#)
[Rhombus](#) class, which inherits from the [IShape](#) interface and stores information about a rhombus shape.

7.7 mySquare Namespace Reference

Classes

- class [Square](#)
[Square](#) class, which inherits from the [IShape](#) interface and stores information about a square shape.

7.8 myTriangle Namespace Reference

Classes

- class [Triangle](#)
[Triangle](#) class, which inherits from the [IShape](#) interface and stores information about a triangle shape.

Chapter 8

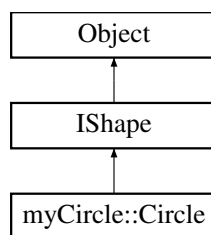
Class Documentation

8.1 myCircle::Circle Class Reference

[Circle](#) class, which inherits from the [IShape](#) interface and stores information about a circle shape.

```
#include <Circle.h>
```

Inheritance diagram for myCircle::Circle:



Public Member Functions

- [Circle](#) (double R) noexcept(false)
Constructor for [Circle](#) class.
- double [area](#) () override
Calculates and returns the area of the circle.
- double [perimeter](#) () override
Calculates and returns the perimeter of the circle.
- string [toString](#) () override
Returns a string representation of the [Circle](#) object.
- double [radius](#) ()
Gets the length of the radius of the circle.

- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.

- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double `_radius`

The length of the radius of the circle.

8.1.1 Detailed Description

`Circle` class, which inherits from the `IShape` interface and stores information about a circle shape.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 `Circle()`

```
myCircle::Circle::Circle (
    double R )
```

Constructor for `Circle` class.

Parameters

<i>Length</i>	of the radius of the circle
---------------	-----------------------------

8.1.3 Member Function Documentation

8.1.3.1 `area()`

```
double myCircle::Circle::area ( ) [override], [virtual]
```

Calculates and returns the area of the circle.

Returns

Area of the circle

Implements `IShape`.

8.1.3.2 `perimeter()`

```
double myCircle::Circle::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the circle.

Returns

Perimeter of the circle

Implements `IShape`.

8.1.3.3 radius()

```
double myCircle::Circle::radius ( )
```

Gets the length of the radius of the circle.

Returns

Length of the radius of the circle

8.1.3.4 toString()

```
string myCircle::Circle::toString ( ) [override], [virtual]
```

Returns a string representation of the [Circle](#) object.

Returns

String representation of the [Circle](#) object

Implements [Object](#).

8.1.4 Member Data Documentation

8.1.4.1 _radius

```
double myCircle::Circle::_radius [private]
```

The length of the radius of the circle.

The documentation for this class was generated from the following files:

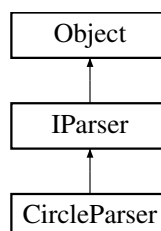
- [Circle/Circle.h](#)
- [Circle/Circle.cpp](#)

8.2 CircleParser Class Reference

[CircleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing circle shapes.

```
#include <CircleParser.h>
```

Inheritance diagram for CircleParser:



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns a [Circle](#) object.
- string [toString](#) () override
Returns a string representation of the [CircleParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [CircleParser](#) * [getInstance](#) ()
Gets the singleton instance of [CircleParser](#).

Private Member Functions

- [CircleParser](#) ()=default
Private constructor for [CircleParser](#) class.
- ~[CircleParser](#) ()=default
Private destructor for [CircleParser](#) class.
- [CircleParser](#) (const [CircleParser](#) &)=delete
Private copy constructor for [CircleParser](#) class.
- [CircleParser](#) & [operator=](#) (const [CircleParser](#) &)=delete
Private copy assignment operator for [CircleParser](#) class.

Static Private Attributes

- static [CircleParser](#) * [_instance](#) = nullptr
Singleton instance of [CircleParser](#).

8.2.1 Detailed Description

[CircleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing circle shapes.

8.2.2 Constructor & Destructor Documentation

8.2.2.1 CircleParser() [1/2]

```
CircleParser::CircleParser ( ) [private], [default]
```

Private constructor for [CircleParser](#) class.

8.2.2.2 ~CircleParser()

```
CircleParser::~~CircleParser ( ) [private], [default]
```

Private destructor for [CircleParser](#) class.

8.2.2.3 CircleParser() [2/2]

```
CircleParser::CircleParser (
    const CircleParser & ) [private], [delete]
```

Private copy constructor for [CircleParser](#) class.

8.2.3 Member Function Documentation

8.2.3.1 getInstance()

```
CircleParser * CircleParser::getInstance ( ) [static]
```

Gets the singleton instance of [CircleParser](#).

Returns

Singleton instance of [CircleParser](#)

8.2.3.2 operator=()

```
CircleParser & CircleParser::operator= (
    const CircleParser & ) [private], [delete]
```

Private copy assignment operator for [CircleParser](#) class.

8.2.3.3 parse()

```
IShape * CircleParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns a Circle object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

Circle object parsed from the input data

Exceptions

<code>std::exception</code>	if unable to parse the input data
-----------------------------	-----------------------------------

Implements [IParser](#).

8.2.3.4 toString()

```
string CircleParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [CircleParser](#) object.

Returns

String representation of the [CircleParser](#) object

Implements [Object](#).

8.2.4 Member Data Documentation**8.2.4.1 _instance**

```
CircleParser* CircleParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [CircleParser](#).

The documentation for this class was generated from the following files:

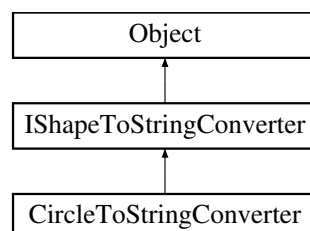
- Circle/[CircleParser.h](#)
- Circle/[CircleParser.cpp](#)

8.3 CircleToStringConverter Class Reference

[CircleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting circle shape information to data set.

```
#include <CircleToStringConverter.h>
```

Inheritance diagram for CircleToStringConverter:



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape *](#)) override
Converts a Circle object to SHAPE_DATA format.
- string [toString](#) () override
Returns a string representation of the [CircleToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape *shape](#))=0
Method to convert [IShape](#) object to SHAPE_DATA data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.3.1 Detailed Description

[CircleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting circle shape information to data set.

8.3.2 Member Function Documentation

8.3.2.1 convert()

```
SHAPE_DATA CircleToStringConverter::convert (
    IShape * shape ) [override], [virtual]
```

Converts a Circle object to SHAPE_DATA format.

Parameters

<i>Pointer</i>	to the Circle object to be converted
----------------	--------------------------------------

Returns

SHAPE_DATA formatted version of the Circle object

Implements [IShapeToStringConverter](#).

8.3.2.2 toString()

```
string CircleToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [CircleToStringConverter](#) object.

Returns

String representation of the [CircleToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

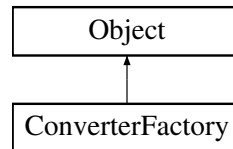
- Circle/[CircleToStringConverter.h](#)
- Circle/[CircleToStringConverter.cpp](#)

8.4 ConverterFactory Class Reference

Class to manage a list of prototypes for [IShapeToStringConverter](#) objects.

```
#include <ConverterFactory.h>
```

Inheritance diagram for ConverterFactory:



Public Member Functions

- void [registerWith](#) (string type, [IShapeToStringConverter](#) *parser)
Register a new prototype with the factory.
- [IShapeToStringConverter](#) * [select](#) (string type)
Select a prototype from the factory based on its type.
- string [toString](#) () override
Return a string representation of the list of prototypes registered with the factory.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- map< string, [IShapeToStringConverter](#) * > [_prototypes](#)

8.4.1 Detailed Description

Class to manage a list of prototypes for [IShapeToStringConverter](#) objects.

8.4.2 Member Function Documentation

8.4.2.1 registerWith()

```
void ConverterFactory::registerWith (
    string type,
    IShapeToStringConverter * parser )
```

Register a new prototype with the factory.

Parameters

<i>type</i>	The name of the type of the prototype being registered.
<i>parser</i>	A pointer to the prototype object.

8.4.2.2 select()

```
IShapeToStringConverter * ConverterFactory::select (
    string type )
```

Select a prototype from the factory based on its type.

Parameters

<i>type</i>	The name of the type of the prototype being selected.
-------------	---

Returns

A pointer to the selected prototype object. If no prototype is found with the given type, returns null.

8.4.2.3 toString()

```
string ConverterFactory::toString ( ) [override], [virtual]
```

Return a string representation of the list of prototypes registered with the factory.

Returns

A string describing the list of prototypes registered with the factory.

Implements [Object](#).

8.4.3 Member Data Documentation

8.4.3.1 _prototypes

```
map<string, IShapeToStringConverter*> ConverterFactory::_prototypes [private]
```

The documentation for this class was generated from the following files:

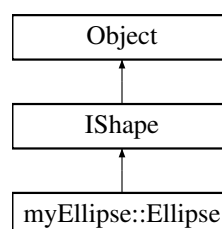
- ShapesParser/[ConverterFactory.h](#)
- ShapesParser/[ConverterFactory.cpp](#)

8.5 myEllipse::Ellipse Class Reference

[Ellipse](#) class, which inherits from the [IShape](#) interface and stores information about an ellipse shape.

```
#include <Ellipse.h>
```

Inheritance diagram for myEllipse::Ellipse:



Public Member Functions

- [Ellipse](#) (double, double) noexcept(false)
Constructor for [Ellipse](#) class.
- double [area](#) () override
Calculates and returns the area of the ellipse.
- double [perimeter](#) () override
Calculates and returns the perimeter of the ellipse.
- string [toString](#) () override
Returns a string representation of the [Ellipse](#) object.
- double [semi_minor_axis](#) ()
Gets the semi-minor axis of the ellipse.
- double [semi_major_axis](#) ()
Gets the semi-major axis of the ellipse.

- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.

- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double [_semi_minor_axis](#)
The semi-minor axis of the ellipse.
- double [_semi_major_axis](#)
The semi-major axis of the ellipse.

8.5.1 Detailed Description

[Ellipse](#) class, which inherits from the [IShape](#) interface and stores information about an ellipse shape.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Ellipse()

```
myEllipse::Ellipse::Ellipse (
    double semi_minor_axis,
    double semi_major_axis )
```

Constructor for [Ellipse](#) class.

Parameters

<i>Semi-minor</i>	axis of the ellipse
<i>Semi-major</i>	axis of the ellipse

8.5.3 Member Function Documentation

8.5.3.1 area()

```
double myEllipse::Ellipse::area ( ) [override], [virtual]
```

Calculates and returns the area of the ellipse.

Returns

Area of the ellipse

Implements [IShape](#).

8.5.3.2 perimeter()

```
double myEllipse::Ellipse::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the ellipse.

Returns

Perimeter of the ellipse

Implements [IShape](#).

8.5.3.3 semi_major_axis()

```
double myEllipse::Ellipse::semi_major_axis ( )
```

Gets the semi-major axis of the ellipse.

Returns

Semi-major axis of the ellipse

8.5.3.4 semi_minor_axis()

```
double myEllipse::Ellipse::semi_minor_axis ( )
```

Gets the semi-minor axis of the ellipse.

Returns

Semi-minor axis of the ellipse

8.5.3.5 toString()

```
string myEllipse::Ellipse::toString ( ) [override], [virtual]
```

Returns a string representation of the [Ellipse](#) object.

Returns

String representation of the [Ellipse](#) object

Implements [Object](#).

8.5.4 Member Data Documentation

8.5.4.1 _semi_major_axis

```
double myEllipse::Ellipse::_semi_major_axis [private]
```

The semi-major axis of the ellipse.

8.5.4.2 _semi_minor_axis

```
double myEllipse::Ellipse::_semi_minor_axis [private]
```

The semi-minor axis of the ellipse.

The documentation for this class was generated from the following files:

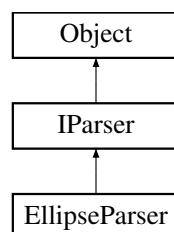
- [Ellipse/Ellipse.h](#)
- [Ellipse/Ellipse.cpp](#)

8.6 EllipseParser Class Reference

[EllipseParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing ellipse shapes.

```
#include <EllipseParser.h>
```

Inheritance diagram for EllipseParser:



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns an [Ellipse](#) object.
- string [toString](#) () override
Returns a string representation of the [EllipseParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [EllipseParser](#) * [getInstance](#) ()
Gets the singleton instance of [EllipseParser](#).

Private Member Functions

- [EllipseParser](#) ()=default
Private constructor for [EllipseParser](#) class.
- ~[EllipseParser](#) ()=default
Private destructor for [EllipseParser](#) class.
- [EllipseParser](#) (const [EllipseParser](#) &)=delete
Private copy constructor for [EllipseParser](#) class.
- [EllipseParser](#) & [operator=](#) (const [EllipseParser](#) &)=delete
Private copy assignment operator for [EllipseParser](#) class.

Static Private Attributes

- static [EllipseParser](#) * [_instance](#) = nullptr
Singleton instance of [EllipseParser](#).

8.6.1 Detailed Description

[EllipseParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing ellipse shapes.

8.6.2 Constructor & Destructor Documentation

8.6.2.1 [EllipseParser](#)() [1/2]

```
EllipseParser::EllipseParser ( ) [private], [default]
```

Private constructor for [EllipseParser](#) class.

8.6.2.2 ~EllipseParser()

```
EllipseParser::~~EllipseParser ( ) [private], [default]
```

Private destructor for [EllipseParser](#) class.

8.6.2.3 EllipseParser() [2/2]

```
EllipseParser::EllipseParser (
    const EllipseParser & ) [private], [delete]
```

Private copy constructor for [EllipseParser](#) class.

8.6.3 Member Function Documentation

8.6.3.1 getInstance()

```
EllipseParser * EllipseParser::getInstance ( ) [static]
```

Gets the singleton instance of [EllipseParser](#).

Returns

Singleton instance of [EllipseParser](#)

8.6.3.2 operator=()

```
EllipseParser & EllipseParser::operator= (
    const EllipseParser & ) [private], [delete]
```

Private copy assignment operator for [EllipseParser](#) class.

8.6.3.3 parse()

```
IShape * EllipseParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns an Ellipse object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

Ellipse object parsed from the input data

Exceptions

<code>std::exception</code>	if unable to parse the input data
-----------------------------	-----------------------------------

Implements [IParser](#).

8.6.3.4 toString()

```
string EllipseParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [EllipseParser](#) object.

Returns

String representation of the [EllipseParser](#) object

Implements [Object](#).

8.6.4 Member Data Documentation

8.6.4.1 _instance

```
EllipseParser* EllipseParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [EllipseParser](#).

The documentation for this class was generated from the following files:

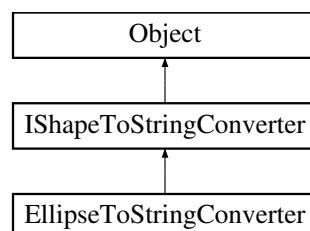
- [Ellipse/EllipseParser.h](#)
- [Ellipse/EllipseParser.cpp](#)

8.7 EllipseToStringConverter Class Reference

[EllipseToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting ellipse shape information to data set.

```
#include <EllipseToStringConverter.h>
```

Inheritance diagram for EllipseToStringConverter:



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape *](#)) override
Converts an [Ellipse](#) object to [SHAPE_DATA](#) format.
- string [toString](#) () override
Returns a string representation of the [EllipseToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape *shape](#))=0
Method to convert [IShape](#) object to [SHAPE_DATA](#) data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.7.1 Detailed Description

[EllipseToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting ellipse shape information to data set.

8.7.2 Member Function Documentation

8.7.2.1 convert()

```
SHAPE_DATA EllipseToStringConverter::convert (
    IShape * shape ) [override], [virtual]
```

Converts an [Ellipse](#) object to [SHAPE_DATA](#) format.

Parameters

<i>Pointer</i>	to the Ellipse object to be converted
----------------	---

Returns

[SHAPE_DATA](#) formatted version of the [Ellipse](#) object

Implements [IShapeToStringConverter](#).

8.7.2.2 toString()

```
string EllipseToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [EllipseToStringConverter](#) object.

Returns

String representation of the [EllipseToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

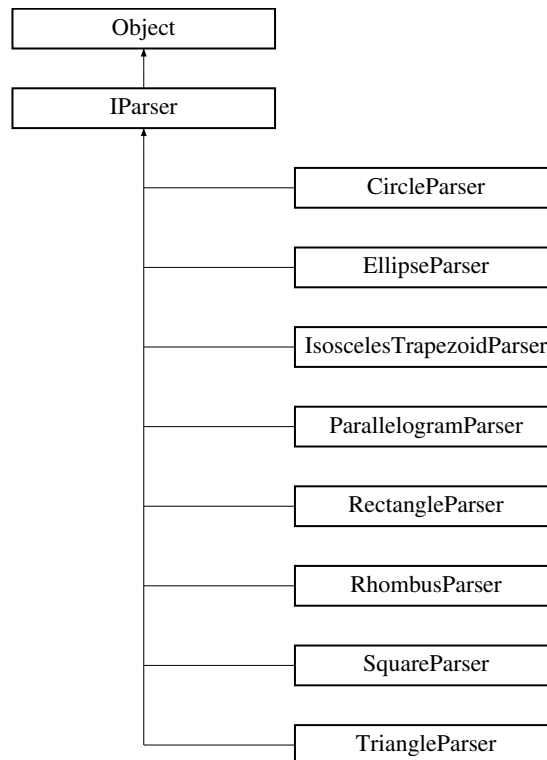
- [Ellipse/EllipseToStringConverter.h](#)
- [Ellipse/EllipseToStringConverter.cpp](#)

8.8 IParser Class Reference

[IParser](#) interface is used for declare methods for subclasses to implement.

```
#include <IParser.h>
```

Inheritance diagram for IParser:



Public Member Functions

- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.

Public Member Functions inherited from [Object](#)

- virtual string [toString](#) ()=0
Get a string representation of an object.

8.8.1 Detailed Description

[IParser](#) interface is used for declare methods for subclasses to implement.

8.8.2 Member Function Documentation

8.8.2.1 [parse\(\)](#)

```
virtual IShape * IParser::parse (
    stringstream data ) [pure virtual]
```

Method to parse from user input.

Parameters

<i>data</i>	User input
-------------	------------

Returns

IShape* object

Implemented in [CircleParser](#), [EllipseParser](#), [IsoscelesTrapezoidParser](#), [ParallelogramParser](#), [RectangleParser](#), [RhombusParser](#), [SquareParser](#), and [TriangleParser](#).

The documentation for this class was generated from the following file:

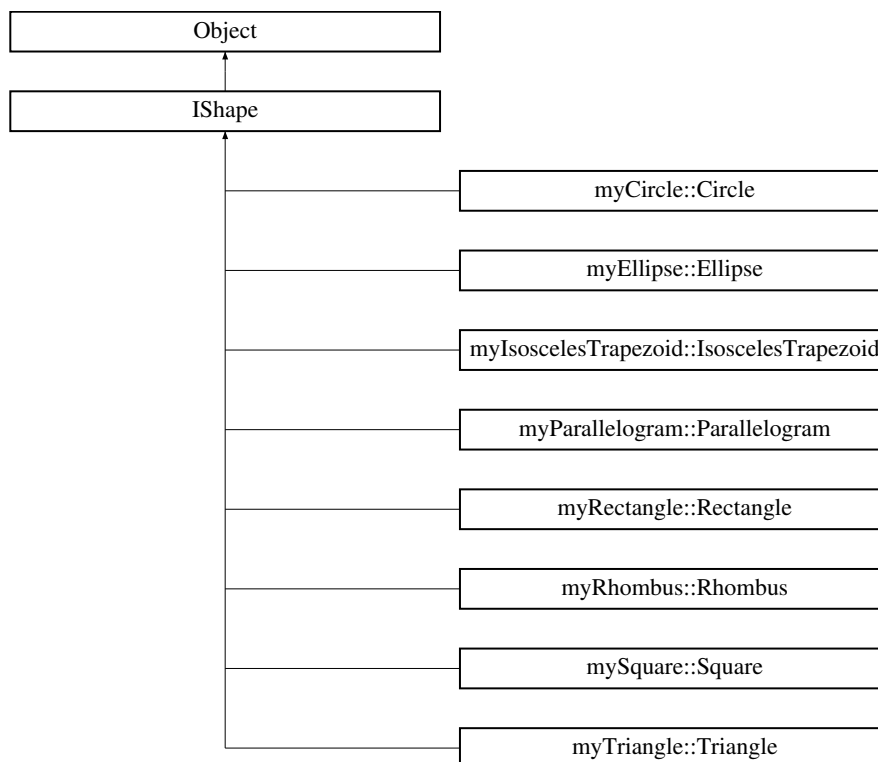
- [ShapesParser/IParser.h](#)

8.9 IShape Class Reference

[IShape](#) interface is used for declare methods for subclasses to implement.

```
#include <IShape.h>
```

Inheritance diagram for IShape:



Public Member Functions

- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.

Public Member Functions inherited from [Object](#)

- virtual string [toString](#) ()=0
Get a string representation of an object.

8.9.1 Detailed Description

[IShape](#) interface is used for declare methods for subclasses to implement.

8.9.2 Member Function Documentation

8.9.2.1 [area\(\)](#)

```
virtual double IShape::area ( ) [pure virtual]
```

Get the area of an object.

Returns

The area of the object

Implemented in [myCircle::Circle](#), [myEllipse::Ellipse](#), [myIsoscelesTrapezoid::IsoscelesTrapezoid](#), [myParallelogram::Parallelogram](#), [myRectangle::Rectangle](#), [myRhombus::Rhombus](#), [mySquare::Square](#), and [myTriangle::Triangle](#).

8.9.2.2 [perimeter\(\)](#)

```
virtual double IShape::perimeter ( ) [pure virtual]
```

Get the perimeter of an object.

Returns

The perimeter of an object

Implemented in [myCircle::Circle](#), [myEllipse::Ellipse](#), [myIsoscelesTrapezoid::IsoscelesTrapezoid](#), [myParallelogram::Parallelogram](#), [myRectangle::Rectangle](#), [myRhombus::Rhombus](#), [mySquare::Square](#), and [myTriangle::Triangle](#).

The documentation for this class was generated from the following file:

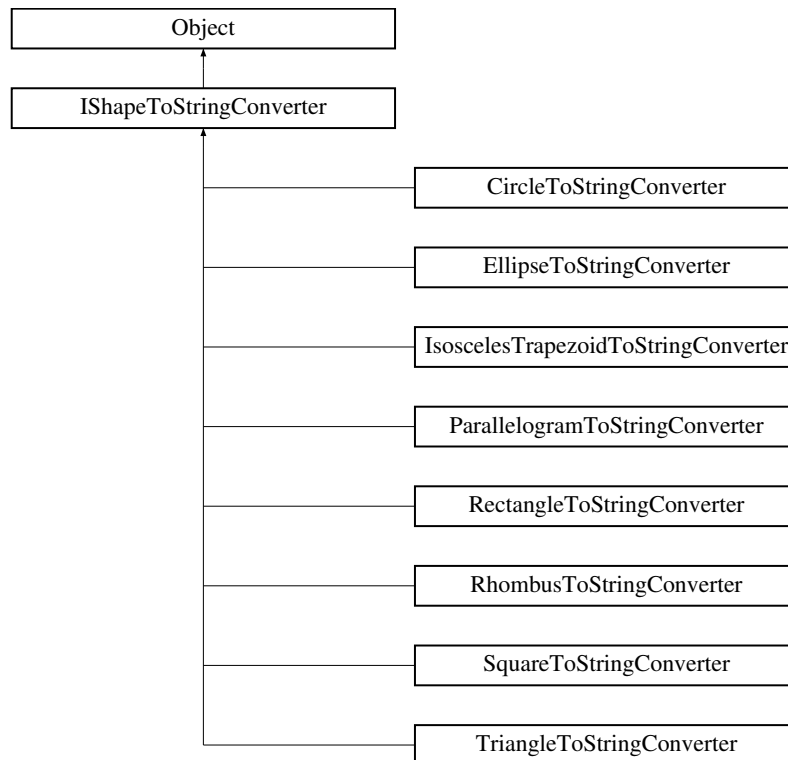
- [ShapesParser/IShape.h](#)

8.10 IShapeToStringConverter Class Reference

[IShapeToStringConverter](#) interface is used for declare methods for subclasses to implement.

```
#include <IShapeToStringConverter.h>
```

Inheritance diagram for IShapeToStringConverter:



Public Member Functions

- virtual [SHAPE_DATA](#) [convert](#) ([IShape](#) *shape)=0
Method to convert [IShape](#) object to [SHAPE_DATA](#) data type.

Public Member Functions inherited from [Object](#)

- virtual string [toString](#) ()=0
Get a string representation of an object.

8.10.1 Detailed Description

[IShapeToStringConverter](#) interface is used for declare methods for subclasses to implement.

8.10.2 Member Function Documentation

8.10.2.1 [convert\(\)](#)

```
virtual SHAPE\_DATA IShapeToStringConverter::convert (
    IShape * shape ) [pure virtual]
```

Method to convert [IShape](#) object to [SHAPE_DATA](#) data type.

Parameters

<i>shape</i>	IShape object needs to be converted
--------------	---

Returns

SHAPE_DATA data type of the input object

Implemented in [CircleToStringConverter](#), [EllipseToStringConverter](#), [IsoscelesTrapezoidToStringConverter](#), [ParallelogramToStringConverter](#), [RectangleToStringConverter](#), [RhombusToStringConverter](#), [SquareToStringConverter](#), and [TriangleToStringConverter](#).

The documentation for this class was generated from the following file:

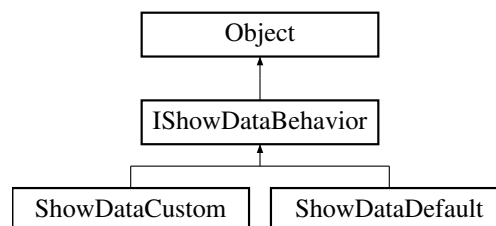
- [ShapesParser/IShapeToStringConverter.h](#)

8.11 IShowDataBehavior Class Reference

[IShowDataBehavior](#) interface is used for declare methods for subclasses to implement.

```
#include <IShowDataBehavior.h>
```

Inheritance diagram for IShowDataBehavior:



Public Member Functions

- virtual void [showData](#) (vector< [SHAPE_DATA](#) > data)=0
Setting method for printing as data line.

Public Member Functions inherited from [Object](#)

- virtual string [toString](#) ()=0
Get a string representation of an object.

8.11.1 Detailed Description

[IShowDataBehavior](#) interface is used for declare methods for subclasses to implement.

8.11.2 Member Function Documentation

8.11.2.1 showData()

```
virtual void IShowDataBehavior::showData (
    vector< SHAPE\_DATA > data ) [pure virtual]
```

Setting method for printing as data line.

Parameters

<i>data</i>	Vector of SHAPE_DATA of IShape objects need to be printed
-------------	---

Implemented in [ShowDataCustom](#), and [ShowDataDefault](#).

The documentation for this class was generated from the following file:

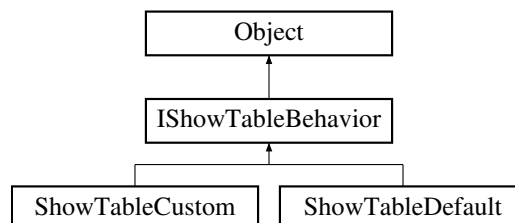
- ShapesParser/[IShowDataBehavior.h](#)

8.12 IShowTableBehavior Class Reference

[IShowTableBehavior](#) interface is used for declare methods for subclasses to implement.

```
#include <IShowTableBehavior.h>
```

Inheritance diagram for IShowTableBehavior:



Public Member Functions

- virtual void [showTable](#) (vector< [SHAPE_DATA](#) >)=0
Setting method for printing as data sheet.

Public Member Functions inherited from [Object](#)

- virtual string [toString](#) ()=0
Get a string representation of an object.

8.12.1 Detailed Description

[IShowTableBehavior](#) interface is used for declare methods for subclasses to implement.

8.12.2 Member Function Documentation

8.12.2.1 [showTable\(\)](#)

```
virtual void IShowTableBehavior::showTable (
    vector< SHAPE\_DATA > ) [pure virtual]
```

Setting method for printing as data sheet.

Parameters

<i>data</i>	Vector of SHAPE_DATA of IShape objects need to be printed
-------------	---

Implemented in [ShowTableCustom](#), and [ShowTableDefault](#).

The documentation for this class was generated from the following file:

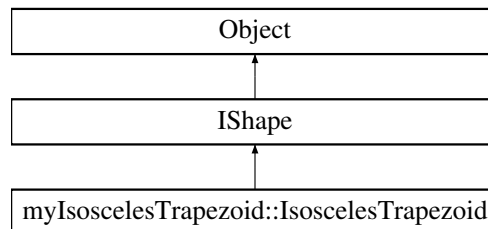
- ShapesParser/[IShowTableBehavior.h](#)

8.13 myIsoscelesTrapezoid::IsoscelesTrapezoid Class Reference

[IsoscelesTrapezoid](#) class, which inherits from the [IShape](#) interface and stores information about an isosceles trapezoid shape.

```
#include <IsoscelesTrapezoid.h>
```

Inheritance diagram for myIsoscelesTrapezoid::IsoscelesTrapezoid:



Public Member Functions

- [IsoscelesTrapezoid](#) (double, double, double) noexcept(false)
Constructor for [IsoscelesTrapezoid](#) class.
- double [area](#) () override
Calculates and returns the area of the isosceles trapezoid.
- double [perimeter](#) () override
Calculates and returns the perimeter of the isosceles trapezoid.
- string [toString](#) () override
Returns a string representation of the [IsoscelesTrapezoid](#) object.
- double [top](#) ()
Gets the length of the top base of the isosceles trapezoid.
- double [base](#) ()
Gets the length of the bottom base of the isosceles trapezoid.
- double [height](#) ()
Gets the height of the isosceles trapezoid.
- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double [_top](#)
The length of the top base of the isosceles trapezoid.
- double [_base](#)
The length of the bottom base of the isosceles trapezoid.
- double [_height](#)
The height of the isosceles trapezoid.

8.13.1 Detailed Description

[IsoscelesTrapezoid](#) class, which inherits from the [IShape](#) interface and stores information about an isosceles trapezoid shape.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 IsoscelesTrapezoid()

```
myIsoscelesTrapezoid::IsoscelesTrapezoid::IsoscelesTrapezoid (
    double top,
    double base,
    double height )
```

Constructor for [IsoscelesTrapezoid](#) class.

Parameters

<i>Length</i>	of the top base of the isosceles trapezoid
<i>Length</i>	of the bottom base of the isosceles trapezoid
<i>Height</i>	of the isosceles trapezoid

8.13.3 Member Function Documentation

8.13.3.1 area()

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::area ( ) [override], [virtual]
```

Calculates and returns the area of the isosceles trapezoid.

Returns

Area of the isosceles trapezoid

Implements [IShape](#).

8.13.3.2 base()

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::base ( )
```

Gets the length of the bottom base of the isosceles trapezoid.

Returns

Length of the bottom base of the isosceles trapezoid

8.13.3.3 height()

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::height ( )
```

Gets the height of the isosceles trapezoid.

Returns

Height of the isosceles trapezoid

8.13.3.4 perimeter()

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the isosceles trapezoid.

Returns

Perimeter of the isosceles trapezoid

Implements [IShape](#).

8.13.3.5 top()

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::top ( )
```

Gets the length of the top base of the isosceles trapezoid.

Returns

Length of the top base of the isosceles trapezoid

8.13.3.6 toString()

```
string myIsoscelesTrapezoid::IsoscelesTrapezoid::toString ( ) [override], [virtual]
```

Returns a string representation of the [IsoscelesTrapezoid](#) object.

Returns

String representation of the [IsoscelesTrapezoid](#) object

Implements [Object](#).

8.13.4 Member Data Documentation

8.13.4.1 _base

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::_base [private]
```

The length of the bottom base of the isosceles trapezoid.

8.13.4.2 _height

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::_height [private]
```

The height of the isosceles trapezoid.

8.13.4.3 _top

```
double myIsoscelesTrapezoid::IsoscelesTrapezoid::_top [private]
```

The length of the top base of the isosceles trapezoid.

The documentation for this class was generated from the following files:

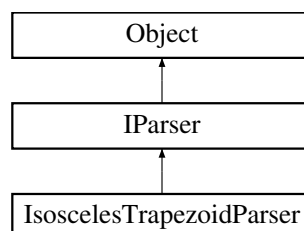
- IsoscelesTrapezoid/[IsoscelesTrapezoid.h](#)
- IsoscelesTrapezoid/[IsoscelesTrapezoid.cpp](#)

8.14 IsoscelesTrapezoidParser Class Reference

[IsoscelesTrapezoidParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing isosceles trapezoid shapes.

```
#include <IsoscelesTrapezoidParser.h>
```

Inheritance diagram for IsoscelesTrapezoidParser:



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns an [IsoscelesTrapezoid](#) object.
- string [toString](#) () override
Returns a string representation of the [IsoscelesTrapezoidParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [IsoscelesTrapezoidParser](#) * [getInstance](#) ()
Gets the singleton instance of [IsoscelesTrapezoidParser](#).

Private Member Functions

- [IsoscelesTrapezoidParser](#) ()=default
Private constructor for [IsoscelesTrapezoidParser](#) class.
- ~[IsoscelesTrapezoidParser](#) ()=default
Private destructor for [IsoscelesTrapezoidParser](#) class.
- [IsoscelesTrapezoidParser](#) (const [IsoscelesTrapezoidParser](#) &)=delete
Private copy constructor for [IsoscelesTrapezoidParser](#) class.
- [IsoscelesTrapezoidParser](#) & operator= (const [IsoscelesTrapezoidParser](#) &)=delete
Private copy assignment operator for [IsoscelesTrapezoidParser](#) class.

Static Private Attributes

- static [IsoscelesTrapezoidParser](#) * [_instance](#) = nullptr
Singleton instance of [IsoscelesTrapezoidParser](#).

8.14.1 Detailed Description

[IsoscelesTrapezoidParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing isosceles trapezoid shapes.

8.14.2 Constructor & Destructor Documentation

8.14.2.1 IsoscelesTrapezoidParser() [1/2]

```
IsoscelesTrapezoidParser::IsoscelesTrapezoidParser ( ) [private], [default]
```

Private constructor for [IsoscelesTrapezoidParser](#) class.

8.14.2.2 ~IsoscelesTrapezoidParser()

```
IsoscelesTrapezoidParser::~~IsoscelesTrapezoidParser ( ) [private], [default]
```

Private destructor for [IsoscelesTrapezoidParser](#) class.

8.14.2.3 IsoscelesTrapezoidParser() [2/2]

```
IsoscelesTrapezoidParser::IsoscelesTrapezoidParser (
    const IsoscelesTrapezoidParser & ) [private], [delete]
```

Private copy constructor for [IsoscelesTrapezoidParser](#) class.

8.14.3 Member Function Documentation

8.14.3.1 getInstance()

```
IsoscelesTrapezoidParser * IsoscelesTrapezoidParser::getInstance ( ) [static]
```

Gets the singleton instance of [IsoscelesTrapezoidParser](#).

Returns

Singleton instance of [IsoscelesTrapezoidParser](#)

8.14.3.2 operator=()

```
IsoscelesTrapezoidParser & IsoscelesTrapezoidParser::operator= (
    const IsoscelesTrapezoidParser & ) [private], [delete]
```

Private copy assignment operator for [IsoscelesTrapezoidParser](#) class.

8.14.3.3 parse()

```
IShape * IsoscelesTrapezoidParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns an [IsoscelesTrapezoid](#) object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

[IsoscelesTrapezoid](#) object parsed from the input data

Exceptions

<code>std::exception</code>	if unable to parse the input data
-----------------------------	-----------------------------------

Implements [IParser](#).

8.14.3.4 toString()

```
string IsoscelesTrapezoidParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [IsoscelesTrapezoidParser](#) object.

Returns

String representation of the [IsoscelesTrapezoidParser](#) object

Implements [Object](#).

8.14.4 Member Data Documentation

8.14.4.1 _instance

```
IsoscelesTrapezoidParser* IsoscelesTrapezoidParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [IsoscelesTrapezoidParser](#).

The documentation for this class was generated from the following files:

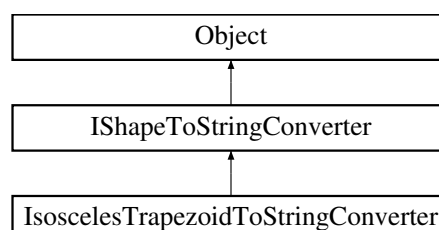
- [IsoscelesTrapezoid/IsoscelesTrapezoidParser.h](#)
- [IsoscelesTrapezoid/IsoscelesTrapezoidParser.cpp](#)

8.15 IsoscelesTrapezoidToStringConverter Class Reference

[IsoscelesTrapezoidToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting isosceles trapezoid shape information to data set.

```
#include <IsoscelesTrapezoidToStringConverter.h>
```

Inheritance diagram for [IsoscelesTrapezoidToStringConverter](#):



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape *](#)) override
Converts an [IsoscelesTrapezoid](#) object to [SHAPE_DATA](#) format.
- string [toString](#) () override
Returns a string representation of the [IsoscelesTrapezoidToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape *shape](#))=0
Method to convert [IShape](#) object to [SHAPE_DATA](#) data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.15.1 Detailed Description

[IsoscelesTrapezoidToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting isosceles trapezoid shape information to data set.

8.15.2 Member Function Documentation

8.15.2.1 convert()

```
SHAPE_DATA IsoscelesTrapezoidToStringConverter::convert (
    IShape * shape ) [override], [virtual]
```

Converts an [IsoscelesTrapezoid](#) object to [SHAPE_DATA](#) format.

Parameters

<i>Pointer</i>	to the IsoscelesTrapezoid object to be converted
----------------	--

Returns

[SHAPE_DATA](#) formatted version of the [IsoscelesTrapezoid](#) object

Implements [IShapeToStringConverter](#).

8.15.2.2 toString()

```
string IsoscelesTrapezoidToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [IsoscelesTrapezoidToStringConverter](#) object.

Returns

String representation of the [IsoscelesTrapezoidToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

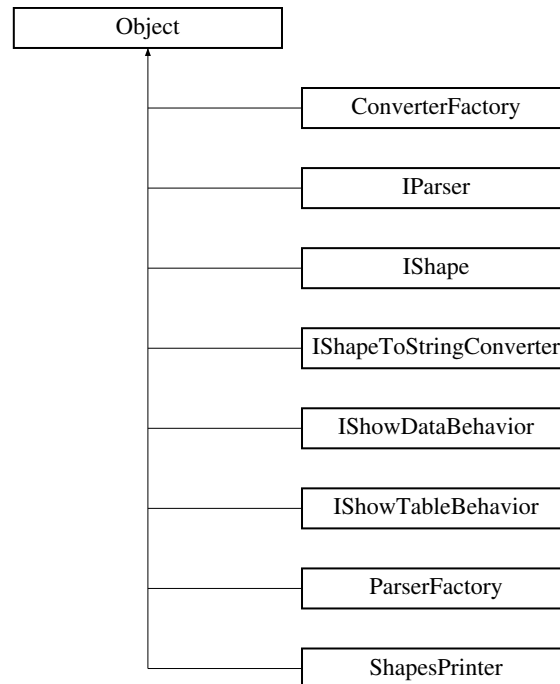
- [IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.h](#)
- [IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.cpp](#)

8.16 Object Class Reference

`Object` class is the largest superclass of all classes in the program.

```
#include <Object.h>
```

Inheritance diagram for `Object`:



Public Member Functions

- virtual string `toString()`=0
Get a string representation of an object.

8.16.1 Detailed Description

`Object` class is the largest superclass of all classes in the program.

8.16.2 Member Function Documentation

8.16.2.1 `toString()`

```
virtual string Object::toString ( ) [pure virtual]
```

Get a string representation of an object.

Returns

The string representation of an object

Implemented in [myCircle::Circle](#), [CircleParser](#), [CircleToStringConverter](#), [myEllipse::Ellipse](#), [EllipseParser](#), [EllipseToStringConverter](#), [myIsoscelesTrapezoid::IsoscelesTrapezoid](#), [IsoscelesTrapezoidParser](#), [IsoscelesTrapezoidToStringConverter](#), [myParallelogram::Parallelogram](#), [ParallelogramParser](#), [ParallelogramToStringConverter](#), [myRectangle::Rectangle](#), [RectangleParser](#), [RectangleToStringConverter](#), [myRhombus::Rhombus](#), [RhombusParser](#), [RhombusToStringConverter](#), [ConverterFactory](#), [ParserFactory](#), [ShapesPrinter](#), [ShowDataCustom](#), [ShowDataDefault](#), [ShowTableCustom](#), [ShowTableDefault](#), [mySquare::Square](#), [SquareParser](#), [SquareToStringConverter](#), [myTriangle::Triangle](#), [TriangleParser](#), and [TriangleToStringConverter](#).

The documentation for this class was generated from the following file:

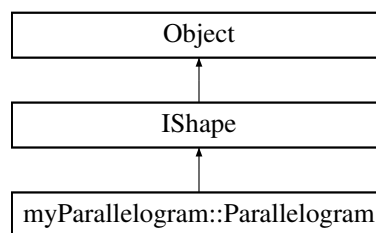
- [ShapesParser/Object.h](#)

8.17 myParallelogram::Parallelogram Class Reference

[Parallelogram](#) class, which inherits from the [IShape](#) interface and stores information about a parallelogram shape.

```
#include <Parallelogram.h>
```

Inheritance diagram for myParallelogram::Parallelogram:

**Public Member Functions**

- [Parallelogram](#) (double, double, double) noexcept(false)
Constructor for [Parallelogram](#) class.
- double [area](#) () override
Calculates and returns the area of the parallelogram.
- double [perimeter](#) () override
Calculates and returns the perimeter of the parallelogram.
- string [toString](#) () override
Returns a string representation of the [Parallelogram](#) object.
- double [side](#) ()
Gets the length of one of the sides of the parallelogram.
- double [base](#) ()
Gets the length of the base of the parallelogram.
- double [height](#) ()
Gets the height of the parallelogram.
- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double `_side`
The length of one of the sides of the parallelogram.
- double `_base`
The length of the base of the parallelogram.
- double `_height`
The height of the parallelogram.

8.17.1 Detailed Description

`Parallelogram` class, which inherits from the `IShape` interface and stores information about a parallelogram shape.

8.17.2 Constructor & Destructor Documentation

8.17.2.1 `Parallelogram()`

```
myParallelogram::Parallelogram::Parallelogram (
    double side,
    double base,
    double height )
```

Constructor for `Parallelogram` class.

Parameters

<i>Length</i>	of one of the sides of the parallelogram
<i>Length</i>	of the base of the parallelogram
<i>Height</i>	of the parallelogram

8.17.3 Member Function Documentation

8.17.3.1 `area()`

```
double myParallelogram::Parallelogram::area ( ) [override], [virtual]
```

Calculates and returns the area of the parallelogram.

Returns

Area of the parallelogram

Implements `IShape`.

8.17.3.2 base()

```
double myParallelogram::Parallelogram::base ( )
```

Gets the length of the base of the parallelogram.

Returns

Length of the base of the parallelogram

8.17.3.3 height()

```
double myParallelogram::Parallelogram::height ( )
```

Gets the height of the parallelogram.

Returns

Height of the parallelogram

8.17.3.4 perimeter()

```
double myParallelogram::Parallelogram::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the parallelogram.

Returns

Perimeter of the parallelogram

Implements [IShape](#).

8.17.3.5 side()

```
double myParallelogram::Parallelogram::side ( )
```

Gets the length of one of the sides of the parallelogram.

Returns

Length of one of the sides of the parallelogram

8.17.3.6 toString()

```
string myParallelogram::Parallelogram::toString ( ) [override], [virtual]
```

Returns a string representation of the [Parallelogram](#) object.

Returns

String representation of the [Parallelogram](#) object

Implements [Object](#).

8.17.4 Member Data Documentation

8.17.4.1 _base

```
double myParallelogram::Parallelogram::_base [private]
```

The length of the base of the parallelogram.

8.17.4.2 _height

```
double myParallelogram::Parallelogram::_height [private]
```

The height of the parallelogram.

8.17.4.3 _side

```
double myParallelogram::Parallelogram::_side [private]
```

The length of one of the sides of the parallelogram.

The documentation for this class was generated from the following files:

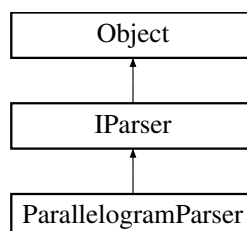
- [Parallelogram/Parallelogram.h](#)
- [Parallelogram/Parallelogram.cpp](#)

8.18 ParallelogramParser Class Reference

[ParallelogramParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing parallelogram shapes.

```
#include <ParallelogramParser.h>
```

Inheritance diagram for [ParallelogramParser](#):



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns a [Parallelogram](#) object.
- string [toString](#) () override
Returns a string representation of the [ParallelogramParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [ParallelogramParser](#) * [getInstance](#) ()
Gets the singleton instance of [ParallelogramParser](#).

Private Member Functions

- [ParallelogramParser](#) ()=default
Private constructor for [ParallelogramParser](#) class.
- [~ParallelogramParser](#) ()=default
Private destructor for [ParallelogramParser](#) class.
- [ParallelogramParser](#) (const [ParallelogramParser](#) &)=delete
Private copy constructor for [ParallelogramParser](#) class.
- [ParallelogramParser](#) & [operator=](#) (const [ParallelogramParser](#) &)=delete
Private copy assignment operator for [ParallelogramParser](#) class.

Static Private Attributes

- static [ParallelogramParser](#) * [_instance](#) = nullptr
Singleton instance of [ParallelogramParser](#).

8.18.1 Detailed Description

[ParallelogramParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing parallelogram shapes.

8.18.2 Constructor & Destructor Documentation

8.18.2.1 [ParallelogramParser](#)() [1/2]

```
ParallelogramParser::ParallelogramParser ( ) [private], [default]
```

Private constructor for [ParallelogramParser](#) class.

8.18.2.2 ~ParallelogramParser()

```
ParallelogramParser::~~ParallelogramParser ( ) [private], [default]
```

Private destructor for [ParallelogramParser](#) class.

8.18.2.3 ParallelogramParser() [2/2]

```
ParallelogramParser::ParallelogramParser (
    const ParallelogramParser & ) [private], [delete]
```

Private copy constructor for [ParallelogramParser](#) class.

8.18.3 Member Function Documentation

8.18.3.1 getInstance()

```
ParallelogramParser * ParallelogramParser::getInstance ( ) [static]
```

Gets the singleton instance of [ParallelogramParser](#).

Returns

Singleton instance of [ParallelogramParser](#)

8.18.3.2 operator=()

```
ParallelogramParser & ParallelogramParser::operator= (
    const ParallelogramParser & ) [private], [delete]
```

Private copy assignment operator for [ParallelogramParser](#) class.

8.18.3.3 parse()

```
IShape * ParallelogramParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns a [Parallelogram](#) object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

[Parallelogram](#) object parsed from the input data

Exceptions

<code>std::exception</code>	if unable to parse the input data
-----------------------------	-----------------------------------

Implements [IParser](#).

8.18.3.4 toString()

```
string ParallelogramParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [ParallelogramParser](#) object.

Returns

String representation of the [ParallelogramParser](#) object

Implements [Object](#).

8.18.4 Member Data Documentation

8.18.4.1 _instance

```
ParallelogramParser* ParallelogramParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [ParallelogramParser](#).

The documentation for this class was generated from the following files:

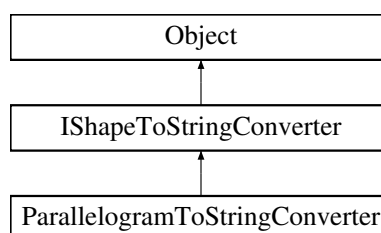
- Parallelogram/[ParallelogramParser.h](#)
- Parallelogram/[ParallelogramParser.cpp](#)

8.19 ParallelogramToStringConverter Class Reference

[ParallelogramToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting parallelogram shape information to data set.

```
#include <ParallelogramToStringConverter.h>
```

Inheritance diagram for [ParallelogramToStringConverter](#):



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape *](#)) override
Converts a Parallelogram object to SHAPE_DATA format.
- string [toString](#) () override
Returns a string representation of the [ParallelogramToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape *shape](#))=0
Method to convert [IShape](#) object to SHAPE_DATA data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.19.1 Detailed Description

[ParallelogramToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting parallelogram shape information to data set.

8.19.2 Member Function Documentation

8.19.2.1 convert()

```
SHAPE_DATA ParallelogramToStringConverter::convert (
    IShape * shape ) [override], [virtual]
```

Converts a Parallelogram object to SHAPE_DATA format.

Parameters

<i>Pointer</i>	to the Parallelogram object to be converted
----------------	---

Returns

SHAPE_DATA formatted version of the Parallelogram object

Implements [IShapeToStringConverter](#).

8.19.2.2 toString()

```
string ParallelogramToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [ParallelogramToStringConverter](#) object.

Returns

String representation of the [ParallelogramToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

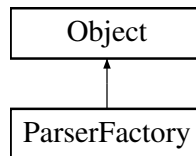
- Parallelogram/[ParallelogramToStringConverter.h](#)
- Parallelogram/[ParallelogramToStringConverter.cpp](#)

8.20 ParserFactory Class Reference

Class to manage a list of prototypes for [IParser](#) objects.

```
#include <ParserFactory.h>
```

Inheritance diagram for ParserFactory:



Public Member Functions

- void [registerWith](#) (string type, [IParser](#) *parser)
Register a new prototype with the factory.
- [IParser](#) * [select](#) (string type)
Select a prototype from the factory based on its type.
- string [toString](#) () override
Return a string representation of the list of prototypes registered with the factory.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- map< string, [IParser](#) * > [_prototypes](#)

8.20.1 Detailed Description

Class to manage a list of prototypes for [IParser](#) objects.

8.20.2 Member Function Documentation

8.20.2.1 registerWith()

```
void ParserFactory::registerWith (
    string type,
    IParser * parser )
```

Register a new prototype with the factory.

Parameters

<i>type</i>	The name of the type of the prototype being registered.
<i>parser</i>	A pointer to the prototype object.

8.20.2.2 select()

```
IParser * ParserFactory::select (
    string type )
```

Select a prototype from the factory based on its type.

Parameters

<i>type</i>	The name of the type of the prototype being selected.
-------------	---

Returns

A pointer to the selected prototype object. If no prototype is found with the given type, returns null.

8.20.2.3 toString()

```
string ParserFactory::toString ( ) [override], [virtual]
```

Return a string representation of the list of prototypes registered with the factory.

Returns

A string describing the list of prototypes registered with the factory.

Implements [Object](#).

8.20.3 Member Data Documentation

8.20.3.1 _prototypes

```
map<string, IParser*> ParserFactory::_prototypes [private]
```

The documentation for this class was generated from the following files:

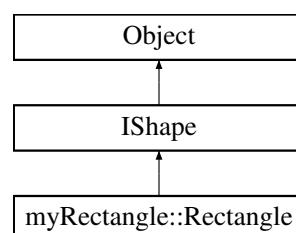
- ShapesParser/[ParserFactory.h](#)
- ShapesParser/[ParserFactory.cpp](#)

8.21 myRectangle::Rectangle Class Reference

[Rectangle](#) class, which inherits from the [IShape](#) interface and stores information about a rectangle shape.

```
#include <Rectangle.h>
```

Inheritance diagram for myRectangle::Rectangle:



Public Member Functions

- [Rectangle](#) (double, double) noexcept(false)
Constructor for [Rectangle](#) class.
- double [area](#) () override
Calculates and returns the area of the rectangle.
- double [perimeter](#) () override
Calculates and returns the perimeter of the rectangle.
- string [toString](#) () override
Returns a string representation of the [Rectangle](#) object.
- double [width](#) ()
Gets the width of the rectangle.
- double [height](#) ()
Gets the height of the rectangle.

- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.

- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double [_width](#)
The width of the rectangle.
- double [_height](#)
The height of the rectangle.

8.21.1 Detailed Description

[Rectangle](#) class, which inherits from the [IShape](#) interface and stores information about a rectangle shape.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 Rectangle()

```
myRectangle::Rectangle::Rectangle (
    double width,
    double height )
```

Constructor for [Rectangle](#) class.

Parameters

<i>Width</i>	of the rectangle
<i>Height</i>	of the rectangle

8.21.3 Member Function Documentation

8.21.3.1 area()

```
double myRectangle::Rectangle::area ( ) [override], [virtual]
```

Calculates and returns the area of the rectangle.

Returns

Area of the rectangle

Implements [IShape](#).

8.21.3.2 height()

```
double myRectangle::Rectangle::height ( )
```

Gets the height of the rectangle.

Returns

Height of the rectangle

8.21.3.3 perimeter()

```
double myRectangle::Rectangle::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the rectangle.

Returns

Perimeter of the rectangle

Implements [IShape](#).

8.21.3.4 toString()

```
string myRectangle::Rectangle::toString ( ) [override], [virtual]
```

Returns a string representation of the [Rectangle](#) object.

Returns

String representation of the [Rectangle](#) object

Implements [Object](#).

8.21.3.5 width()

```
double myRectangle::Rectangle::width ( )
```

Gets the width of the rectangle.

Returns

Width of the rectangle

8.21.4 Member Data Documentation

8.21.4.1 _height

```
double myRectangle::Rectangle::_height [private]
```

The height of the rectangle.

8.21.4.2 _width

```
double myRectangle::Rectangle::_width [private]
```

The width of the rectangle.

The documentation for this class was generated from the following files:

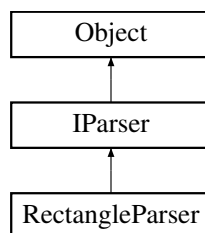
- Rectangle/[Rectangle.h](#)
- Rectangle/[Rectangle.cpp](#)

8.22 RectangleParser Class Reference

[RectangleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing rectangle shapes.

```
#include <RectangleParser.h>
```

Inheritance diagram for RectangleParser:



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns a [Rectangle](#) object.
- string [toString](#) () override
Returns a string representation of the [RectangleParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [RectangleParser](#) * [getInstance](#) ()
Gets the singleton instance of [RectangleParser](#).

Private Member Functions

- [RectangleParser](#) ()=default
Private constructor for [RectangleParser](#) class.
- ~[RectangleParser](#) ()=default
Private destructor for [RectangleParser](#) class.
- [RectangleParser](#) (const [RectangleParser](#) &)=delete
Private copy constructor for [RectangleParser](#) class.
- [RectangleParser](#) & [operator=](#) (const [RectangleParser](#) &)=delete
Private copy assignment operator for [RectangleParser](#) class.

Static Private Attributes

- static [RectangleParser](#) * [_instance](#) = nullptr
Singleton instance of [RectangleParser](#).

8.22.1 Detailed Description

[RectangleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing rectangle shapes.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 [RectangleParser](#)() [1/2]

```
RectangleParser::RectangleParser ( ) [private], [default]
```

Private constructor for [RectangleParser](#) class.

8.22.2.2 ~RectangleParser()

```
RectangleParser::~~RectangleParser ( ) [private], [default]
```

Private destructor for [RectangleParser](#) class.

8.22.2.3 RectangleParser() [2/2]

```
RectangleParser::RectangleParser (
    const RectangleParser & ) [private], [delete]
```

Private copy constructor for [RectangleParser](#) class.

8.22.3 Member Function Documentation

8.22.3.1 getInstance()

```
RectangleParser * RectangleParser::getInstance ( ) [static]
```

Gets the singleton instance of [RectangleParser](#).

Returns

Singleton instance of [RectangleParser](#)

8.22.3.2 operator=()

```
RectangleParser & RectangleParser::operator= (
    const RectangleParser & ) [private], [delete]
```

Private copy assignment operator for [RectangleParser](#) class.

8.22.3.3 parse()

```
IShape * RectangleParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns a Rectangle object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

Rectangle object parsed from the input data

Exceptions

<code>std::exception</code>	if unable to parse the input data
-----------------------------	-----------------------------------

Implements [IParser](#).

8.22.3.4 toString()

```
string RectangleParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [RectangleParser](#) object.

Returns

String representation of the [RectangleParser](#) object

Implements [Object](#).

8.22.4 Member Data Documentation

8.22.4.1 _instance

```
RectangleParser* RectangleParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [RectangleParser](#).

The documentation for this class was generated from the following files:

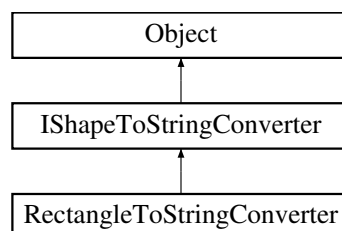
- [Rectangle/RectangleParser.h](#)
- [Rectangle/RectangleParser.cpp](#)

8.23 RectangleToStringConverter Class Reference

[RectangleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting rectangle shape information to data set.

```
#include <RectangleToStringConverter.h>
```

Inheritance diagram for RectangleToStringConverter:



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape *](#)) override
Converts a [Rectangle](#) object to [SHAPE_DATA](#) format.
- string [toString](#) () override
Returns a string representation of the [RectangleToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape *shape](#))=0
Method to convert [IShape](#) object to [SHAPE_DATA](#) data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.23.1 Detailed Description

[RectangleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting rectangle shape information to data set.

8.23.2 Member Function Documentation

8.23.2.1 convert()

```
SHAPE_DATA RectangleToStringConverter::convert (
    IShape * shape ) [override], [virtual]
```

Converts a [Rectangle](#) object to [SHAPE_DATA](#) format.

Parameters

<i>Pointer</i>	to the Rectangle object to be converted
----------------	---

Returns

[SHAPE_DATA](#) formatted version of the [Rectangle](#) object

Implements [IShapeToStringConverter](#).

8.23.2.2 toString()

```
string RectangleToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [RectangleToStringConverter](#) object.

Returns

String representation of the [RectangleToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

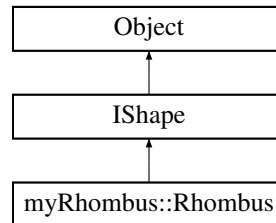
- [Rectangle/RectangleToStringConverter.h](#)
- [Rectangle/RectangleToStringConverter.cpp](#)

8.24 myRhombus::Rhombus Class Reference

[Rhombus](#) class, which inherits from the [IShape](#) interface and stores information about a rhombus shape.

```
#include <Rhombus.h>
```

Inheritance diagram for myRhombus::Rhombus:



Public Member Functions

- [Rhombus](#) (double, double) noexcept(false)
Constructor for [Rhombus](#) class.
- double [area](#) () override
Calculates and returns the area of the rhombus.
- double [perimeter](#) () override
Calculates and returns the perimeter of the rhombus.
- string [toString](#) () override
Returns a string representation of the [Rhombus](#) object.
- double [short_diagonal](#) ()
Gets the length of the short diagonal of the rhombus.
- double [long_diagonal](#) ()
Gets the length of the long diagonal of the rhombus.
- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double [_short_diagonal](#)
The length of the short diagonal of the rhombus.
- double [_long_diagonal](#)
The length of the long diagonal of the rhombus.

8.24.1 Detailed Description

[Rhombus](#) class, which inherits from the [IShape](#) interface and stores information about a rhombus shape.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 Rhombus()

```
myRhombus::Rhombus::Rhombus (
    double short_diagonal,
    double long_diagonal )
```

Constructor for [Rhombus](#) class.

Parameters

<i>Length</i>	of the short diagonal of the rhombus
<i>Length</i>	of the long diagonal of the rhombus

8.24.3 Member Function Documentation

8.24.3.1 area()

```
double myRhombus::Rhombus::area ( ) [override], [virtual]
```

Calculates and returns the area of the rhombus.

Returns

Area of the rhombus

Implements [IShape](#).

8.24.3.2 long_diagonal()

```
double myRhombus::Rhombus::long_diagonal ( )
```

Gets the length of the long diagonal of the rhombus.

Returns

Length of the long diagonal of the rhombus

8.24.3.3 perimeter()

```
double myRhombus::Rhombus::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the rhombus.

Returns

Perimeter of the rhombus

Implements [IShape](#).

8.24.3.4 short_diagonal()

```
double myRhombus::Rhombus::short_diagonal ( )
```

Gets the length of the short diagonal of the rhombus.

Returns

Length of the short diagonal of the rhombus

8.24.3.5 toString()

```
string myRhombus::Rhombus::toString ( ) [override], [virtual]
```

Returns a string representation of the [Rhombus](#) object.

Returns

String representation of the [Rhombus](#) object

Implements [Object](#).

8.24.4 Member Data Documentation

8.24.4.1 _long_diagonal

```
double myRhombus::Rhombus::_long_diagonal [private]
```

The length of the long diagonal of the rhombus.

8.24.4.2 _short_diagonal

```
double myRhombus::Rhombus::_short_diagonal [private]
```

The length of the short diagonal of the rhombus.

The documentation for this class was generated from the following files:

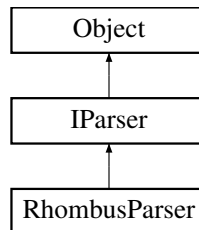
- Rhombus/[Rhombus.h](#)
- Rhombus/[Rhombus.cpp](#)

8.25 RhombusParser Class Reference

[RhombusParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing rhombus shapes.

```
#include <RhombusParser.h>
```

Inheritance diagram for RhombusParser:



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns a Rhombus object.
- string [toString](#) () override
Returns a string representation of the [RhombusParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [RhombusParser](#) * [getInstance](#) ()
Gets the singleton instance of [RhombusParser](#).

Private Member Functions

- [RhombusParser](#) ()=default
Private constructor for [RhombusParser](#) class.
- ~[RhombusParser](#) ()=default
Private destructor for [RhombusParser](#) class.
- [RhombusParser](#) (const [RhombusParser](#) &)=delete
Private copy constructor for [RhombusParser](#) class.
- [RhombusParser](#) & [operator=](#) (const [RhombusParser](#) &)=delete
Private copy assignment operator for [RhombusParser](#) class.

Static Private Attributes

- static [RhombusParser](#) * [_instance](#) = nullptr
Singleton instance of [RhombusParser](#).

8.25.1 Detailed Description

[RhombusParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing rhombus shapes.

8.25.2 Constructor & Destructor Documentation

8.25.2.1 RhombusParser() [1/2]

```
RhombusParser::RhombusParser ( ) [private], [default]
```

Private constructor for [RhombusParser](#) class.

8.25.2.2 ~RhombusParser()

```
RhombusParser::~~RhombusParser ( ) [private], [default]
```

Private destructor for [RhombusParser](#) class.

8.25.2.3 RhombusParser() [2/2]

```
RhombusParser::RhombusParser (
    const RhombusParser & ) [private], [delete]
```

Private copy constructor for [RhombusParser](#) class.

8.25.3 Member Function Documentation

8.25.3.1 getInstance()

```
RhombusParser * RhombusParser::getInstance ( ) [static]
```

Gets the singleton instance of [RhombusParser](#).

Returns

Singleton instance of [RhombusParser](#)

8.25.3.2 operator=()

```
RhombusParser & RhombusParser::operator= (
    const RhombusParser & ) [private], [delete]
```

Private copy assignment operator for [RhombusParser](#) class.

8.25.3.3 parse()

```
IShape * RhombusParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns a Rhombus object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

Rhombus object parsed from the input data

Exceptions

<i>std::exception</i>	if unable to parse the input data
-----------------------	-----------------------------------

Implements [IParser](#).

8.25.3.4 toString()

```
string RhombusParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [RhombusParser](#) object.

Returns

String representation of the [RhombusParser](#) object

Implements [Object](#).

8.25.4 Member Data Documentation**8.25.4.1 _instance**

```
RhombusParser* RhombusParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [RhombusParser](#).

The documentation for this class was generated from the following files:

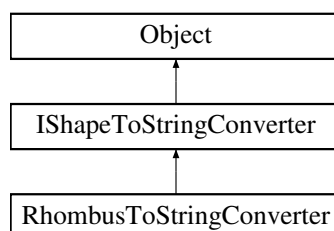
- [Rhombus/RhombusParser.h](#)
- [Rhombus/RhombusParser.cpp](#)

8.26 RhombusToStringConverter Class Reference

[RhombusToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting rhombus shape information to data set.

```
#include <RhombusToStringConverter.h>
```

Inheritance diagram for [RhombusToStringConverter](#):



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape *](#)) override
Converts a Rhombus object to SHAPE_DATA format.
- string [toString](#) () override
Returns a string representation of the [RhombusToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape *shape](#))=0
Method to convert [IShape](#) object to SHAPE_DATA data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.26.1 Detailed Description

[RhombusToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting rhombus shape information to data set.

8.26.2 Member Function Documentation

8.26.2.1 convert()

```
SHAPE_DATA RhombusToStringConverter::convert (
    IShape * shape ) [override], [virtual]
```

Converts a Rhombus object to SHAPE_DATA format.

Parameters

<i>Pointer</i>	to the Rhombus object to be converted
----------------	---------------------------------------

Returns

SHAPE_DATA formatted version of the Rhombus object

Implements [IShapeToStringConverter](#).

8.26.2.2 toString()

```
string RhombusToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [RhombusToStringConverter](#) object.

Returns

String representation of the [RhombusToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

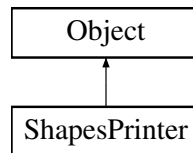
- Rhombus/[RhombusToStringConverter.h](#)
- Rhombus/[RhombusToStringConverter.cpp](#)

8.27 ShapesPrinter Class Reference

[ShapesPrinter](#) class, responsible for printing shapes to the screen.

```
#include <ShapesPrinter.h>
```

Inheritance diagram for ShapesPrinter:



Public Member Functions

- [ShapesPrinter](#) ()
Default constructor for [ShapesPrinter](#) class.
- void [setShowDataBehavior](#) ([IShowDataBehavior](#) *)
Sets the behavior for showing data.
- void [performShowDataBehavior](#) (vector< [SHAPE_DATA](#) >)
Formats the way data is displayed.
- void [setShowTableBehavior](#) ([IShowTableBehavior](#) *)
Sets the behavior for showing tables.
- void [performShowTableBehavior](#) (vector< [SHAPE_DATA](#) >)
Formats the way tables are displayed.
- void [push](#) ([SHAPE_DATA](#))
Adds a shape object to the vector.
- void [clear](#) ()
Clears all shape objects from the vector.
- vector< [SHAPE_DATA](#) > [getData](#) ()
Gets all the shape objects that have been added.
- string [toString](#) () override
Returns a string representation of the [ShapesPrinter](#) object.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- vector< [SHAPE_DATA](#) > [_data](#)
Vector storing information of the shapes.
- [IShowTableBehavior](#) * [_showTableBehavior](#)
Outputs in table format.
- [IShowDataBehavior](#) * [_showDataBehavior](#)
Outputs in data format.

8.27.1 Detailed Description

[ShapesPrinter](#) class, responsible for printing shapes to the screen.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 ShapesPrinter()

```
ShapesPrinter::ShapesPrinter ( )
```

Default constructor for [ShapesPrinter](#) class.

8.27.3 Member Function Documentation

8.27.3.1 clear()

```
void ShapesPrinter::clear ( )
```

Clears all shape objects from the vector.

Parameters

<i>Shape</i>	data
--------------	------

8.27.3.2 getData()

```
vector< SHAPE\_DATA > ShapesPrinter::getData ( )
```

Gets all the shape objects that have been added.

Returns

Vector containing all shape data

8.27.3.3 performShowDataBehavior()

```
void ShapesPrinter::performShowDataBehavior (
    vector< SHAPE\_DATA > data )
```

Formats the way data is displayed.

Parameters

<i>Vector</i>	containing shape data
---------------	-----------------------

8.27.3.4 performShowTableBehavior()

```
void ShapesPrinter::performShowTableBehavior (
    vector< SHAPE_DATA > data )
```

Formats the way tables are displayed.

Parameters

<i>Vector</i>	containing shape data
---------------	-----------------------

8.27.3.5 push()

```
void ShapesPrinter::push (
    SHAPE_DATA other )
```

Adds a shape object to the vector.

Parameters

<i>Shape</i>	data
--------------	------

8.27.3.6 setShowDataBehavior()

```
void ShapesPrinter::setShowDataBehavior (
    IShowDataBehavior * showDataBehavior )
```

Sets the behavior for showing data.

Parameters

<i>The</i>	show data behavior
------------	--------------------

8.27.3.7 setShowTableBehavior()

```
void ShapesPrinter::setShowTableBehavior (
    IShowTableBehavior * showTableBehavior )
```

Sets the behavior for showing tables.

Parameters

<i>The</i>	show table behavior
------------	---------------------

8.27.3.8 toString()

```
string ShapesPrinter::toString ( ) [override], [virtual]
```

Returns a string representation of the [ShapesPrinter](#) object.

Returns

String representation of the [ShapesPrinter](#) object

Implements [Object](#).

8.27.4 Member Data Documentation

8.27.4.1 _data

```
vector<SHAPE_DATA> ShapesPrinter::_data [private]
```

Vector storing information of the shapes.

8.27.4.2 _showDataBehavior

```
IShowDataBehavior* ShapesPrinter::_showDataBehavior [private]
```

Outputs in data format.

8.27.4.3 _showTableBehavior

```
IShowTableBehavior* ShapesPrinter::_showTableBehavior [private]
```

Outputs in table format.

The documentation for this class was generated from the following files:

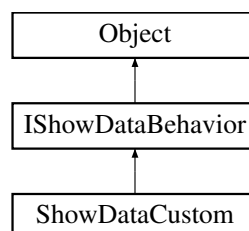
- ShapesParser/[ShapesPrinter.h](#)
- ShapesParser/[ShapesPrinter.cpp](#)

8.28 ShowDataCustom Class Reference

Custom implementation of [IShowDataBehavior](#), responsible for displaying shape data in a customized format.

```
#include <ShowDataCustom.h>
```

Inheritance diagram for ShowDataCustom:



Public Member Functions

- void [showData](#) (vector< [SHAPE_DATA](#) >)
Displays shape data in a customized format.
- string [toString](#) () override
Returns a string representation of the [ShowDataCustom](#) object.
- virtual void [showData](#) (vector< [SHAPE_DATA](#) > data)=0
Setting method for printing as data line.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.28.1 Detailed Description

Custom implementation of [IShowDataBehavior](#), responsible for displaying shape data in a customized format.

8.28.2 Member Function Documentation

8.28.2.1 [showData\(\)](#)

```
void ShowDataCustom::showData (
    vector< SHAPE\_DATA > data ) [virtual]
```

Displays shape data in a customized format.

Parameters

<i>Vector</i>	containing shape data to be displayed
---------------	---------------------------------------

Implements [IShowDataBehavior](#).

8.28.2.2 [toString\(\)](#)

```
string ShowDataCustom::toString ( ) [override], [virtual]
```

Returns a string representation of the [ShowDataCustom](#) object.

Returns

String representation of the [ShowDataCustom](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

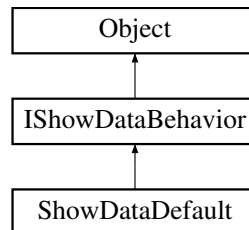
- ShapesParser/[ShowDataCustom.h](#)
- ShapesParser/[ShowDataCustom.cpp](#)

8.29 ShowDataDefault Class Reference

Default implementation of [IShowDataBehavior](#), responsible for displaying shape data in a default format.

```
#include <ShowDataDefault.h>
```

Inheritance diagram for ShowDataDefault:



Public Member Functions

- void [showData](#) (vector< [SHAPE_DATA](#) >)
Displays shape data in a default format.
- string [toString](#) () override
Returns a string representation of the [ShowDataDefault](#) object.
- virtual void [showData](#) (vector< [SHAPE_DATA](#) > data)=0
Setting method for printing as data line.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.29.1 Detailed Description

Default implementation of [IShowDataBehavior](#), responsible for displaying shape data in a default format.

8.29.2 Member Function Documentation

8.29.2.1 showData()

```
void ShowDataDefault::showData (
    vector< SHAPE\_DATA > data ) [virtual]
```

Displays shape data in a default format.

Parameters

<i>Vector</i>	containing shape data to be displayed
---------------	---------------------------------------

Implements [IShowDataBehavior](#).

8.29.2.2 toString()

```
string ShowDataDefault::toString ( ) [override], [virtual]
```

Returns a string representation of the [ShowDataDefault](#) object.

Returns

String representation of the [ShowDataDefault](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

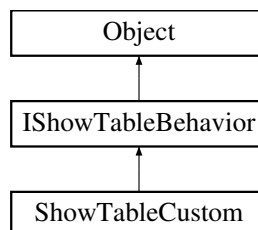
- ShapesParser/[ShowDataDefault.h](#)
- ShapesParser/[ShowDataDefault.cpp](#)

8.30 ShowTableCustom Class Reference

Custom implementation of [IShowTableBehavior](#), responsible for displaying shape data in a customized table format.

```
#include <ShowTableCustom.h>
```

Inheritance diagram for ShowTableCustom:



Public Member Functions

- void [showTable](#) (vector< [SHAPE_DATA](#) >)
Displays shape data in a customized table format.
- string [toString](#) () override
Returns a string representation of the [ShowTableCustom](#) object.
- virtual void [showTable](#) (vector< [SHAPE_DATA](#) >)=0
Setting method for printing as data sheet.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.30.1 Detailed Description

Custom implementation of [IShowTableBehavior](#), responsible for displaying shape data in a customized table format.

8.30.2 Member Function Documentation

8.30.2.1 showTable()

```
void ShowTableCustom::showTable (
    vector< SHAPE\_DATA > data ) [virtual]
```

Displays shape data in a customized table format.

Parameters

<i>Vector</i>	containing shape data to be displayed
---------------	---------------------------------------

Implements [IShowTableBehavior](#).

8.30.2.2 toString()

```
string ShowTableCustom::toString ( ) [override], [virtual]
```

Returns a string representation of the [ShowTableCustom](#) object.

Returns

String representation of the [ShowTableCustom](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

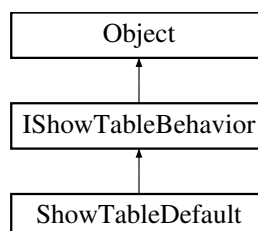
- ShapesParser/[ShowTableCustom.h](#)
- ShapesParser/[ShowTableCustom.cpp](#)

8.31 ShowTableDefault Class Reference

Default implementation of [IShowTableBehavior](#), responsible for displaying shape data in a default table format.

```
#include <ShowTableDefault.h>
```

Inheritance diagram for ShowTableDefault:



Public Member Functions

- void [showTable](#) (vector< [SHAPE_DATA](#) >)
Displays shape data in a default table format.
- string [toString](#) () override
Returns a string representation of the [ShowTableDefault](#) object.
- virtual void [showTable](#) (vector< [SHAPE_DATA](#) >)=0
Setting method for printing as data sheet.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.31.1 Detailed Description

Default implementation of [IShowTableBehavior](#), responsible for displaying shape data in a default table format.

8.31.2 Member Function Documentation

8.31.2.1 [showTable\(\)](#)

```
void ShowTableDefault::showTable (
    vector< SHAPE\_DATA > data ) [virtual]
```

Displays shape data in a default table format.

Parameters

<i>Vector</i>	containing shape data to be displayed
---------------	---------------------------------------

Implements [IShowTableBehavior](#).

8.31.2.2 [toString\(\)](#)

```
string ShowTableDefault::toString ( ) [override], [virtual]
```

Returns a string representation of the [ShowTableDefault](#) object.

Returns

String representation of the [ShowTableDefault](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

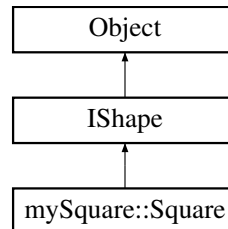
- ShapesParser/[ShowTableDefault.h](#)
- ShapesParser/[ShowTableDefault.cpp](#)

8.32 mySquare::Square Class Reference

[Square](#) class, which inherits from the [IShape](#) interface and stores information about a square shape.

```
#include <Square.h>
```

Inheritance diagram for mySquare::Square:



Public Member Functions

- [Square](#) (double) noexcept(false)
Constructor for [Square](#) class.
- double [area](#) () override
Calculates and returns the area of the square.
- double [perimeter](#) () override
Calculates and returns the perimeter of the square.
- std::string [toString](#) () override
Returns a string representation of the [Square](#) object.
- double [length](#) ()
Gets the length of the sides of the square.
- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double [_length](#)
The length of the sides of the square.

8.32.1 Detailed Description

[Square](#) class, which inherits from the [IShape](#) interface and stores information about a square shape.

8.32.2 Constructor & Destructor Documentation

8.32.2.1 Square()

```
mySquare::Square::Square (
    double len )
```

Constructor for [Square](#) class.

Parameters

<i>Length</i>	of the sides of the square
---------------	----------------------------

8.32.3 Member Function Documentation

8.32.3.1 area()

```
double mySquare::Square::area ( ) [override], [virtual]
```

Calculates and returns the area of the square.

Returns

Area of the square

Implements [IShape](#).

8.32.3.2 length()

```
double mySquare::Square::length ( )
```

Gets the length of the sides of the square.

Returns

Length of the sides of the square

8.32.3.3 perimeter()

```
double mySquare::Square::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the square.

Returns

Perimeter of the square

Implements [IShape](#).

8.32.3.4 toString()

```
string mySquare::Square::toString ( ) [override], [virtual]
```

Returns a string representation of the [Square](#) object.

Returns

String representation of the [Square](#) object

Implements [Object](#).

8.32.4 Member Data Documentation

8.32.4.1 _length

```
double mySquare::Square::_length [private]
```

The length of the sides of the square.

The documentation for this class was generated from the following files:

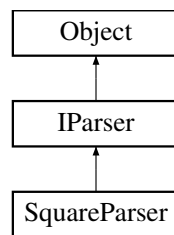
- [Square/Square.h](#)
- [Square/Square.cpp](#)

8.33 SquareParser Class Reference

[SquareParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing square shapes.

```
#include <SquareParser.h>
```

Inheritance diagram for SquareParser:



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns a Square object.
- string [toString](#) () override
Returns a string representation of the [SquareParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [SquareParser](#) * [getInstance](#) ()
Gets the singleton instance of [SquareParser](#).

Private Member Functions

- [SquareParser](#) ()=default
Private constructor for [SquareParser](#) class.
- [~SquareParser](#) ()=default
Private destructor for [SquareParser](#) class.
- [SquareParser](#) (const [SquareParser](#) &)=delete
Private copy constructor for [SquareParser](#) class.
- [SquareParser](#) & operator= (const [SquareParser](#) &)=delete
Private copy assignment operator for [SquareParser](#) class.

Static Private Attributes

- static [SquareParser](#) * [_instance](#) = nullptr
Singleton instance of [SquareParser](#).

8.33.1 Detailed Description

[SquareParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing square shapes.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 SquareParser() [1/2]

```
SquareParser::SquareParser ( ) [private], [default]
```

Private constructor for [SquareParser](#) class.

8.33.2.2 ~SquareParser()

```
SquareParser::~~SquareParser ( ) [private], [default]
```

Private destructor for [SquareParser](#) class.

8.33.2.3 SquareParser() [2/2]

```
SquareParser::SquareParser (
    const SquareParser & ) [private], [delete]
```

Private copy constructor for [SquareParser](#) class.

8.33.3 Member Function Documentation

8.33.3.1 getInstance()

```
SquareParser * SquareParser::getInstance ( ) [static]
```

Gets the singleton instance of [SquareParser](#).

Returns

Singleton instance of [SquareParser](#)

8.33.3.2 operator=()

```
SquareParser & SquareParser::operator= (
    const SquareParser & ) [private], [delete]
```

Private copy assignment operator for [SquareParser](#) class.

8.33.3.3 parse()

```
IShape * SquareParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns a Square object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

Square object parsed from the input data

Exceptions

<i>std::exception</i>	if unable to parse the input data
-----------------------	-----------------------------------

Implements [IParser](#).

8.33.3.4 toString()

```
string SquareParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [SquareParser](#) object.

Returns

String representation of the [SquareParser](#) object

Implements [Object](#).

8.33.4 Member Data Documentation

8.33.4.1 `_instance`

```
SquareParser* SquareParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [SquareParser](#).

The documentation for this class was generated from the following files:

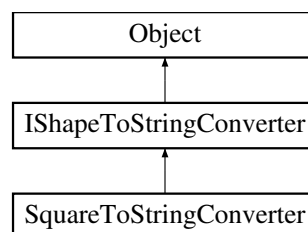
- [Square/SquareParser.h](#)
- [Square/SquareParser.cpp](#)

8.34 SquareToStringConverter Class Reference

[SquareToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting square shape information to data set.

```
#include <SquareToStringConverter.h>
```

Inheritance diagram for [SquareToStringConverter](#):



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape](#) *) override
Converts a *Square* object to *SHAPE_DATA* format.
- string [toString](#) () override
Returns a string representation of the [SquareToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape](#) *shape)=0
Method to convert *IShape* object to *SHAPE_DATA* data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.34.1 Detailed Description

[SquareToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting square shape information to data set.

8.34.2 Member Function Documentation

8.34.2.1 convert()

```
SHAPE_DATA SquareToStringConverter::convert (
    IShape * shape ) [override], [virtual]
```

Converts a Square object to SHAPE_DATA format.

Parameters

<i>Pointer</i>	to the Square object to be converted
----------------	--------------------------------------

Returns

SHAPE_DATA formatted version of the Square object

Implements [IShapeToStringConverter](#).

8.34.2.2 toString()

```
string SquareToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [SquareToStringConverter](#) object.

Returns

String representation of the [SquareToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

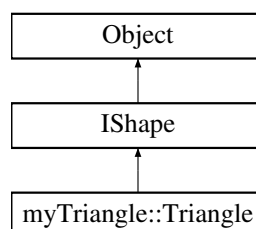
- [Square/SquareToStringConverter.h](#)
- [Square/SquareToStringConverter.cpp](#)

8.35 myTriangle::Triangle Class Reference

[Triangle](#) class, which inherits from the [IShape](#) interface and stores information about a triangle shape.

```
#include <Triangle.h>
```

Inheritance diagram for myTriangle::Triangle:



Public Member Functions

- [Triangle](#) (double, double, double) noexcept(false)
Constructor for [Triangle](#) class.
- double [area](#) () override
Calculates and returns the area of the triangle.
- double [perimeter](#) () override
Calculates and returns the perimeter of the triangle.
- string [toString](#) () override
Returns a string representation of the [Triangle](#) object.
- double [first_edge](#) ()
Gets the length of the first edge of the triangle.
- double [second_edge](#) ()
Gets the length of the second edge of the triangle.
- double [third_edge](#) ()
Gets the length of the third edge of the triangle.

- virtual double [area](#) ()=0
Get the area of an object.
- virtual double [perimeter](#) ()=0
Get the perimeter of an object.

- virtual string [toString](#) ()=0
Get a string representation of an object.

Private Attributes

- double [_first_edge](#)
The length of the first edge of the triangle.
- double [_second_edge](#)
The length of the second edge of the triangle.
- double [_third_edge](#)
The length of the third edge of the triangle.

8.35.1 Detailed Description

[Triangle](#) class, which inherits from the [IShape](#) interface and stores information about a triangle shape.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 Triangle()

```
myTriangle::Triangle::Triangle (
    double firstEdge,
    double secondEdge,
    double thirdEdge )
```

Constructor for [Triangle](#) class.

Parameters

<i>Length</i>	of the first edge of the triangle
<i>Length</i>	of the second edge of the triangle
<i>Length</i>	of the third edge of the triangle

8.35.3 Member Function Documentation

8.35.3.1 area()

```
double myTriangle::Triangle::area ( ) [override], [virtual]
```

Calculates and returns the area of the triangle.

Returns

Area of the triangle

Implements [IShape](#).

8.35.3.2 first_edge()

```
double myTriangle::Triangle::first_edge ( )
```

Gets the length of the first edge of the triangle.

Returns

Length of the first edge of the triangle

8.35.3.3 perimeter()

```
double myTriangle::Triangle::perimeter ( ) [override], [virtual]
```

Calculates and returns the perimeter of the triangle.

Returns

Perimeter of the triangle

Implements [IShape](#).

8.35.3.4 second_edge()

```
double myTriangle::Triangle::second_edge ( )
```

Gets the length of the second edge of the triangle.

Returns

Length of the second edge of the triangle

8.35.3.5 third_edge()

```
double myTriangle::Triangle::third_edge ( )
```

Gets the length of the third edge of the triangle.

Returns

Length of the third edge of the triangle

8.35.3.6 toString()

```
string myTriangle::Triangle::toString ( ) [override], [virtual]
```

Returns a string representation of the [Triangle](#) object.

Returns

String representation of the [Triangle](#) object

Implements [Object](#).

8.35.4 Member Data Documentation

8.35.4.1 _first_edge

```
double myTriangle::Triangle::_first_edge [private]
```

The length of the first edge of the triangle.

8.35.4.2 _second_edge

```
double myTriangle::Triangle::_second_edge [private]
```

The length of the second edge of the triangle.

8.35.4.3 _third_edge

```
double myTriangle::Triangle::_third_edge [private]
```

The length of the third edge of the triangle.

The documentation for this class was generated from the following files:

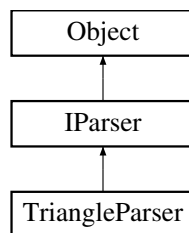
- Triangle/[Triangle.h](#)
- Triangle/[Triangle.cpp](#)

8.36 TriangleParser Class Reference

[TriangleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing triangle shapes.

```
#include <TriangleParser.h>
```

Inheritance diagram for TriangleParser:



Public Member Functions

- [IShape](#) * [parse](#) (stringstream data) noexcept(false) override
Parses the input data and returns a Triangle object.
- string [toString](#) () override
Returns a string representation of the [TriangleParser](#) object.
- virtual [IShape](#) * [parse](#) (stringstream data) noexcept(false)=0
Method to parse from user input.
- virtual string [toString](#) ()=0
Get a string representation of an object.

Static Public Member Functions

- static [TriangleParser](#) * [getInstance](#) ()
Gets the singleton instance of [TriangleParser](#).

Private Member Functions

- [TriangleParser](#) ()=default
Private constructor for [TriangleParser](#) class.
- [~TriangleParser](#) ()=default
Private destructor for [TriangleParser](#) class.
- [TriangleParser](#) (const [TriangleParser](#) &)=delete
Private copy constructor for [TriangleParser](#) class.
- [TriangleParser](#) & operator= (const [TriangleParser](#) &)=delete
Private copy assignment operator for [TriangleParser](#) class.

Static Private Attributes

- static [TriangleParser](#) * [_instance](#) = nullptr
Singleton instance of [TriangleParser](#).

8.36.1 Detailed Description

[TriangleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing triangle shapes.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 [TriangleParser](#)() [1/2]

```
TriangleParser::TriangleParser ( ) [private], [default]
```

Private constructor for [TriangleParser](#) class.

8.36.2.2 [~TriangleParser](#)()

```
TriangleParser::~~TriangleParser ( ) [private], [default]
```

Private destructor for [TriangleParser](#) class.

8.36.2.3 [TriangleParser](#)() [2/2]

```
TriangleParser::TriangleParser (
    const TriangleParser & ) [private], [delete]
```

Private copy constructor for [TriangleParser](#) class.

8.36.3 Member Function Documentation

8.36.3.1 getInstance()

```
TriangleParser * TriangleParser::getInstance ( ) [static]
```

Gets the singleton instance of [TriangleParser](#).

Returns

Singleton instance of [TriangleParser](#)

8.36.3.2 operator=()

```
TriangleParser & TriangleParser::operator= (
    const TriangleParser & ) [private], [delete]
```

Private copy assignment operator for [TriangleParser](#) class.

8.36.3.3 parse()

```
IShape * TriangleParser::parse (
    stringstream data ) [override], [virtual]
```

Parses the input data and returns a Triangle object.

Parameters

<i>Input</i>	data to parse
--------------	---------------

Returns

Triangle object parsed from the input data

Exceptions

<i>std::exception</i>	if unable to parse the input data
-----------------------	-----------------------------------

Implements [IParser](#).

8.36.3.4 toString()

```
string TriangleParser::toString ( ) [override], [virtual]
```

Returns a string representation of the [TriangleParser](#) object.

Returns

String representation of the [TriangleParser](#) object

Implements [Object](#).

8.36.4 Member Data Documentation

8.36.4.1 `_instance`

```
TriangleParser* TriangleParser::_instance = nullptr [inline], [static], [private]
```

Singleton instance of [TriangleParser](#).

The documentation for this class was generated from the following files:

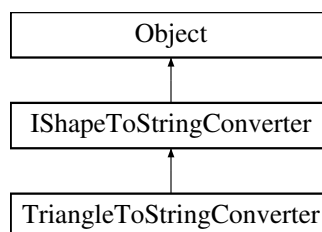
- Triangle/[TriangleParser.h](#)
- Triangle/[TriangleParser.cpp](#)

8.37 TriangleToStringConverter Class Reference

[TriangleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting triangle shape information to data set.

```
#include <TriangleToStringConverter.h>
```

Inheritance diagram for TriangleToStringConverter:



Public Member Functions

- [SHAPE_DATA convert](#) ([IShape](#) *) override
Converts a *Triangle* object to *SHAPE_DATA* format.
- string [toString](#) () override
Returns a string representation of the [TriangleToStringConverter](#) object.
- virtual [SHAPE_DATA convert](#) ([IShape](#) *shape)=0
Method to convert *IShape* object to *SHAPE_DATA* data type.
- virtual string [toString](#) ()=0
Get a string representation of an object.

8.37.1 Detailed Description

[TriangleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting triangle shape information to data set.

8.37.2 Member Function Documentation

8.37.2.1 convert()

```
SHAPE_DATA TriangleToStringConverter::convert (  
    IShape * shape ) [override], [virtual]
```

Converts a Triangle object to SHAPE_DATA format.

Parameters

<i>Pointer</i>	to the Triangle object to be converted
----------------	--

Returns

SHAPE_DATA formatted version of the Triangle object

Implements [IShapeToStringConverter](#).

8.37.2.2 toString()

```
string TriangleToStringConverter::toString ( ) [override], [virtual]
```

Returns a string representation of the [TriangleToStringConverter](#) object.

Returns

String representation of the [TriangleToStringConverter](#) object

Implements [Object](#).

The documentation for this class was generated from the following files:

- Triangle/[TriangleToStringConverter.h](#)
- Triangle/[TriangleToStringConverter.cpp](#)

Chapter 9

File Documentation

9.1 Circle/Circle.cpp File Reference

```
#include "pch.h"
#include "Circle.h"
```

9.2 Circle/Circle.h File Reference

```
#include "pch.h"
```

Classes

- class [myCircle::Circle](#)
[Circle](#) class, which inherits from the [IShape](#) interface and stores information about a circle shape.

Namespaces

- namespace [myCircle](#)

9.3 Circle.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "pch.h"
00003
00004 extern "C" {
00005     namespace myCircle {
00009         class Circle :
00010             public IShape
00011         {
00013             double _radius;
00014         public:
00019             Circle(double R) noexcept(false);
00020
00025             double area() override;
00026
00031             double perimeter() override;
00032
00037             string toString() override;
00038
00043             double radius();
00044         };
00045     }
00046 }
00047
```

9.4 Circle/CircleParser.cpp File Reference

```
#include "pch.h"
#include "CircleParser.h"
```

9.5 Circle/CircleParser.h File Reference

```
#include "pch.h"
#include "Circle.h"
```

Classes

- class [CircleParser](#)

[CircleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing circle shapes.

9.6 CircleParser.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Circle.h"
00005
00006 extern "C" {
00010     class CircleParser :
00011         public IParser
00012     {
00013     private:
00015         inline static CircleParser* _instance = nullptr;
00016
00020         CircleParser() = default;
00021
00025         ~CircleParser() = default;
00026
00030         CircleParser(const CircleParser&) = delete;
00031
00035         CircleParser& operator=(const CircleParser&) = delete;
00036     public:
00041         static CircleParser* getInstance();
00042
00049         IShape* parse(stringstream data) noexcept(false) override;
00050
00055         string toString() override;
00056     };
00057 }
```

9.7 Circle/CircleToStringConverter.cpp File Reference

```
#include "pch.h"
#include "CircleToStringConverter.h"
```

9.8 Circle/CircleToStringConverter.h File Reference

```
#include "pch.h"
#include "Circle.h"
```

Classes

- class [CircleToStringConverter](#)

[CircleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting circle shape information to data set.

9.9 CircleToStringConverter.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Circle.h"
00005
00006 extern "C" {
00010     class CircleToStringConverter :
00011     public IShapeToStringConverter
00012     {
00013     public:
00019         SHAPE_DATA convert(IShape*) override;
00020
00025         string toString() override;
00026     };
00027 }
```

9.10 Circle/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "Circle.h"
#include "CircleToStringConverter.h"
#include "CircleParser.h"
```

Functions

- [__declspec](#) (dllexport) [IParser](#) *__stdcall getParserInstance()

9.10.1 Function Documentation

9.10.1.1 __declspec()

```
__declspec (
    dllexport )
```

9.11 Ellipse/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "Ellipse.h"
#include "EllipseToStringConverter.h"
#include "EllipseParser.h"
```

Functions

- [__declspec](#) (dllexport) [IParser](#) *__stdcall getParserInstance()

9.11.1 Function Documentation

9.11.1.1 [__declspec\(\)](#)

```
\_\_declspec (
    dllexport )
```

9.12 IsoscelesTrapezoid/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "IsoscelesTrapezoid.h"
#include "IsoscelesTrapezoidToStringConverter.h"
#include "IsoscelesTrapezoidParser.h"
```

Functions

- [__declspec](#) (dllexport) [IParser](#) *__stdcall getParserInstance()

9.12.1 Function Documentation

9.12.1.1 [__declspec\(\)](#)

```
\_\_declspec (
    dllexport )
```

9.13 Parallelogram/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "Parallelogram.h"
#include "ParallelogramToStringConverter.h"
#include "ParallelogramParser.h"
```

Functions

- [`__declspec`](#) (dllexport) [`IParser`](#) *__stdcall getParserInstance()

9.13.1 Function Documentation

9.13.1.1 `__declspec()`

```
__declspec (
    dllexport )
```

9.14 Rectangle/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "Rectangle.h"
#include "RectangleToStringConverter.h"
#include "RectangleParser.h"
```

Functions

- [`__declspec`](#) (dllexport) [`IParser`](#) *__stdcall getParserInstance()

9.14.1 Function Documentation

9.14.1.1 `__declspec()`

```
__declspec (
    dllexport )
```

9.15 Rhombus/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "Rhombus.h"
#include "RhombusToStringConverter.h"
#include "RhombusParser.h"
```

Functions

- [__declspec](#) (dllexport) [IParser](#) *__stdcall getParserInstance()

9.15.1 Function Documentation

9.15.1.1 [__declspec\(\)](#)

```
\_\_declspec (
    dllexport )
```

9.16 Square/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "Square.h"
#include "SquareToStringConverter.h"
#include "SquareParser.h"
```

Functions

- [__declspec](#) (dllexport) [IParser](#) *__stdcall getParserInstance()

9.16.1 Function Documentation

9.16.1.1 [__declspec\(\)](#)

```
\_\_declspec (
    dllexport )
```


9.17 Triangle/dllmain.cpp File Reference

```
#include "pch.h"
#include <windows.h>
#include <objbase.h>
#include "Triangle.h"
#include "TriangleToStringConverter.h"
#include "TriangleParser.h"
```

Functions

- [__declspec](#) (dllexport) [IParser](#) *__stdcall getParserInstance()

9.17.1 Function Documentation

9.17.1.1 [__declspec\(\)](#)

```
__declspec (
    dllexport )
```

9.18 Circle/framework.h File Reference

```
#include <windows.h>
```

Macros

- [#define](#) [WIN32_LEAN_AND_MEAN](#)

9.18.1 Macro Definition Documentation

9.18.1.1 [WIN32_LEAN_AND_MEAN](#)

```
#define WIN32_LEAN_AND_MEAN
```

9.19 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.20 Ellipse/framework.h File Reference

```
#include <windows.h>
```

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.20.1 Macro Definition Documentation

9.20.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.21 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.22 IsoscelesTrapezoid/framework.h File Reference

```
#include <windows.h>
```

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.22.1 Macro Definition Documentation

9.22.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.23 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.24 Parallelogram/framework.h File Reference

```
#include <windows.h>
```

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.24.1 Macro Definition Documentation

9.24.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.25 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.26 Rectangle/framework.h File Reference

```
#include <windows.h>
```

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.26.1 Macro Definition Documentation

9.26.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.27 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.28 Rhombus/framework.h File Reference

```
#include <windows.h>
```

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.28.1 Macro Definition Documentation

9.28.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.29 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.30 Square/framework.h File Reference

```
#include <windows.h>
```

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.30.1 Macro Definition Documentation

9.30.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.31 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.32 Triangle/framework.h File Reference

```
#include <windows.h>
```

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.32.1 Macro Definition Documentation

9.32.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.33 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
00004 // Windows Header Files
00005 #include <windows.h>
```

9.34 utils/framework.h File Reference

Macros

- #define [WIN32_LEAN_AND_MEAN](#)

9.34.1 Macro Definition Documentation

9.34.1.1 WIN32_LEAN_AND_MEAN

```
#define WIN32_LEAN_AND_MEAN
```

9.35 framework.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #define WIN32_LEAN_AND_MEAN           // Exclude rarely-used stuff from Windows headers
```

9.36 Circle/pch.cpp File Reference

```
#include "pch.h"
```

9.37 Ellipse/pch.cpp File Reference

```
#include "pch.h"
```

9.38 IsoscelesTrapezoid/pch.cpp File Reference

```
#include "pch.h"
```

9.39 Parallelogram/pch.cpp File Reference

```
#include "pch.h"
```

9.40 Rectangle/pch.cpp File Reference

```
#include "pch.h"
```

9.41 Rhombus/pch.cpp File Reference

```
#include "pch.h"
```

9.42 Square/pch.cpp File Reference

```
#include "pch.h"
```

9.43 Triangle/pch.cpp File Reference

```
#include "pch.h"
```

9.44 utils/pch.cpp File Reference

```
#include "pch.h"
```

9.45 Circle/pch.h File Reference

```
#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"
```

9.46 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
00004 // features.
00004 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00005 // Do not add files here that you will be updating frequently as this negates the performance
00006 // advantage.
00006
00007 #ifndef PCH_H
00008 #define PCH_H
00009
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012 #include "../utils/utils.h"
00013 #include "../ShapesParser/IShape.h"
00014 #include "../ShapesParser/IParser.h"
00015 #include "../ShapesParser/IShapeToStringConverter.h"
00016
00017 #endif //PCH_H
```

9.47 Ellipse/pch.h File Reference

```
#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"
```

9.48 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
00004 // features.
00004 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
```

```

00005 // Do not add files here that you will be updating frequently as this negates the performance
      advantage.
00006
00007 #ifndef PCH_H
00008 #define PCH_H
00009
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012 #include "../utils/utils.h"
00013 #include "../ShapesParser/IShape.h"
00014 #include "../ShapesParser/IParser.h"
00015 #include "../ShapesParser/IShapeToStringConverter.h"
00016
00017 #endif //PCH_H

```

9.49 IsoscelesTrapezoid/pch.h File Reference

```

#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"

```

9.50 pch.h

[Go to the documentation of this file.](#)

```

00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
      features.
00004 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00005 // Do not add files here that you will be updating frequently as this negates the performance
      advantage.
00006
00007 #ifndef PCH_H
00008 #define PCH_H
00009
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012 #include "../utils/utils.h"
00013 #include "../ShapesParser/IShape.h"
00014 #include "../ShapesParser/IParser.h"
00015 #include "../ShapesParser/IShapeToStringConverter.h"
00016
00017 #endif //PCH_H

```

9.51 Parallelogram/pch.h File Reference

```

#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"

```


9.52 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
00004 // features.
00004 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00005 // Do not add files here that you will be updating frequently as this negates the performance
00005 // advantage.
00006
00007 #ifndef PCH_H
00008 #define PCH_H
00009
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012 #include "../utils/utils.h"
00013 #include "../ShapesParser/IShape.h"
00014 #include "../ShapesParser/IParser.h"
00015 #include "../ShapesParser/IShapeToStringConverter.h"
00016
00017 #endif //PCH_H
```

9.53 Rectangle/pch.h File Reference

```
#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"
```

9.54 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
00004 // features.
00004 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00005 // Do not add files here that you will be updating frequently as this negates the performance
00005 // advantage.
00006
00007 #ifndef PCH_H
00008 #define PCH_H
00009
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012 #include "../utils/utils.h"
00013 #include "../ShapesParser/IShape.h"
00014 #include "../ShapesParser/IParser.h"
00015 #include "../ShapesParser/IShapeToStringConverter.h"
00016
00017 #endif //PCH_H
```

9.55 Rhombus/pch.h File Reference

```
#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"
```

9.56 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
00004 // features.
00005 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00006 // Do not add files here that you will be updating frequently as this negates the performance
00007 // advantage.
00008 #ifndef PCH_H
00009 #define PCH_H
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012 #include "../utils/utils.h"
00013 #include "../ShapesParser/IShape.h"
00014 #include "../ShapesParser/IParser.h"
00015 #include "../ShapesParser/IShapeToStringConverter.h"
00016 #endif //PCH_H
```

9.57 Square/pch.h File Reference

```
#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"
```

9.58 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
00004 // features.
00005 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00006 // Do not add files here that you will be updating frequently as this negates the performance
00007 // advantage.
00008 // Nhng header trong file này s đc b tin x lí biên dch trc
00009 #ifndef PCH_H
00010 #define PCH_H
00011 // add headers that you want to pre-compile here
00012 #include "framework.h"
00013 #include "../utils/utils.h"
00014 #include "../ShapesParser/IShape.h"
00015 #include "../ShapesParser/IParser.h"
00016 #include "../ShapesParser/IShapeToStringConverter.h"
00017 #endif //PCH_H
```

9.59 Triangle/pch.h File Reference

```
#include "framework.h"
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "../ShapesParser/IParser.h"
#include "../ShapesParser/IShapeToStringConverter.h"
```

9.60 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
    features.
00004 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00005 // Do not add files here that you will be updating frequently as this negates the performance
    advantage.
00006
00007 #ifndef PCH_H
00008 #define PCH_H
00009
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012 #include "../utils/utils.h"
00013 #include "../ShapesParser/IShape.h"
00014 #include "../ShapesParser/IParser.h"
00015 #include "../ShapesParser/IShapeToStringConverter.h"
00016
00017 #endif //PCH_H
```

9.61 utils/pch.h File Reference

```
#include "framework.h"
```

9.62 pch.h

[Go to the documentation of this file.](#)

```
00001 // pch.h: This is a precompiled header file.
00002 // Files listed below are compiled only once, improving build performance for future builds.
00003 // This also affects IntelliSense performance, including code completion and many code browsing
    features.
00004 // However, files listed here are ALL re-compiled if any one of them is updated between builds.
00005 // Do not add files here that you will be updating frequently as this negates the performance
    advantage.
00006
00007 #ifndef PCH_H
00008 #define PCH_H
00009
00010 // add headers that you want to pre-compile here
00011 #include "framework.h"
00012
00013 #endif //PCH_H
```

9.63 Ellipse/Ellipse.cpp File Reference

```
#include "pch.h"
#include "Ellipse.h"
```

9.64 Ellipse/Ellipse.h File Reference

```
#include "pch.h"
```

Classes

- class [myEllipse::Ellipse](#)
Ellipse class, which inherits from the [IShape](#) interface and stores information about an ellipse shape.

Namespaces

- namespace [myEllipse](#)

9.65 Ellipse.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "pch.h"
00003
00004 extern "C" {
00005     namespace myEllipse {
00009         class Ellipse :
00010             public IShape
00011         {
00013             double _semi_minor_axis;
00014
00016             double _semi_major_axis;
00017         public:
00023             Ellipse(double, double) noexcept(false);
00024
00029             double area() override;
00030
00035             double perimeter() override;
00036
00041             string toString() override;
00042
00047             double semi_minor_axis();
00048
00053             double semi_major_axis();
00054         };
00055     };
00056 }
00057
```

9.66 Ellipse/EllipseParser.cpp File Reference

```
#include "pch.h"
#include "EllipseParser.h"
```

9.67 Ellipse/EllipseParser.h File Reference

```
#include "pch.h"
#include "Ellipse.h"
```

Classes

- class [EllipseParser](#)
EllipseParser class, which inherits from the [IParser](#) interface and performs the task of parsing ellipse shapes.

9.68 EllipseParser.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include "pch.h"
00003 #include "Ellipse.h"
00004
00005 extern "C" {
00006     class EllipseParser :
00007     public IParser
00008     {
00009     private:
00010         inline static EllipseParser* _instance = nullptr;
00011
00012         EllipseParser() = default;
00013
00014         ~EllipseParser() = default;
00015
00016         EllipseParser(const EllipseParser&) = delete;
00017
00018         EllipseParser& operator=(const EllipseParser&) = delete;
00019     public:
00020         static EllipseParser* getInstance();
00021
00022         IShape* parse(stringstream data) noexcept(false) override;
00023
00024         string toString() override;
00025     };
00026 }
00027
00028

```

9.69 Ellipse/EllipseToStringConverter.cpp File Reference

```

#include "pch.h"
#include "EllipseToStringConverter.h"

```

9.70 Ellipse/EllipseToStringConverter.h File Reference

```

#include "pch.h"
#include "Ellipse.h"

```

Classes

- class [EllipseToStringConverter](#)

[EllipseToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting ellipse shape information to data set.

9.71 EllipseToStringConverter.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002 #include "pch.h"
00003 #include "Ellipse.h"
00004
00005 extern "C" {
00006     class EllipseToStringConverter :
00007     public IShapeToStringConverter
00008     {
00009     public:
00010         SHAPE_DATA convert(IShape*) override;
00011
00012         string toString() override;
00013     };
00014 }
00015

```

9.72 IsoscelesTrapezoid/IsoscelesTrapezoid.cpp File Reference

```
#include "pch.h"
#include "IsoscelesTrapezoid.h"
```

9.73 IsoscelesTrapezoid/IsoscelesTrapezoid.h File Reference

```
#include "pch.h"
```

Classes

- class [myIsoscelesTrapezoid::IsoscelesTrapezoid](#)
IsoscelesTrapezoid class, which inherits from the *IShape* interface and stores information about an isosceles trapezoid shape.

Namespaces

- namespace [myIsoscelesTrapezoid](#)

9.74 IsoscelesTrapezoid.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004
00005 extern "C" {
00006     namespace myIsoscelesTrapezoid {
00010         class IsoscelesTrapezoid :
00011             public IShape
00012         {
00014             double _top;
00015
00017             double _base;
00018
00020             double _height;
00021
00022         public:
00029             IsoscelesTrapezoid(double, double, double) noexcept(false);
00030
00035             double area() override;
00036
00041             double perimeter() override;
00042
00047             string toString() override;
00048
00053             double top();
00054
00059             double base();
00060
00065             double height();
00066         };
00067     };
00068 }
```

9.75 IsoscelesTrapezoid/IsoscelesTrapezoidParser.cpp File Reference

```
#include "pch.h"
#include "IsoscelesTrapezoidParser.h"
```

9.76 IsoscelesTrapezoid/IsoscelesTrapezoidParser.h File Reference

```
#include "pch.h"
#include "IsoscelesTrapezoid.h"
```

Classes

- class [IsoscelesTrapezoidParser](#)

[IsoscelesTrapezoidParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing isosceles trapezoid shapes.

9.77 IsoscelesTrapezoidParser.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "IsoscelesTrapezoid.h"
00005
00006 extern "C" {
00010     class IsoscelesTrapezoidParser :
00011         public IParser
00012     {
00013     private:
00015         inline static IsoscelesTrapezoidParser* _instance = nullptr;
00016
00020         IsoscelesTrapezoidParser() = default;
00021
00025         ~IsoscelesTrapezoidParser() = default;
00026
00030         IsoscelesTrapezoidParser(const IsoscelesTrapezoidParser&) = delete;
00031
00035         IsoscelesTrapezoidParser& operator=(const IsoscelesTrapezoidParser&) = delete;
00036     public:
00041         static IsoscelesTrapezoidParser* getInstance();
00042
00049         IShape* parse(stringstream data) noexcept(false) override;
00050
00055         string toString() override;
00056     };
00057 }
```

9.78 IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.cpp File Reference

```
#include "pch.h"
#include "IsoscelesTrapezoidToStringConverter.h"
```

9.79 IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.h File Reference

```
#include "pch.h"
#include "IsoscelesTrapezoid.h"
```

Classes

- class [IsoscelesTrapezoidToStringConverter](#)
IsoscelesTrapezoidToStringConverter class, which inherits from the *IShapeToStringConverter* interface and performs the task of converting isosceles trapezoid shape information to data set.

9.80 IsoscelesTrapezoidToStringConverter.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "IsoscelesTrapezoid.h"
00005
00006 extern "C" {
00010     class IsoscelesTrapezoidToStringConverter :
00011     public IShapeToStringConverter
00012     {
00013     public:
00019         SHAPE_DATA convert(IShape*) override;
00020
00025         string toString() override;
00026     };
00027 }
```

9.81 Parallelogram/Parallelogram.cpp File Reference

```
#include "pch.h"
#include "Parallelogram.h"
```

9.82 Parallelogram/Parallelogram.h File Reference

```
#include "pch.h"
```

Classes

- class [myParallelogram::Parallelogram](#)
Parallelogram class, which inherits from the *IShape* interface and stores information about a parallelogram shape.

Namespaces

- namespace [myParallelogram](#)

9.83 Parallelogram.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include "pch.h"
00004
00005 extern "C" {
00006     namespace myParallelogram {
00010         class Parallelogram :
00011             public IShape
00012         {
00014             double _side;
00015
00017             double _base;
00018
00020             double _height;
00021
00022         public:
00029             Parallelogram(double, double, double) noexcept(false);
00030
00035             double area() override;
00036
00041             double perimeter() override;
00042
00047             string toString() override;
00048
00053             double side();
00054
00059             double base();
00060
00065             double height();
00066         };
00067     };
00068 }
```

9.84 Parallelogram/ParallelogramParser.cpp File Reference

```

#include "pch.h"
#include "ParallelogramParser.h"
```

9.85 Parallelogram/ParallelogramParser.h File Reference

```

#include "pch.h"
#include "Parallelogram.h"
```

Classes

- class [ParallelogramParser](#)

[ParallelogramParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing parallelogram shapes.

9.86 ParallelogramParser.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Parallelogram.h"
00005
00006 extern "C" {
00010     class ParallelogramParser :
00011     public IParser
00012     {
00013     private:
00015         inline static ParallelogramParser* _instance = nullptr;
00016
00020         ParallelogramParser() = default;
00021
00025         ~ParallelogramParser() = default;
00026
00030         ParallelogramParser(const ParallelogramParser&) = delete;
00031
00035         ParallelogramParser& operator=(const ParallelogramParser&) = delete;
00036     public:
00041         static ParallelogramParser* getInstance();
00042
00049         IShape* parse(stringstream data) noexcept(false) override;
00050
00055         string toString() override;
00056     };
00057 }
```

9.87 Parallelogram/ParallelogramToStringConverter.cpp File Reference

```
#include "pch.h"
#include "ParallelogramToStringConverter.h"
```

9.88 Parallelogram/ParallelogramToStringConverter.h File Reference

```
#include "pch.h"
#include "Parallelogram.h"
```

Classes

- class [ParallelogramToStringConverter](#)

[ParallelogramToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting parallelogram shape information to data set.

9.89 ParallelogramToStringConverter.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Parallelogram.h"
00005
00006 extern "C" {
00010     class ParallelogramToStringConverter :
00011     public IShapeToStringConverter
00012     {
00013     public:
00019         SHAPE_DATA convert(IShape*) override;
00020
00025         string toString() override;
00026     };
00027 }
```

9.90 README.md File Reference

9.91 Rectangle/Rectangle.cpp File Reference

```
#include "pch.h"
#include "Rectangle.h"
```

9.92 Rectangle/Rectangle.h File Reference

```
#include "pch.h"
```

Classes

- class [myRectangle::Rectangle](#)
[Rectangle](#) class, which inherits from the [IShape](#) interface and stores information about a rectangle shape.

Namespaces

- namespace [myRectangle](#)

9.93 Rectangle.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004
00005 extern "C" {
00006     namespace myRectangle {
00010         class Rectangle :
00011             public IShape
00012         {
00014             double _width;
00015
00017             double _height;
00018
00019         public:
00025             Rectangle(double, double) noexcept(false);
00026
00031             double area() override;
00032
00037             double perimeter() override;
00038
00043             string toString() override;
00044
00049             double width();
00050
00055             double height();
00056         };
00057     };
00058 }
```

9.94 Rectangle/RectangleParser.cpp File Reference

```
#include "pch.h"
#include "RectangleParser.h"
```

9.95 Rectangle/RectangleParser.h File Reference

```
#include "pch.h"
#include "Rectangle.h"
```

Classes

- class [RectangleParser](#)

[RectangleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing rectangle shapes.

9.96 RectangleParser.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Rectangle.h"
00005
00006 extern "C" {
00010     class RectangleParser :
00011     public IParser
00012     {
00013     private:
00015         inline static RectangleParser* _instance = nullptr;
00016
00020         RectangleParser() = default;
00021
00025         ~RectangleParser() = default;
00026
00030         RectangleParser(const RectangleParser&) = delete;
00031
00035         RectangleParser& operator=(const RectangleParser&) = delete;
00036     public:
00041         static RectangleParser* getInstance();
00042
00049         IShape* parse(stringstream data) noexcept(false) override;
00050
00055         string toString() override;
00056     };
00057 }
```

9.97 Rectangle/RectangleToStringConverter.cpp File Reference

```
#include "pch.h"
#include "RectangleToStringConverter.h"
```

9.98 Rectangle/RectangleToStringConverter.h File Reference

```
#include "pch.h"
#include "Rectangle.h"
```

Classes

- class [RectangleToStringConverter](#)

[RectangleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting rectangle shape information to data set.

9.99 RectangleToStringConverter.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Rectangle.h"
00005
00006 extern "C" {
00010     class RectangleToStringConverter :
00011     public IShapeToStringConverter
00012     {
00013     public:
00019         SHAPE_DATA convert(IShape*) override;
00020
00025         string toString() override;
00026     };
00027 }
```

9.100 Rhombus/Rhombus.cpp File Reference

```
#include "pch.h"
#include "Rhombus.h"
```

9.101 Rhombus/Rhombus.h File Reference

```
#include "pch.h"
```

Classes

- class [myRhombus::Rhombus](#)

[Rhombus](#) class, which inherits from the [IShape](#) interface and stores information about a rhombus shape.

Namespaces

- namespace [myRhombus](#)

9.102 Rhombus.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004
00005 extern "C" {
00006     namespace myRhombus {
00010         class Rhombus :
00011             public IShape
00012         {
00014             double _short_diagonal;
00015
00017             double _long_diagonal;
00018
00019         public:
00025             Rhombus(double, double) noexcept(false);
00026
00031             double area() override;
00032
00037             double perimeter() override;
00038
00043             string toString() override;
00044
00049             double short_diagonal();
00050
00055             double long_diagonal();
00056         };
00057     };
00058 }
```

9.103 Rhombus/RhombusParser.cpp File Reference

```
#include "pch.h"
#include "RhombusParser.h"
```

9.104 Rhombus/RhombusParser.h File Reference

```
#include "pch.h"
#include "Rhombus.h"
```

Classes

- class [RhombusParser](#)

[RhombusParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing rhombus shapes.

9.105 RhombusParser.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Rhombus.h"
00005
00006 extern "C" {
00010     class RhombusParser :
00011         public IParser
```

```

00012     {
00013     private:
00015         inline static RhombusParser* _instance = nullptr;
00016
00020         RhombusParser() = default;
00021
00025         ~RhombusParser() = default;
00026
00030         RhombusParser(const RhombusParser&) = delete;
00031
00035         RhombusParser& operator=(const RhombusParser&) = delete;
00036     public:
00041         static RhombusParser* getInstance();
00042
00049         IShape* parse(stringstream data) noexcept(false) override;
00050
00055         string toString() override;
00056     };
00057 }

```

9.106 Rhombus/RhombusToStringConverter.cpp File Reference

```

#include "pch.h"
#include "RhombusToStringConverter.h"

```

9.107 Rhombus/RhombusToStringConverter.h File Reference

```

#include "pch.h"
#include "Rhombus.h"

```

Classes

- class [RhombusToStringConverter](#)

[RhombusToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting rhombus shape information to data set.

9.108 RhombusToStringConverter.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Rhombus.h"
00005
00006 extern "C" {
00010     class RhombusToStringConverter :
00011     public IShapeToStringConverter
00012     {
00013     public:
00019         SHAPE_DATA convert(IShape*) override;
00020
00025         string toString() override;
00026     };
00027 }

```

9.109 ShapesParser/ConverterFactory.cpp File Reference

```
#include "ConverterFactory.h"
```

9.110 ShapesParser/ConverterFactory.h File Reference

```
#include "IShape.h"
#include "IShapeToStringConverter.h"
#include "../utils/utils.h"
#include "Object.h"
```

Classes

- class [ConverterFactory](#)

Class to manage a list of prototypes for [IShapeToStringConverter](#) objects.

9.111 ConverterFactory.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "IShape.h"
00003 #include "IShapeToStringConverter.h"
00004 #include "../utils/utils.h"
00005 #include "Object.h"
00006
00010 class ConverterFactory : public Object
00011 {
00012     map<string, IShapeToStringConverter*> _prototypes;
00013 public:
00020     void registerWith(string type, IShapeToStringConverter* parser);
00021
00028     IShapeToStringConverter* select(string type);
00029
00035     string toString() override;
00036 };
00037
00038
```

9.112 ShapesParser/IParser.cpp File Reference

```
#include "IParser.h"
```

9.113 ShapesParser/IParser.h File Reference

```
#include "IShape.h"
#include "Object.h"
#include "../utils/utils.h"
```


Classes

- class [IParser](#)

IParser interface is used for declare methods for subclasses to implement.

9.114 IParser.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "IShape.h"
00003 #include "Object.h"
00004 #include "../utils/utils.h"
00005
00009 class IParser : public Object
00010 {
00011 public:
00017     virtual IShape* parse(stringstream data) noexcept(false) = 0;
00018 };
00019
```

9.115 ShapesParser/IShape.cpp File Reference

```
#include "IShape.h"
```

9.116 ShapesParser/IShape.h File Reference

```
#include "Object.h"
#include "../utils/utils.h"
```

Classes

- class [IShape](#)

IShape interface is used for declare methods for subclasses to implement.

9.117 IShape.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "Object.h"
00003 #include "../utils/utils.h"
00004
00008 class IShape : public Object
00009 {
00010 public:
00015     virtual double area() = 0;
00016
00021     virtual double perimeter() = 0;
00022 };
00023
```

9.118 ShapesParser/IShapeToStringConverter.cpp File Reference

```
#include "IShapeToStringConverter.h"
```

9.119 ShapesParser/IShapeToStringConverter.h File Reference

```
#include "IShape.h"
#include "Object.h"
#include "../utils/utils.h"
```

Classes

- class [IShapeToStringConverter](#)
IShapeToStringConverter interface is used for declare methods for subclasses to implement.

9.120 IShapeToStringConverter.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "IShape.h"
00003 #include "Object.h"
00004 #include "../utils/utils.h"
00005
00009 class IShapeToStringConverter : public Object
00010 {
00011 public:
00017     virtual SHAPE_DATA convert(IShape* shape) = 0;
00018 };
00019
```

9.121 ShapesParser/IShowDataBehavior.cpp File Reference

```
#include "IShowDataBehavior.h"
```

9.122 ShapesParser/IShowDataBehavior.h File Reference

```
#include "../utils/utils.h"
#include "Object.h"
```

Classes

- class [IShowDataBehavior](#)
IShowDataBehavior interface is used for declare methods for subclasses to implement.

9.123 IShowDataBehavior.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "../utils/utils.h"
00003 #include "Object.h"
00004
00008 class IShowDataBehavior : public Object
00009 {
00010 public:
00015     virtual void showData(vector<SHAPE_DATA> data) = 0;
00016 };
00017
```

9.124 ShapesParser/IShowTableBehavior.cpp File Reference

```
#include "IShowTableBehavior.h"
```

9.125 ShapesParser/IShowTableBehavior.h File Reference

```
#include "../utils/utils.h"
#include "Object.h"
```

Classes

- class [IShowTableBehavior](#)

IShowTableBehavior interface is used for declare methods for subclasses to implement.

9.126 IShowTableBehavior.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "../utils/utils.h"
00003 #include "Object.h"
00004
00008 class IShowTableBehavior : public Object
00009 {
00010 public:
00015     virtual void showTable(vector<SHAPE_DATA>) = 0;
00016 };
00017
```

9.127 ShapesParser/main.cpp File Reference

```
#include "IShape.h"
#include "IParser.h"
#include "IShapeToStringConverter.h"
#include "IShowTableBehavior.h"
#include "IShowDataBehavior.h"
#include "ParserFactory.h"
#include "ConverterFactory.h"
#include "ShapesPrinter.h"
#include "ShowTableCustom.h"
#include "ShowDataCustom.h"
```

Functions

- void `setMode ()`
Function to set Vietnamese character mode for console output.
- void `readFile` (wstring `textFile`, int &`count`, vector< shared_ptr< [IShape](#) > > &`shapes`, [ParserFactory](#) &`parser_factory`)
Function to read text file and store shape objects in a vector.
- void `sortWithLambdaExpression` (vector< shared_ptr< [IShape](#) > > &`shapes`)
Function to sort shape objects in ascending order of area.
- void `loadShapesToPrinter` ([ShapesPrinter](#) &`printer`, vector< shared_ptr< [IShape](#) > > &`shapes`, [ConverterFactory](#) &`converter_factory`)
Function to load [IShape](#) objects into a [ShapesPrinter](#) for printing.
- void `setCustomPrinter` ([ShapesPrinter](#) &`printer`, [IShowDataBehavior](#) *&`showDataBehavior`, [IShowTableBehavior](#) *&`showTableBehavior`)
Function to set custom printing behavior for the [ShapesPrinter](#) object.
- void `printToScreen` ([ShapesPrinter](#) &`printer`, vector< shared_ptr< [IShape](#) > > &`shapes`, int `count`)
Function to print shape objects to the console.
- int `main ()`

9.127.1 Function Documentation

9.127.1.1 loadShapesToPrinter()

```
void loadShapesToPrinter (
    ShapesPrinter & printer,
    vector< shared_ptr< IShape > > & shapes,
    ConverterFactory & converter_factory )
```

Function to load [IShape](#) objects into a [ShapesPrinter](#) for printing.

Parameters

<i>printer</i>	ShapesPrinter object that handles printing
<i>shapes</i>	Vector of shape objects to be printed
<i>converter_factory</i>	ConverterFactory to select object instantiation method through conversion

9.127.1.2 main()

```
int main ( )
```

9.127.1.3 printToScreen()

```
void printToScreen (
    ShapesPrinter & printer,
    vector< shared_ptr< IShape > > & shapes,
    int count )
```

Function to print shape objects to the console.

Parameters

<i>printer</i>	ShapesPrinter object that handles printing
<i>shapes</i>	Vector of shape objects to be printed
<i>count</i>	Number of shape objects declared in the file

9.127.1.4 readFile()

```
void readFile (
    wstring textFile,
    int & count,
    vector< shared_ptr< IShape > > & shapes,
    ParserFactory & parser_factory )
```

Function to read text file and store shape objects in a vector.

Parameters

<i>textFile</i>	Name of the text file
<i>count</i>	Number of shape objects declared in the file
<i>shapes</i>	Vector to store shape objects
<i>parser_factory</i>	Parser factory to select object instantiation method through parsing

9.127.1.5 setCustomPrinter()

```
void setCustomPrinter (
    ShapesPrinter & printer,
    IShowDataBehavior *& showDataBehavior,
    IShowTableBehavior *& showTableBehavior )
```

Function to set custom printing behavior for the [ShapesPrinter](#) object.

Parameters

<i>printer</i>	ShapesPrinter object that handles printing
<i>showDataBehavior</i>	Behavior for printing data
<i>showTableBehavior</i>	Behavior for printing table

9.127.1.6 setMode()

```
void setMode ( )
```

Function to set Vietnamese character mode for console output.

9.127.1.7 `sortWithLambdaExpression()`

```
void sortWithLambdaExpression (
    vector< shared_ptr< IShape > > & shapes )
```

Function to sort shape objects in ascending order of area.

Parameters

<code>shapes</code>	Vector of shape objects to be sorted
---------------------	--------------------------------------

9.128 ShapesParser/Object.cpp File Reference

```
#include "Object.h"
```

9.129 ShapesParser/Object.h File Reference

```
#include "../utils/utils.h"
```

Classes

- class [Object](#)
[Object](#) class is the largest superclass of all classes in the program.

9.130 Object.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "../utils/utils.h"
00003
00007 class Object
00008 {
00009 public:
00014     virtual string toString() = 0;
00015 };
```

9.131 ShapesParser/ParserFactory.cpp File Reference

```
#include "ParserFactory.h"
```

9.132 ShapesParser/ParserFactory.h File Reference

```
#include "IShape.h"
#include "IParser.h"
#include "../utils/utils.h"
#include "Object.h"
```

Classes

- class [ParserFactory](#)
Class to manage a list of prototypes for [IParser](#) objects.

9.133 ParserFactory.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "IShape.h"
00003 #include "IParser.h"
00004 #include "../utils/utils.h"
00005 #include "Object.h"
00006
00010 class ParserFactory : public Object
00011 {
00012     map<string, IParser*> _prototypes;
00013 public:
00020     void registerWith(string type, IParser* parser);
00021
00028     IParser* select(string type);
00029
00035     string toString() override;
00036 };
00037
00038
```

9.134 ShapesParser/ShapesPrinter.cpp File Reference

```
#include "ShapesPrinter.h"
```

9.135 ShapesParser/ShapesPrinter.h File Reference

```
#include "../utils/utils.h"
#include "../ShapesParser/IShape.h"
#include "IShowTableBehavior.h"
#include "IShowDataBehavior.h"
#include "ShowTableDefault.h"
#include "ShowDataDefault.h"
#include "Object.h"
```

Classes

- class [ShapesPrinter](#)
[ShapesPrinter](#) class, responsible for printing shapes to the screen.

9.136 ShapesPrinter.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "../utils/utils.h"
00004 #include "../ShapesParser/IShape.h"
00005 #include "IShowTableBehavior.h"
00006 #include "IShowDataBehavior.h"
00007 #include "ShowTableDefault.h"
00008 #include "ShowDataDefault.h"
00009 #include "Object.h"
00010
00014 class ShapesPrinter : public Object
00015 {
00016 private:
00018     vector<SHAPE_DATA> _data;
00019
00021     IShowTableBehavior* _showTableBehavior;
00022
00024     IShowDataBehavior* _showDataBehavior;
00025 public:
00029     ShapesPrinter();
00030
00035     void setShowDataBehavior(IShowDataBehavior*);
00036
00041     void performShowDataBehavior(vector<SHAPE_DATA>);
00042
00047     void setShowTableBehavior(IShowTableBehavior*);
00048
00053     void performShowTableBehavior(vector<SHAPE_DATA>);
00054
00059     void push(SHAPE_DATA);
00060
00065     void clear();
00066
00071     vector<SHAPE_DATA> getData();
00072
00077     string toString() override;
00078 };
```

9.137 ShapesParser/ShowDataCustom.cpp File Reference

```
#include "ShowDataCustom.h"
```

9.138 ShapesParser/ShowDataCustom.h File Reference

```
#include "IShowDataBehavior.h"
#include "../utils/utils.h"
```

Classes

- class [ShowDataCustom](#)

Custom implementation of [IShowDataBehavior](#), responsible for displaying shape data in a customized format.

9.139 ShowDataCustom.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "IShowDataBehavior.h"
00004 #include "../utils/utils.h"
00005
00009 class ShowDataCustom : public IShowDataBehavior
00010 {
00011 public:
00016     void showData(vector<SHAPE_DATA>);
00017
00022     string toString() override;
00023 };
00024
```

9.140 ShapesParser/ShowDataDefault.cpp File Reference

```
#include "ShowDataDefault.h"
```

9.141 ShapesParser/ShowDataDefault.h File Reference

```
#include "IShowDataBehavior.h"
#include "../utils/utils.h"
```

Classes

- class [ShowDataDefault](#)

Default implementation of [IShowDataBehavior](#), responsible for displaying shape data in a default format.

9.142 ShowDataDefault.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "IShowDataBehavior.h"
00004 #include "../utils/utils.h"
00005
00009 class ShowDataDefault : public IShowDataBehavior
00010 {
00011 public:
00016     void showData(vector<SHAPE_DATA>);
00017
00022     string toString() override;
00023 };
00024
```

9.143 ShapesParser/ShowTableCustom.cpp File Reference

```
#include "ShowTableCustom.h"
```

9.144 ShapesParser/ShowTableCustom.h File Reference

```
#include "IShowTableBehavior.h"
#include "../utils/utils.h"
```

Classes

- class [ShowTableCustom](#)

Custom implementation of [IShowTableBehavior](#), responsible for displaying shape data in a customized table format.

9.145 ShowTableCustom.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "IShowTableBehavior.h"
00004 #include "../utils/utils.h"
00005
00009 class ShowTableCustom : public IShowTableBehavior
00010 {
00011 public:
00016     void showTable(vector<SHAPE_DATA>);
00017
00022     string toString() override;
00023 };
00024
```

9.146 ShapesParser/ShowTableDefault.cpp File Reference

```
#include "ShowTableDefault.h"
```

9.147 ShapesParser/ShowTableDefault.h File Reference

```
#include "IShowTableBehavior.h"
#include "../utils/utils.h"
```

Classes

- class [ShowTableDefault](#)

Default implementation of [IShowTableBehavior](#), responsible for displaying shape data in a default table format.

9.148 ShowTableDefault.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002 #include "IShowTableBehavior.h"
00003 #include "../utils/utils.h"
00004
00005
00006 #include "IShowTableBehavior.h"
00007
00011 class ShowTableDefault : public IShowTableBehavior
00012 {
00013 public:
00018     void showTable(vector<SHAPE_DATA>);
00019
00024     string toString() override;
00025 };
00026
```

9.149 ShapesParser/Strategy.cpp File Reference

```
#include "Strategy.h"
```

9.150 Square/Square.cpp File Reference

```
#include "pch.h"
#include "Square.h"
```

9.151 Square/Square.h File Reference

```
#include "pch.h"
```

Classes

- class [mySquare::Square](#)
Square class, which inherits from the [IShape](#) interface and stores information about a square shape.

Namespaces

- namespace [mySquare](#)

9.152 Square.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004
00005 extern "C" {
00006     namespace mySquare {
00010         class Square :
00011             public IShape
00012         {
00014             double _length;
00015
00016         public:
00021             Square(double) noexcept(false);
00022
00027             double area() override;
00028
00033             double perimeter() override;
00034
00039             std::string toString() override;
00040
00045             double length();
00046         };
00047     };
00048 }
```

9.153 Square/SquareParser.cpp File Reference

```
#include "pch.h"
#include "SquareParser.h"
```

9.154 Square/SquareParser.h File Reference

```
#include "pch.h"
#include "Square.h"
```

Classes

- class [SquareParser](#)

[SquareParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing square shapes.

9.155 SquareParser.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Square.h"
00005
00006 extern "C" {
00010     class SquareParser :
00011         public IParser
00012     {
00013     private:
00015         inline static SquareParser* _instance = nullptr;
00016     }
```

```

00020         SquareParser() = default;
00021
00025         ~SquareParser() = default;
00026
00030         SquareParser(const SquareParser&) = delete;
00031
00035         SquareParser& operator=(const SquareParser&) = delete;
00036     public:
00041         static SquareParser* getInstance();
00042
00049         IShape* parse(stringstream data) noexcept(false) override;
00050
00055         string toString() override;
00056     };
00057 }

```

9.156 Square/SquareToStringConverter.cpp File Reference

```

#include "pch.h"
#include "SquareToStringConverter.h"

```

9.157 Square/SquareToStringConverter.h File Reference

```

#include "pch.h"
#include "Square.h"

```

Classes

- class [SquareToStringConverter](#)

[SquareToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting square shape information to data set.

9.158 SquareToStringConverter.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Square.h"
00005
00006 extern "C" {
00010     class SquareToStringConverter :
00011     public IShapeToStringConverter
00012     {
00013     public:
00019         SHAPE_DATA convert(IShape*) override;
00020
00025         string toString() override;
00026     };
00027 }

```

9.159 Triangle/Triangle.cpp File Reference

```

#include "pch.h"
#include "Triangle.h"

```

9.160 Triangle/Triangle.h File Reference

```
#include "pch.h"
```

Classes

- class [myTriangle::Triangle](#)
Triangle class, which inherits from the *IShape* interface and stores information about a triangle shape.

Namespaces

- namespace [myTriangle](#)

9.161 Triangle.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004
00005 extern "C" {
00006     namespace myTriangle {
00010         class Triangle :
00011             public IShape
00012         {
00014             double _first_edge;
00015
00017             double _second_edge;
00018
00020             double _third_edge;
00021
00022         public:
00029             Triangle(double, double, double) noexcept(false);
00030
00035             double area() override;
00036
00041             double perimeter() override;
00042
00047             string toString() override;
00048
00053             double first_edge();
00054
00059             double second_edge();
00060
00065             double third_edge();
00066         };
00067     };
00068 }
```

9.162 Triangle/TriangleParser.cpp File Reference

```
#include "pch.h"
#include "TriangleParser.h"
```

9.163 Triangle/TriangleParser.h File Reference

```
#include "pch.h"
#include "Triangle.h"
```

Classes

- class [TriangleParser](#)

[TriangleParser](#) class, which inherits from the [IParser](#) interface and performs the task of parsing triangle shapes.

9.164 TriangleParser.h

[Go to the documentation of this file.](#)

```

00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Triangle.h"
00005
00006 extern "C" {
00010     class TriangleParser :
00011     public IParser
00012     {
00013     private:
00015         inline static TriangleParser* _instance = nullptr;
00016
00020         TriangleParser() = default;
00021
00025         ~TriangleParser() = default;
00026
00030         TriangleParser(const TriangleParser&) = delete;
00031
00035         TriangleParser& operator=(const TriangleParser&) = delete;
00036     public:
00041         static TriangleParser* getInstance();
00042
00049         IShape* parse(stringstream data) noexcept(false) override;
00050
00055         string toString() override;
00056     };
00057 }
```

9.165 Triangle/TriangleToStringConverter.cpp File Reference

```

#include "pch.h"
#include "TriangleToStringConverter.h"
```

9.166 Triangle/TriangleToStringConverter.h File Reference

```

#include "pch.h"
#include "Triangle.h"
```

Classes

- class [TriangleToStringConverter](#)

[TriangleToStringConverter](#) class, which inherits from the [IShapeToStringConverter](#) interface and performs the task of converting triangle shape information to data set.

9.167 TriangleToStringConverter.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "Triangle.h"
00005
00006 extern "C" {
00010     class TriangleToStringConverter :
00011     public IShapeToStringConverter
00012     {
00013     public:
00019         SHAPE_DATA convert(IShape*) override;
00020
00025         string toString() override;
00026     };
00027 }
```

9.168 utils/utils.cpp File Reference

```
#include "pch.h"
#include "framework.h"
#include "utils.h"
```

9.169 utils/utils.h File Reference

```
#include "pch.h"
#include "framework.h"
#include <iostream>
#include <string>
#include <sstream>
#include <regex>
#include <vector>
#include <fstream>
#include <windows.h>
#include <iomanip>
#include <map>
#include <tuple>
#include <io.h>
#include <filesystem>
#include <fcntl.h>
#include <exception>
#include <memory>
#include <algorithm>
#include <cmath>
```

Typedefs

- typedef tuple< wstring, wstring, wstring, wstring > [SHAPE_DATA](#)
SHAPE_DATA is the data type defined to store the information of the [IShape](#) object:

Functions

- const regex `DOUBLE_PATTERN` (`"[+-]?([0-9]+([.][0-9]*)?|[.][0-9]+)"`)
DOUBLE_PATTERN is a regular expression that determines whether a string is a double value or not.

Variables

- const double `PI` = 3.1415

9.169.1 Typedef Documentation

9.169.1.1 SHAPE_DATA

```
typedef tuple<wstring, wstring, wstring, wstring> SHAPE_DATA
```

`SHAPE_DATA` is the data type defined to store the information of the `IShape` object:

- name /n
- attributes /n
- perimeter /n
- area

9.169.2 Function Documentation

9.169.2.1 DOUBLE_PATTERN()

```
const regex DOUBLE_PATTERN (  
    "[+-]?([0-9]+([.][0-9]*)?|[.][0-9]+)" [+-] )
```

`DOUBLE_PATTERN` is a regular expression that determines whether a string is a double value or not.

9.169.3 Variable Documentation

9.169.3.1 PI

```
const double PI = 3.1415
```

9.170 utils.h

[Go to the documentation of this file.](#)

```
00001 #pragma once
00002
00003 #include "pch.h"
00004 #include "framework.h"
00005
00006 #include <iostream>
00007 #include <string>
00008 #include <sstream>
00009 #include <regex>
00010 #include <vector>
00011 #include <fstream>
00012 #include <windows.h>
00013 #include <iomanip>
00014 #include <map>
00015 #include <tuple>
00016 #include <io.h>
00017 #include <filesystem>
00018 #include <fcntl.h>
00019 #include <exception>
00020 #include <memory>
00021 #include <algorithm>
00022 #include <cmath>
00023
00024 using std::cout, std::cin, std::endl;
00025 using std::string, std::stringstream, std::wstring, std::wstringstream;
00026 using std::wcout;
00027 using std::vector;
00028 using std::fstream;
00029 using std::map, std::tuple;
00030 using std::ifstream, std::ofstream;
00031 using std::setw, std::fixed, std::setprecision, std::left;
00032 using std::regex, std::regex_match;
00033 using std::exception;
00034 using std::to_wstring;
00035 using std::sort;
00036 using std::unique_ptr, std::shared_ptr, std::make_unique, std::make_shared;
00037
00045 typedef tuple<wstring, wstring, wstring, wstring> SHAPE_DATA;
00046
00047 const double PI = 3.1415;
00048
00052 const regex DOUBLE_PATTERN("[+-]?([0-9]+([.][0-9]*)?|[.][0-9]+)");
```

Index

- `__declspec`
 - `dllmain.cpp`, 101–105
 - `_base`
 - `myIsoscelesTrapezoid::IsoscelesTrapezoid`, 44
 - `myParallelogram::Parallelogram`, 53
 - `_data`
 - `ShapesPrinter`, 77
 - `_first_edge`
 - `myTriangle::Triangle`, 92
 - `_height`
 - `myIsoscelesTrapezoid::IsoscelesTrapezoid`, 44
 - `myParallelogram::Parallelogram`, 53
 - `myRectangle::Rectangle`, 62
 - `_instance`
 - `CircleParser`, 24
 - `EllipseParser`, 33
 - `IsoscelesTrapezoidParser`, 47
 - `ParallelogramParser`, 56
 - `RectangleParser`, 65
 - `RhombusParser`, 72
 - `SquareParser`, 88
 - `TriangleParser`, 96
 - `_length`
 - `mySquare::Square`, 85
 - `_long_diagonal`
 - `myRhombus::Rhombus`, 69
 - `_prototypes`
 - `ConverterFactory`, 27
 - `ParserFactory`, 59
 - `_radius`
 - `myCircle::Circle`, 21
 - `_second_edge`
 - `myTriangle::Triangle`, 92
 - `_semi_major_axis`
 - `myEllipse::Ellipse`, 30
 - `_semi_minor_axis`
 - `myEllipse::Ellipse`, 30
 - `_short_diagonal`
 - `myRhombus::Rhombus`, 69
 - `_showDataBehavior`
 - `ShapesPrinter`, 77
 - `_showTableBehavior`
 - `ShapesPrinter`, 77
 - `_side`
 - `myParallelogram::Parallelogram`, 53
 - `_third_edge`
 - `myTriangle::Triangle`, 92
 - `_top`
 - `myIsoscelesTrapezoid::IsoscelesTrapezoid`, 44
 - `_width`
 - `myRectangle::Rectangle`, 62
 - `~CircleParser`
 - `CircleParser`, 22
 - `~EllipseParser`
 - `EllipseParser`, 31
 - `~IsoscelesTrapezoidParser`
 - `IsoscelesTrapezoidParser`, 45
 - `~ParallelogramParser`
 - `ParallelogramParser`, 54
 - `~RectangleParser`
 - `RectangleParser`, 63
 - `~RhombusParser`
 - `RhombusParser`, 71
 - `~SquareParser`
 - `SquareParser`, 86
 - `~TriangleParser`
 - `TriangleParser`, 94
- `area`
 - `IShape`, 37
 - `myCircle::Circle`, 20
 - `myEllipse::Ellipse`, 29
 - `myIsoscelesTrapezoid::IsoscelesTrapezoid`, 42
 - `myParallelogram::Parallelogram`, 51
 - `myRectangle::Rectangle`, 61
 - `myRhombus::Rhombus`, 68
 - `mySquare::Square`, 84
 - `myTriangle::Triangle`, 91
- `base`
 - `myIsoscelesTrapezoid::IsoscelesTrapezoid`, 42
 - `myParallelogram::Parallelogram`, 51
- `Circle`
 - `myCircle::Circle`, 20
- `Circle/Circle.cpp`, 99
- `Circle/Circle.h`, 99
- `Circle/CircleParser.cpp`, 100
- `Circle/CircleParser.h`, 100
- `Circle/CircleToStringConverter.cpp`, 100
- `Circle/CircleToStringConverter.h`, 101
- `Circle/dllmain.cpp`, 101
- `Circle/framework.h`, 105
- `Circle/pch.cpp`, 110
- `Circle/pch.h`, 111
- `CircleParser`, 21
 - `_instance`, 24
 - `~CircleParser`, 22
 - `CircleParser`, 22, 23

- getInstance, 23
 - operator=, 23
 - parse, 23
 - toString, 24
- CircleToStringConverter, 24
 - convert, 25
 - toString, 25
- clear
 - ShapesPrinter, 75
- convert
 - CircleToStringConverter, 25
 - EllipseToStringConverter, 34
 - IShapeToStringConverter, 38
 - IsoscelesTrapezoidToStringConverter, 48
 - ParallelogramToStringConverter, 57
 - RectangleToStringConverter, 66
 - RhombusToStringConverter, 73
 - SquareToStringConverter, 89
 - TriangleToStringConverter, 97
- ConverterFactory, 26
 - _prototypes, 27
 - registerWith, 26
 - select, 27
 - toString, 27
- dllmain.cpp
 - _declspec, 101–105
- DOUBLE_PATTERN
 - utils.h, 145
- Ellipse
 - myEllipse::Ellipse, 28
- Ellipse/dllmain.cpp, 102
- Ellipse/Ellipse.cpp, 115
- Ellipse/Ellipse.h, 115, 116
- Ellipse/EllipseParser.cpp, 116
- Ellipse/EllipseParser.h, 116, 117
- Ellipse/EllipseToStringConverter.cpp, 117
- Ellipse/EllipseToStringConverter.h, 117
- Ellipse/framework.h, 106
- Ellipse/pch.cpp, 110
- Ellipse/pch.h, 111
- EllipseParser, 30
 - _instance, 33
 - ~EllipseParser, 31
 - EllipseParser, 31, 32
 - getInstance, 32
 - operator=, 32
 - parse, 32
 - toString, 33
- EllipseToStringConverter, 33
 - convert, 34
 - toString, 34
- first_edge
 - myTriangle::Triangle, 91
- framework.h
 - WIN32_LEAN_AND_MEAN, 105–109
- getData
 - ShapesPrinter, 75
- getInstance
 - CircleParser, 23
 - EllipseParser, 32
 - IsoscelesTrapezoidParser, 46
 - ParallelogramParser, 55
 - RectangleParser, 64
 - RhombusParser, 71
 - SquareParser, 87
 - TriangleParser, 95
- height
 - myIsoscelesTrapezoid::IsoscelesTrapezoid, 43
 - myParallelogram::Parallelogram, 52
 - myRectangle::Rectangle, 61
- IParser, 35
 - parse, 35
- IShape, 36
 - area, 37
 - perimeter, 37
- IShapeToStringConverter, 38
 - convert, 38
- IShowDataBehavior, 39
 - showData, 39
- IShowTableBehavior, 40
 - showTable, 40
- IsoscelesTrapezoid
 - myIsoscelesTrapezoid::IsoscelesTrapezoid, 42
- IsoscelesTrapezoid/dllmain.cpp, 102
- IsoscelesTrapezoid/framework.h, 106
- IsoscelesTrapezoid/IsoscelesTrapezoid.cpp, 118
- IsoscelesTrapezoid/IsoscelesTrapezoid.h, 118
- IsoscelesTrapezoid/IsoscelesTrapezoidParser.cpp, 119
- IsoscelesTrapezoid/IsoscelesTrapezoidParser.h, 119
- IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.cpp, 119
- IsoscelesTrapezoid/IsoscelesTrapezoidToStringConverter.h, 120
- IsoscelesTrapezoid/pch.cpp, 110
- IsoscelesTrapezoid/pch.h, 112
- IsoscelesTrapezoidParser, 44
 - _instance, 47
 - ~IsoscelesTrapezoidParser, 45
 - getInstance, 46
 - IsoscelesTrapezoidParser, 45, 46
 - operator=, 46
 - parse, 46
 - toString, 47
- IsoscelesTrapezoidToStringConverter, 47
 - convert, 48
 - toString, 48
- length
 - mySquare::Square, 84
- loadShapesToPrinter
 - main.cpp, 132
- long_diagonal

- myRhombus::Rhombus, 68
- main
 - main.cpp, 132
- main.cpp
 - loadShapesToPrinter, 132
 - main, 132
 - printToScreen, 132
 - readFile, 133
 - setCustomPrinter, 133
 - setMode, 133
 - sortWithLambdaExpression, 133
- myCircle, 17
- myCircle::Circle, 19
 - _radius, 21
 - area, 20
 - Circle, 20
 - perimeter, 20
 - radius, 20
 - toString, 21
- myEllipse, 17
- myEllipse::Ellipse, 27
 - _semi_major_axis, 30
 - _semi_minor_axis, 30
 - area, 29
 - Ellipse, 28
 - perimeter, 29
 - semi_major_axis, 29
 - semi_minor_axis, 29
 - toString, 29
- myIsoscelesTrapezoid, 17
- myIsoscelesTrapezoid::IsoscelesTrapezoid, 41
 - _base, 44
 - _height, 44
 - _top, 44
 - area, 42
 - base, 42
 - height, 43
 - IsoscelesTrapezoid, 42
 - perimeter, 43
 - top, 43
 - toString, 43
- myParallelogram, 17
- myParallelogram::Parallelogram, 50
 - _base, 53
 - _height, 53
 - _side, 53
 - area, 51
 - base, 51
 - height, 52
 - Parallelogram, 51
 - perimeter, 52
 - side, 52
 - toString, 52
- myRectangle, 18
- myRectangle::Rectangle, 59
 - _height, 62
 - _width, 62
 - area, 61
 - height, 61
 - perimeter, 61
 - Rectangle, 60
 - toString, 61
 - width, 61
- myRhombus, 18
- myRhombus::Rhombus, 67
 - _long_diagonal, 69
 - _short_diagonal, 69
 - area, 68
 - long_diagonal, 68
 - perimeter, 68
 - Rhombus, 68
 - short_diagonal, 68
 - toString, 69
- mySquare, 18
- mySquare::Square, 83
 - _length, 85
 - area, 84
 - length, 84
 - perimeter, 84
 - Square, 83
 - toString, 84
- myTriangle, 18
- myTriangle::Triangle, 89
 - _first_edge, 92
 - _second_edge, 92
 - _third_edge, 92
 - area, 91
 - first_edge, 91
 - perimeter, 91
 - second_edge, 91
 - third_edge, 92
 - toString, 92
 - Triangle, 90
- Object, 49
 - toString, 49
- operator=
 - CircleParser, 23
 - EllipseParser, 32
 - IsoscelesTrapezoidParser, 46
 - ParallelogramParser, 55
 - RectangleParser, 64
 - RhombusParser, 71
 - SquareParser, 87
 - TriangleParser, 95
- Parallelogram
 - myParallelogram::Parallelogram, 51
- Parallelogram/dllmain.cpp, 103
- Parallelogram/framework.h, 107
- Parallelogram/Parallelogram.cpp, 120
- Parallelogram/Parallelogram.h, 120, 121
- Parallelogram/ParallelogramParser.cpp, 121
- Parallelogram/ParallelogramParser.h, 121, 122
- Parallelogram/ParallelogramToStringConverter.cpp, 122
- Parallelogram/ParallelogramToStringConverter.h, 122
- Parallelogram/pch.cpp, 110

Parallelogram/pch.h, 112, 113
 ParallelogramParser, 53
 _instance, 56
 ~ParallelogramParser, 54
 getInstance, 55
 operator=, 55
 ParallelogramParser, 54, 55
 parse, 55
 toString, 56
 ParallelogramToStringConverter, 56
 convert, 57
 toString, 57
 parse
 CircleParser, 23
 EllipseParser, 32
 IParser, 35
 IsoscelesTrapezoidParser, 46
 ParallelogramParser, 55
 RectangleParser, 64
 RhombusParser, 71
 SquareParser, 87
 TriangleParser, 95
 ParserFactory, 58
 _prototypes, 59
 registerWith, 58
 select, 59
 toString, 59
 performShowDataBehavior
 ShapesPrinter, 75
 performShowTableBehavior
 ShapesPrinter, 75
 perimeter
 IShape, 37
 myCircle::Circle, 20
 myEllipse::Ellipse, 29
 myIsoscelesTrapezoid::IsoscelesTrapezoid, 43
 myParallelogram::Parallelogram, 52
 myRectangle::Rectangle, 61
 myRhombus::Rhombus, 68
 mySquare::Square, 84
 myTriangle::Triangle, 91
 PI
 utils.h, 145
 printToScreen
 main.cpp, 132
 push
 ShapesPrinter, 76
 radius
 myCircle::Circle, 20
 readFile
 main.cpp, 133
 README.md, 123
 Rectangle
 myRectangle::Rectangle, 60
 Rectangle/dllmain.cpp, 103
 Rectangle/framework.h, 107
 Rectangle/pch.cpp, 110
 Rectangle/pch.h, 113
 Rectangle/Rectangle.cpp, 123
 Rectangle/Rectangle.h, 123
 Rectangle/RectangleParser.cpp, 124
 Rectangle/RectangleParser.h, 124
 Rectangle/RectangleToStringConverter.cpp, 124
 Rectangle/RectangleToStringConverter.h, 125
 RectangleParser, 62
 _instance, 65
 ~RectangleParser, 63
 getInstance, 64
 operator=, 64
 parse, 64
 RectangleParser, 63, 64
 toString, 65
 RectangleToStringConverter, 65
 convert, 66
 toString, 66
 registerWith
 ConverterFactory, 26
 ParserFactory, 58
 Rhombus
 myRhombus::Rhombus, 68
 Rhombus/dllmain.cpp, 104
 Rhombus/framework.h, 108
 Rhombus/pch.cpp, 110
 Rhombus/pch.h, 113, 114
 Rhombus/Rhombus.cpp, 125
 Rhombus/Rhombus.h, 125, 126
 Rhombus/RhombusParser.cpp, 126
 Rhombus/RhombusParser.h, 126
 Rhombus/RhombusToStringConverter.cpp, 127
 Rhombus/RhombusToStringConverter.h, 127
 RhombusParser, 70
 _instance, 72
 ~RhombusParser, 71
 getInstance, 71
 operator=, 71
 parse, 71
 RhombusParser, 71
 toString, 72
 RhombusToStringConverter, 72
 convert, 73
 toString, 73
 second_edge
 myTriangle::Triangle, 91
 select
 ConverterFactory, 27
 ParserFactory, 59
 semi_major_axis
 myEllipse::Ellipse, 29
 semi_minor_axis
 myEllipse::Ellipse, 29
 setCustomPrinter
 main.cpp, 133
 setMode
 main.cpp, 133
 setShowDataBehavior
 ShapesPrinter, 76

- setShowTableBehavior
 - ShapesPrinter, 76
- shape, 3
- SHAPE_DATA
 - utils.h, 145
- ShapesParser/ConverterFactory.cpp, 128
- ShapesParser/ConverterFactory.h, 128
- ShapesParser/IParser.cpp, 128
- ShapesParser/IParser.h, 128, 129
- ShapesParser/IShape.cpp, 129
- ShapesParser/IShape.h, 129
- ShapesParser/IShapeToStringConverter.cpp, 130
- ShapesParser/IShapeToStringConverter.h, 130
- ShapesParser/IShowDataBehavior.cpp, 130
- ShapesParser/IShowDataBehavior.h, 130, 131
- ShapesParser/IShowTableBehavior.cpp, 131
- ShapesParser/IShowTableBehavior.h, 131
- ShapesParser/main.cpp, 131
- ShapesParser/Object.cpp, 134
- ShapesParser/Object.h, 134
- ShapesParser/ParserFactory.cpp, 134
- ShapesParser/ParserFactory.h, 135
- ShapesParser/ShapesPrinter.cpp, 135
- ShapesParser/ShapesPrinter.h, 135, 136
- ShapesParser/ShowDataCustom.cpp, 136
- ShapesParser/ShowDataCustom.h, 136, 137
- ShapesParser/ShowDataDefault.cpp, 137
- ShapesParser/ShowDataDefault.h, 137
- ShapesParser/ShowTableCustom.cpp, 137
- ShapesParser/ShowTableCustom.h, 138
- ShapesParser/ShowTableDefault.cpp, 138
- ShapesParser/ShowTableDefault.h, 138, 139
- ShapesParser/Strategy.cpp, 139
- ShapesPrinter, 74
 - _data, 77
 - _showDataBehavior, 77
 - _showTableBehavior, 77
 - clear, 75
 - getData, 75
 - performShowDataBehavior, 75
 - performShowTableBehavior, 75
 - push, 76
 - setShowDataBehavior, 76
 - setShowTableBehavior, 76
 - ShapesPrinter, 75
 - toString, 76
- short_diagonal
 - myRhombus::Rhombus, 68
- showData
 - IShowDataBehavior, 39
 - ShowDataCustom, 78
 - ShowDataDefault, 79
- ShowDataCustom, 77
 - showData, 78
 - toString, 78
- ShowDataDefault, 79
 - showData, 79
 - toString, 80
- showTable
 - IShowTableBehavior, 40
 - ShowTableCustom, 81
 - ShowTableDefault, 82
- ShowTableCustom, 80
 - showTable, 81
 - toString, 81
- ShowTableDefault, 81
 - showTable, 82
 - toString, 82
- side
 - myParallelogram::Parallelogram, 52
- sortWithLambdaExpression
 - main.cpp, 133
- Square
 - mySquare::Square, 83
- Square/dllmain.cpp, 104
- Square/framework.h, 108
- Square/pch.cpp, 110
- Square/pch.h, 114
- Square/Square.cpp, 139
- Square/Square.h, 139, 140
- Square/SquareParser.cpp, 140
- Square/SquareParser.h, 140
- Square/SquareToStringConverter.cpp, 141
- Square/SquareToStringConverter.h, 141
- SquareParser, 85
 - _instance, 88
 - ~SquareParser, 86
 - getInstance, 87
 - operator=, 87
 - parse, 87
 - SquareParser, 86
 - toString, 87
- SquareToStringConverter, 88
 - convert, 89
 - toString, 89
- third_edge
 - myTriangle::Triangle, 92
- top
 - myIsoscelesTrapezoid::IsoscelesTrapezoid, 43
- toString
 - CircleParser, 24
 - CircleToStringConverter, 25
 - ConverterFactory, 27
 - EllipseParser, 33
 - EllipseToStringConverter, 34
 - IsoscelesTrapezoidParser, 47
 - IsoscelesTrapezoidToStringConverter, 48
 - myCircle::Circle, 21
 - myEllipse::Ellipse, 29
 - myIsoscelesTrapezoid::IsoscelesTrapezoid, 43
 - myParallelogram::Parallelogram, 52
 - myRectangle::Rectangle, 61
 - myRhombus::Rhombus, 69
 - mySquare::Square, 84
 - myTriangle::Triangle, 92
 - Object, 49

- ParallelogramParser, [56](#)
- ParallelogramToStringConverter, [57](#)
- ParserFactory, [59](#)
- RectangleParser, [65](#)
- RectangleToStringConverter, [66](#)
- RhombusParser, [72](#)
- RhombusToStringConverter, [73](#)
- ShapesPrinter, [76](#)
- ShowDataCustom, [78](#)
- ShowDataDefault, [80](#)
- ShowTableCustom, [81](#)
- ShowTableDefault, [82](#)
- SquareParser, [87](#)
- SquareToStringConverter, [89](#)
- TriangleParser, [95](#)
- TriangleToStringConverter, [97](#)
- Triangle
 - myTriangle::Triangle, [90](#)
- Triangle/dllmain.cpp, [105](#)
- Triangle/framework.h, [109](#)
- Triangle/pch.cpp, [110](#)
- Triangle/pch.h, [114](#), [115](#)
- Triangle/Triangle.cpp, [141](#)
- Triangle/Triangle.h, [142](#)
- Triangle/TriangleParser.cpp, [142](#)
- Triangle/TriangleParser.h, [142](#), [143](#)
- Triangle/TriangleToStringConverter.cpp, [143](#)
- Triangle/TriangleToStringConverter.h, [143](#), [144](#)
- TriangleParser, [93](#)
 - _instance, [96](#)
 - ~TriangleParser, [94](#)
 - getInstance, [95](#)
 - operator=, [95](#)
 - parse, [95](#)
 - toString, [95](#)
 - TriangleParser, [94](#)
- TriangleToStringConverter, [96](#)
 - convert, [97](#)
 - toString, [97](#)
- utils.h
 - DOUBLE_PATTERN, [145](#)
 - PI, [145](#)
 - SHAPE_DATA, [145](#)
- utils/framework.h, [109](#)
- utils/pch.cpp, [111](#)
- utils/pch.h, [115](#)
- utils/utils.cpp, [144](#)
- utils/utils.h, [144](#), [146](#)
- width
 - myRectangle::Rectangle, [61](#)
- WIN32_LEAN_AND_MEAN
 - framework.h, [105–109](#)
- Đ ÁN CUI KÌ MÔN LP TRÌNH HNG ĐI TNG, [1](#)