Namespace Linear_Programming_Calculator_Desktop

Namespaces

Linear_Programming_Calculator_Desktop.Attributes

Linear_Programming_Calculator_Desktop.Converters

Linear_Programming_Calculator_Desktop.DTOs

Linear_Programming_Calculator_Desktop.Models

Linear_Programming_Calculator_Desktop.Services

Linear_Programming_Calculator_Desktop.Stores

Linear_Programming_Calculator_Desktop.ViewModels

Namespace Linear_Programming_Calculator_Desktop.Attributes

Classes

NumericOnlyAttribute

Validation attribute that ensures a value represents a valid numeric (double) input.

ValidVariableCountAttribute

Validation attribute to ensure an integer input meets a minimum value requirement.

Class NumericOnlyAttribute

Namespace: Linear_Programming_Calculator_Desktop.Attributes

Assembly: Linear Programming Calculator Desktop.dll

Validation attribute that ensures a value represents a valid numeric (double) input.

```
public class NumericOnlyAttribute : ValidationAttribute
```

Inheritance

object ← Attribute ← ValidationAttribute ← NumericOnlyAttribute

Inherited Members

ValidationAttribute.FormatErrorMessage(string) ,
ValidationAttribute.GetValidationResult(object, ValidationContext) ,
ValidationAttribute.IsValid(object, ValidationContext) ,
ValidationAttribute.Validate(object, ValidationContext) , ValidationAttribute.Validate(object, string) ,
ValidationAttribute.ErrorMessage , ValidationAttribute.ErrorMessageResourceName ,
ValidationAttribute.ErrorMessageResourceType , ValidationAttribute.ErrorMessageString ,
ValidationAttribute.RequiresValidationContext , Attribute.Equals(object) ,
Attribute.GetCustomAttribute(Assembly, Type) , Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) , Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) , Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) , Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) , Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) , Attribute.GetCustomAttributes(Module) ,
Attribute.GetCustomAttributes(Module, bool) , Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) , Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,
Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) ,

Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

NumericOnlyAttribute()

```
public NumericOnlyAttribute()
```

Methods

IsValid(object?)

Determines whether the specified value is a valid number.

```
public override bool IsValid(object? value)
```

Parameters

`value` object

The value to validate.

Returns

bool

True if `value` is non-null, not whitespace, and can be parsed as a double; otherwise, false.

Class ValidVariableCountAttribute

Namespace: Linear_Programming_Calculator_Desktop.Attributes

Assembly: Linear Programming Calculator Desktop.dll

Validation attribute to ensure an integer input meets a minimum value requirement.

```
public class ValidVariableCountAttribute : ValidationAttribute
```

Inheritance

object ← Attribute ← ValidationAttribute ← ValidVariableCountAttribute

Inherited Members

ValidationAttribute.FormatErrorMessage(string) ,
ValidationAttribute.GetValidationResult(object, ValidationContext) ,
ValidationAttribute.IsValid(object, ValidationContext) ,
ValidationAttribute.Validate(object, ValidationContext) , ValidationAttribute.Validate(object, string) ,
ValidationAttribute.ErrorMessage , ValidationAttribute.ErrorMessageResourceName ,
ValidationAttribute.ErrorMessageResourceType , ValidationAttribute.ErrorMessageString ,
ValidationAttribute.RequiresValidationContext , Attribute.Equals(object) ,
Attribute.GetCustomAttribute(Assembly, Type) , Attribute.GetCustomAttribute(Assembly, Type, bool) ,
Attribute.GetCustomAttribute(MemberInfo, Type) ,
Attribute.GetCustomAttribute(MemberInfo, Type, bool) ,
Attribute.GetCustomAttribute(Module, Type) , Attribute.GetCustomAttribute(Module, Type, bool) ,
Attribute.GetCustomAttribute(ParameterInfo, Type) ,
Attribute.GetCustomAttribute(ParameterInfo, Type, bool) , Attribute.GetCustomAttributes(Assembly) ,
Attribute.GetCustomAttributes(Assembly, bool) , Attribute.GetCustomAttributes(Assembly, Type) ,
Attribute.GetCustomAttributes(Assembly, Type, bool) , Attribute.GetCustomAttributes(MemberInfo) ,
Attribute.GetCustomAttributes(MemberInfo, bool) ,
Attribute.GetCustomAttributes(MemberInfo, Type) ,
Attribute.GetCustomAttributes(MemberInfo, Type, bool) , Attribute.GetCustomAttributes(Module) ,
Attribute.GetCustomAttributes(Module, bool) , Attribute.GetCustomAttributes(Module, Type) ,
Attribute.GetCustomAttributes(Module, Type, bool) , Attribute.GetCustomAttributes(ParameterInfo) ,
Attribute.GetCustomAttributes(ParameterInfo, bool) ,
Attribute.GetCustomAttributes(ParameterInfo, Type) ,
Attribute.GetCustomAttributes(ParameterInfo, Type, bool) , Attribute.GetHashCode() ,
Attribute.IsDefaultAttribute() , Attribute.IsDefined(Assembly, Type) ,
Attribute.IsDefined(Assembly, Type, bool) , Attribute.IsDefined(MemberInfo, Type) ,
Attribute.IsDefined(MemberInfo, Type, bool) , Attribute.IsDefined(Module, Type) ,

Attribute.IsDefined(Module, Type, bool) , Attribute.IsDefined(ParameterInfo, Type) , Attribute.IsDefined(ParameterInfo, Type, bool) , Attribute.Match(object) , Attribute.TypeId , object.Equals(object, object) , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ValidVariableCountAttribute(int)

Validation attribute to ensure an integer input meets a minimum value requirement.

```
public ValidVariableCountAttribute(int minValue)
```

Parameters

`minValue` int

The minimum acceptable integer value.

Properties

MinValue

Gets the minimum allowed value for the input.

```
public int MinValue { get; }
```

Property Value

int

Methods

IsValid(object?)

Validates whether the input value is a non-null, non-empty integer and meets the minimum value requirement.

```
public override bool IsValid(object? value)
```

Parameters

`value` object

The input value to validate.

Returns

bool

True if the input is valid; otherwise, false.

Namespace Linear_Programming_Calculator_Desktop.Converters

Classes

ConstraintTypeToSignConverter

Converts between ConstraintType enum values and their corresponding string symbols used in the UI.

IsMaximizationConverter

Converts the numeric selected value to a boolean.

IsNotLastItemConverter

Converts a pair of values to Visibility,

NullToVisibilityConverter

Converts a null or empty string value to Visibility.

Class ConstraintTypeToSignConverter

Namespace: Linear_Programming_Calculator_Desktop.Converters

Assembly: Linear Programming Calculator Desktop.dll

Converts between ConstraintType enum values and their corresponding string symbols used in the UI.

```
public class ConstraintTypeToSignConverter : IValueConverter
```

Inheritance

object ← ConstraintTypeToSignConverter

Implements

IValueConverter

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Convert(object, Type, object, CultureInfo)

Converts a ConstraintType value to its string representation.

```
public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
```

Parameters

`value` object

The ConstraintType value to convert.

`targetType` Type

The target type.

`parameter` object

Optional parameter.

`culture` CultureInfo

The culture information.

Returns

object

The corresponding symbol as string.

Exceptions

ArgumentException

Thrown if the input value is not of type ConstraintType.

ArgumentOutOfRangeException

Thrown if the ConstraintType is unknown.

ConvertBack(object, Type, object, CultureInfo)

Converts a string symbol back to its corresponding ConstraintType.

```
public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
```

Parameters

`value` object

The symbol to convert back.

`targetType` Type

The target type.

`parameter` object

Optional parameter.

`culture` CultureInfo

The culture information.

Returns

object

The corresponding ConstraintType.

Exceptions

ArgumentException

Thrown if the input is not a valid symbol string or symbol is unknown.

Class IsMaximizationConverter

Namespace: Linear_Programming_Calculator_Desktop.Converters

Assembly: Linear Programming Calculator Desktop.dll

Converts the numeric selected value to a boolean.

```
public class IsMaximizationConverter : IValueConverter
```

Inheritance

object ← IsMaximizationConverter

Implements

IValueConverter

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Convert(object, Type, object, CultureInfo)

Always returns `false`. This method does not perform any conversion.

```
public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
```

Parameters

`value` object

The boolean value to convert.

`targetType` Type

The target type.

`parameter` object

Optional parameter.

`culture` CultureInfo

Culture info.

Returns

object

Always `false`.

ConvertBack(object, Type, object, CultureInfo)

Converts back an integer (expected 0 or 1) to a boolean.

```
public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
```

Parameters

`value` object

The integer value to convert back.

`targetType` Type

The target type.

`parameter` object

Optional parameter.

`culture` CultureInfo

Culture info.

Returns

object

Boolean value: `true` if value is 0, otherwise `false`.

Class IsNotLastItemConverter

Namespace: Linear_Programming_Calculator_Desktop.Converters

Assembly: Linear Programming Calculator Desktop.dll

Converts a pair of values to Visibility,

```
public class IsNotLastItemConverter : IMultiValueConverter
```

Inheritance

object ← IsNotLastItemConverter

Implements

IMultiValueConverter

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Convert(object[], Type, object, CultureInfo)

Converts index and count to Visibility based on whether the index is less than count - 1.

```
public object Convert(object[] values, Type targetType, object parameter,
CultureInfo culture)
```

Parameters

`values` object[]

Array containing index and count.

`targetType` Type

Target type.

`parameter` object

Optional parameter.

`culture` CultureInfo

Culture info.

Returns

object

`Visibility.Collapsed` if null or empty; otherwise `Visibility.Visible`.

ConvertBack(object, Type[], object, CultureInfo)

Converts a binding target value to the source binding values.

```
public object[] ConvertBack(object value, Type[] targetTypes, object parameter,
CultureInfo culture)
```

Parameters

`value` object

The value that the binding target produces.

`targetTypes` Type[]

The array of types to convert to. The array length indicates the number and types of values that are suggested for the method to return.

`parameter` object

The converter parameter to use.

`culture` CultureInfo

The culture to use in the converter.

Returns

object[]

An array of values that have been converted from the target value back to the source values.

Class NullToVisibilityConverter

Namespace: Linear_Programming_Calculator_Desktop.Converters

Assembly: Linear Programming Calculator Desktop.dll

Converts a null or empty string value to Visibility.

```
public class NullToVisibilityConverter : IValueConverter
```

Inheritance

object ← NullToVisibilityConverter

Implements

IValueConverter

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

Convert(object, Type, object, CultureInfo)

Converts a string to Visibility.

```
public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
```

Parameters

`value` object

The value to check.

`targetType` Type

Target type.

`parameter` object

Optional parameter.

`culture` CultureInfo

Culture info.

Returns

object

`Visibility.Collapsed` if null or empty; otherwise `Visibility.Visible`.

ConvertBack(object, Type, object, CultureInfo)

Converts a value.

```
public object ConvertBack(object value, Type targetType, object parameter,
CultureInfo culture)
```

Parameters

`value` object

The value that is produced by the binding target.

`targetType` Type

The type to convert to.

`parameter` object

The converter parameter to use.

`culture` CultureInfo

The culture to use in the converter.

Returns

object

A converted value. If the method returns null, the valid null value is used.

Namespace Linear_Programming_Calculator_Desktop.DTOs

Classes

LinearProgramResultDto

Data transfer object for passing data between views.

Class LinearProgramResultDto

Namespace: Linear_Programming_Calculator_Desktop.DTOs

Assembly: Linear Programming Calculator Desktop.dll

Data transfer object for passing data between views.

```
public record LinearProgramResultDto : IEquatable<LinearProgramResultDto>
```

Inheritance

object ← LinearProgramResultDto

Implements

IEquatable<LinearProgramResultDto>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ErrorMessage

Error message text, if any.

```
public string? ErrorMessage { get; init; }
```

Property Value

string

GHistory

List of Gomory history steps.

```
public List<GomoryHistory>? GHistory { get; init; }
```

Property Value

List <GomoryHistory>

IsIntegerProblem

Indicates whether the problem is an integer programming problem.

```
public required bool IsIntegerProblem { get; init; }
```

Property Value

bool

SHistory

Instance of the Simplex history steps.

```
public required SimplexHistory SHistory { get; init; }
```

Property Value

SimplexHistory

Namespace Linear_Programming_Calculator_Desktop.Models

Classes

FormattedLinearProblem

Represents a formatted text for display in the Result view.

LinearProgramInput

Represents user input for a LPP collected from the view.

SimplexCell

Represents a cell in the Simplex table.

Class FormattedLinearProblem

Namespace: Linear_Programming_Calculator_Desktop.Models

Assembly: Linear Programming Calculator Desktop.dll

Represents a formatted text for display in the Result view.

```
public record FormattedLinearProblem : IEquatable<FormattedLinearProblem>
```

Inheritance

object ← FormattedLinearProblem

Implements

IEquatable<FormattedLinearProblem>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

DomainText

Formatted text of the domain constraint, such as x1, x2, ..., xn ≥ 0.

```
public required string DomainText { get; init; }
```

Property Value

string

FormattedConstraints

Formatted text of the constraints.

```
public required List<string> FormattedConstraints { get; init; }
```

Property Value

List<string>

FormattedObjectiveFunction

Formatted text of the objective function.

```
public required string FormattedObjectiveFunction { get; init; }
```

Property Value

string

IntegerNote

Formatted text of the integer constraint, such as x1, x2, ..., xn are integers.

```
public string? IntegerNote { get; init; }
```

Property Value

string

Class LinearProgramInput

Namespace: Linear_Programming_Calculator_Desktop.Models

Assembly: Linear Programming Calculator Desktop.dll

Represents user input for a LPP collected from the view.

```
public class LinearProgramInput
```

Inheritance

object ← LinearProgramInput

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

ConstraintValues

The values of the constraints entered by the user.

```
public List<ConstraintViewModel> ConstraintValues { get; set; }
```

Property Value

List<ConstraintViewModel>

IntegerCheck

Indicates whether the problem includes integer constraint.

```
public bool IntegerCheck { get; set; }
```

Property Value

bool

IsMaximization

Indicates whether the problem is a maximization problem.

```
public bool IsMaximization { get; set; }
```

Property Value

bool

ObjectiveFunctionValues

The values of the objective function coefficients entered by the user.

```
public List<FieldViewModel> ObjectiveFunctionValues { get; set; }
```

Property Value

List <FieldViewModel>

Class SimplexCell

Namespace: Linear_Programming_Calculator_Desktop.Models

Assembly: Linear Programming Calculator Desktop.dll

Represents a cell in the Simplex table.

```
public class SimplexCell
```

Inheritance

object ← SimplexCell

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Background

The background brush used to highlight the pivot row or column.

```
public Brush Background { get; set; }
```

Property Value

Brush

Text

The text displayed in the cell.

```
public string Text { get; set; }
```

Property Value

string

Namespace Linear_Programming_Calculator_Desktop.Services

Classes

GomoryCutFormatterService

Service for formatting Gomory cuts into readable string.

NavigationService<TViewModel>

Represents a navigation service for navigating between ViewModels without parameters.

NavigationService<TViewModel, TParameter>

Represents a navigation service for navigating between ViewModels with parameters.

OptimalResultSummaryService

Service for formatting the optimal result summary.

ProblemFormatterService

Service that provides methods to format different parts of a LPP for display.

Interfaces

IGomoryCutFormatterService

Service interface for formatting Gomory cuts into readable string representations.

INavigator<TViewModel>

Interface for Navigator without parameters.

INavigator<TViewModel, TParameter>

Interface for Navigator with parameters.

IOptimalResultSummaryService

Service interface for formatting the optimal result summary.

IProblemFormatterService

Service interface that provides methods to format different parts of a LPP for display.

Class GomoryCutFormatterService

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Service for formatting Gomory cuts into readable string.

```
public class GomoryCutFormatterService : IGomoryCutFormatterService
```

Inheritance

object ← GomoryCutFormatterService

Implements

IGomoryCutFormatterService

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

BuildEqualitySection(BranchCut)

Builds the section of the Gomory cut corresponding to equality constraints.

```
public string BuildEqualitySection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

string

A formatted string representing the equality section.

BuildFractionPartsSection(BranchCut)

Builds the section of the Gomory cut showing fractional parts of the variables.

```
public string BuildFractionPartsSection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

string

A formatted string representing the fractional parts section.

BuildGomoryCutLines(BranchCut)

Builds the formatted lines representing the given BranchCut instance.

```
public List<string> BuildGomoryCutLines(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

List <string>

A list of strings, each representing a line of the Gomory cut.

BuildInequalitySection(BranchCut)

Builds the section of the Gomory cut corresponding to inequality constraints.

```
public string BuildInequalitySection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

Returns

string

A formatted string representing the inequality section.

BuildRightHandSideSection(BranchCut)

Builds the section showing the right-hand side (RHS) values of the Gomory cut.

```
public string BuildRightHandSideSection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

string

A formatted string representing the RHS section.

Interface IGomoryCutFormatterService

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Service interface for formatting Gomory cuts into readable string representations.

```
public interface IGomoryCutFormatterService
```

Methods

BuildEqualitySection(BranchCut)

Builds the section of the Gomory cut corresponding to equality constraints.

```
string BuildEqualitySection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

string

A formatted string representing the equality section.

BuildFractionPartsSection(BranchCut)

Builds the section of the Gomory cut showing fractional parts of the variables.

```
string BuildFractionPartsSection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

string

A formatted string representing the fractional parts section.

BuildGomoryCutLines(BranchCut)

Builds the formatted lines representing the given BranchCut instance.

```
List<string> BuildGomoryCutLines(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

List<string>

A list of strings, each representing a line of the Gomory cut.

BuildInequalitySection(BranchCut)

Builds the section of the Gomory cut corresponding to inequality constraints.

```
string BuildInequalitySection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

Returns

string

A formatted string representing the inequality section.

BuildRightHandSideSection(BranchCut)

Builds the section showing the right-hand side (RHS) values of the Gomory cut.

```
string BuildRightHandSideSection(BranchCut branchCut)
```

Parameters

`branchCut` BranchCut

The branch cut.

Returns

string

A formatted string representing the RHS section.

Interface INavigator<TViewModel>

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Interface for Navigator without parameters.

```
public interface INavigator<TViewModel>
```

Type Parameters

`TViewModel`

The type of ViewModel to navigate to.

Methods

Navigate()

Used to navigate to target View Model.

```
void Navigate()
```

Interface INavigator<TViewModel, TParameter>

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Interface for Navigator with parameters.

```
public interface INavigator<TViewModel, TParameter>
```

Type Parameters

`TViewModel`

The type of ViewModel to navigate to.

`TParameter`

Parameters to be passed.

Methods

Navigate(TParameter)

Used to navigate to target View Model with parameters.

```
void Navigate(TParameter parameter)
```

Parameters

`parameter` TParameter

Interface IOptimalResultSummaryService

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Service interface for formatting the optimal result summary.

```
public interface IOptimalResultSummaryService
```

Methods

FormatObjectiveFunctionValue(SimplexTable)

Formats the output of the objective function's optimal value.

```
string FormatObjectiveFunctionValue(SimplexTable table)
```

Parameters

`table` SimplexTable

The table containing the results.

Returns

string

The formatted optimal result of the objective function.

FormatVariableAssignment(SimplexTable, int)

Formats the output of a variable assignment.

```
(Fraction, string) FormatVariableAssignment(SimplexTable table, int currentIndex)
```

Parameters

`table` SimplexTable

The table containing the results.

`currentIndex` int

The current index of the variable in the objective function.

Returns

(Fraction, string)

A tuple containing the variable value and its formatted output.

Interface IProblemFormatterService

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Service interface that provides methods to format different parts of a LPP for display.

```
public interface IProblemFormatterService
```

Methods

BuildConstraints(LinearProgrammingProblem, bool)

Builds a list of formatted strings representing the problem constraints.

```
List<string> BuildConstraints(LinearProgrammingProblem problem, bool isEqual)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

`isEqual` bool

Indicates whether constraints are equalities or inequalities.

Returns

List<string>

A list of formatted constraint strings.

BuildDomain(LinearProgrammingProblem)

Builds a formatted string representing the domain constraint, such as x1, x2, ..., xn ≥ 0..

```
string BuildDomain(LinearProgrammingProblem problem)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

Returns

string

A formatted string of the domain constraints.

BuildIntegerNote(LinearProgrammingProblem, bool)

Builds a formatted note about integer constraint if the problem is integer-constrained.

```
string BuildIntegerNote(LinearProgrammingProblem problem, bool isIntegerProblem)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

`isIntegerProblem` bool

Indicates whether the problem has integer constraint.

Returns

string

A formatted string note about integer constraint, or an empty string if not applicable.

BuildObjectiveFunction(LinearProgrammingProblem)

Builds a formatted string representation of the objective function.

```
string BuildObjectiveFunction(LinearProgrammingProblem problem)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

Returns

string

A formatted string of the objective function.

Class NavigationService<TViewModel>

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Represents a navigation service for navigating between ViewModels without parameters.

```
public class NavigationService<TViewModel> : INavigator<TViewModel> where TViewModel
: ObservableObject
```

Type Parameters

`TViewModel`

The type of the ViewModel to navigate to. Must inherit from ObservableObject.

Inheritance

object ← NavigationService<TViewModel>

Implements

INavigator<TViewModel>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

NavigationService(NavigationStore, Func<TViewModel>)

Represents a navigation service for navigating between ViewModels without parameters.

```
public NavigationService(NavigationStore navigationStore, Func<TViewModel> createViewModel)
```

Parameters

`navigationStore` NavigationStore

The store that contains the current ViewModel instance.

`createViewModel` Func<TViewModel>

Factory method to create a new instance of the target ViewModel.

Methods

Navigate()

Navigates to the target ViewModel.

```
public void Navigate()
```

Class NavigationService<TViewModel, TParameter>

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Represents a navigation service for navigating between ViewModels with parameters.

```
public class NavigationService<TViewModel, TParameter> : INavigator<TViewModel, TParameter>
where TViewModel : ObservableObject
```

Type Parameters

`TViewModel`

The type of the ViewModel to navigate to. Must inherit from ObservableObject.

`TParameter`

The type of parameter to be passed during navigation.

Inheritance

object ← NavigationService<TViewModel, TParameter>

Implements

INavigator<TViewModel, TParameter>

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

NavigationService(NavigationStore, Func<TParameter, TViewModel>)

Represents a navigation service for navigating between ViewModels with parameters.

```
public NavigationService(NavigationStore navigationStore, Func<TParameter,
TViewModel> createViewModelWithParam)
```

Parameters

`navigationStore` NavigationStore

The store that contains the current ViewModel instance.

`createViewModelWithParam` Func<TParameter, TViewModel>

Factory method to create a new instance of the target ViewModel with parameters.

Methods

Navigate(TParameter)

Navigates to the target ViewModel with parameters.

```
public void Navigate(TParameter parameter)
```

Parameters

`parameter` TParameter

The parameter to pass to the target ViewModel.

Class OptimalResultSummaryService

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Service for formatting the optimal result summary.

```
public class OptimalResultSummaryService : IOptimalResultSummaryService
```

Inheritance

object ← OptimalResultSummaryService

Implements

IOptimalResultSummaryService

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

FormatObjectiveFunctionValue(SimplexTable)

Formats the output of the objective function's optimal value.

```
public string FormatObjectiveFunctionValue(SimplexTable table)
```

Parameters

`table` SimplexTable

The table containing the results.

Returns

string

The formatted optimal result of the objective function.

Remarks

Retrieves the first value from the delta row.

FormatVariableAssignment(SimplexTable, int)

Formats the output of a variable assignment.

```
public (Fraction, string) FormatVariableAssignment(SimplexTable table, int currentIndex)
```

Parameters

`table` SimplexTable

The table containing the results.

`currentIndex` int

The current index of the variable in the objective function.

Returns

(Fraction, string)

A tuple containing the variable value and its formatted output.

Remarks

Searches the row variables for the name and index of the variable that corresponds to the objective function's variable index, then retrieves the value from the table if available, and returns the result.

Class ProblemFormatterService

Namespace: Linear_Programming_Calculator_Desktop.Services

Assembly: Linear Programming Calculator Desktop.dll

Service that provides methods to format different parts of a LPP for display.

```
public class ProblemFormatterService : IProblemFormatterService
```

Inheritance

object ← ProblemFormatterService

Implements

IProblemFormatterService

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Methods

BuildConstraints(LinearProgrammingProblem, bool)

Builds a list of formatted strings representing the problem constraints.

```
public List<string> BuildConstraints(LinearProgrammingProblem problem, bool isEqual)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

`isEqual` bool

Indicates whether constraints are equalities or inequalities.

Returns

List<string>

A list of formatted constraint strings.

Remarks

Builds a list of formatted constraint strings by processing each constraint's coefficients, formatting negative coefficients with parentheses, assigning appropriate inequality or equality signs, and combining these into readable constraint expressions.

BuildDomain(LinearProgrammingProblem)

Builds a formatted string representing the domain constraint, such as x1, x2, ..., xn ≥ 0..

```
public string BuildDomain(LinearProgrammingProblem problem)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

Returns

string

A formatted string of the domain constraints.

BuildIntegerNote(LinearProgrammingProblem, bool)

Builds a formatted note about integer constraint if the problem is integer-constrained.

```
public string BuildIntegerNote(LinearProgrammingProblem problem, bool isIntegerProblem)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

`isIntegerProblem` bool

Indicates whether the problem has integer constraint.

Returns

string

A formatted string note about integer constraint, or an empty string if not applicable.

BuildObjectiveFunction(LinearProgrammingProblem)

Builds a formatted string representation of the objective function.

```
public string BuildObjectiveFunction(LinearProgrammingProblem problem)
```

Parameters

`problem` LinearProgrammingProblem

The linear programming problem.

Returns

string

A formatted string of the objective function.

Remarks

Builds a formatted string representation of the objective function by concatenating all coefficients (objective function, slack, and artificial variables) with variable names, properly formatting negative coefficients, and appending the optimization direction (max or min).

Namespace Linear_Programming_Calculator_Desktop.Stores

Classes

LinearProgramInputStore

Stores the current input data for a LPP and notifies subscribers when the input changes.

NavigationStore

Stores the current active ViewModel in the application and notifies subscribers when the current ViewModel changes.

Class LinearProgramInputStore

Namespace: Linear_Programming_Calculator_Desktop.Stores

Assembly: Linear Programming Calculator Desktop.dll

Stores the current input data for a LPP and notifies subscribers when the input changes.

```
public class LinearProgramInputStore
```

Inheritance

object ← LinearProgramInputStore

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

CurrentLinearProgramInput

Current linear program input.

```
public LinearProgramInput CurrentLinearProgramInput { get; set; }
```

Property Value

LinearProgramInput

Remarks

Invokes the CurrentLinearProgramInputChanged event when set.

Events

CurrentLinearProgramInputChanged

Event that is raised when the current linear program input changes.

```
public event Action CurrentLinearProgramInputChanged
```

Event Type

Action

Class NavigationStore

Namespace: Linear_Programming_Calculator_Desktop.Stores

Assembly: Linear Programming Calculator Desktop.dll

Stores the current active ViewModel in the application and notifies subscribers when the current ViewModel changes.

```
public class NavigationStore
```

Inheritance

object ← NavigationStore

Inherited Members

object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

CurrentViewModel

Current ViewModel.

```
public ObservableObject CurrentViewModel { get; set; }
```

Property Value

ObservableObject

Remarks

Raises the CurrentViewModelChanged event.

Events

CurrentViewModelChanged

Event that is raised when the current ViewModel changes.

```
public event Action CurrentViewModelChanged
```

Event Type

Action

Namespace Linear_Programming_Calculator_Desktop.ViewModels

Classes

ConstraintViewModel

Represents a ViewModel that is bound to a single constraint in the View.

EquationInputViewModel

Represent a ViewModel that provides blocks for problem input, shown in the View.

FieldViewModel

Represents a ViewModel that contains a single field for user input.

GomoryViewModel

Represents a ViewModel with the single step of the Gomory cutting-plane method.

MainViewModel

Represents a base ViewModel for storing and handling navigation between ViewModels.

ResultsViewModel

Represents a ViewModel that displays the results of solving a LPP.

SimplexViewModel

Represents a ViewModel holding the table content and related information displayed in the View.

StartViewModel

Represents a ViewModel for entering the number of variables and constraints.

Class ConstraintViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represents a ViewModel that is bound to a single constraint in the View.

```
public class ConstraintViewModel : ObservableValidator, INotifyPropertyChanged,
INotifyPropertyChanging, INotifyDataErrorInfo
```

Inheritance

object ← ObservableObject ← ObservableValidator ← ConstraintViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging, INotifyDataErrorInfo

Inherited Members

ObservableValidator.SetProperty<T>(ref T, T, bool, string) ,
ObservableValidator.SetProperty<T>(ref T, T, IEqualityComparer<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, Action<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, IEqualityComparer<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.ClearErrors(string) , ObservableValidator.GetErrors(string) ,
ObservableValidator.ValidateAllProperties() , ObservableValidator.ValidateProperty(object, string) ,
ObservableValidator.HasErrors , ObservableValidator.ErrorsChanged ,
ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,

ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) , ObservableObject.SetProperty<T>(ref T, T, string) , ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) , ObservableObject.SetProperty<T>(T, T, Action<T>, string) , ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, Action<Task>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, Action<Task<T>>, string) , ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ConstraintViewModel(int)

Constructor that initializes the constraint values.

```
public ConstraintViewModel(int variables)
```

Parameters

`variables` int

Quantity of constraints variables.

Properties

ConstraintType

The type of the constraint.

```
public ConstraintType ConstraintType { get; set; }
```

Property Value

ConstraintType

ConstraintValues

List of coefficients for the variables.

```
public ObservableCollection<FieldViewModel> ConstraintValues { get; set; }
```

Property Value

ObservableCollection <FieldViewModel>

RightSideValue

The right-hand side (RHS) value.

```
[NumericOnly]
public string RightSideValue { get; set; }
```

Property Value

string

Class EquationInputViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represent a ViewModel that provides blocks for problem input, shown in the View.

```
public class EquationInputViewModel : ObservableValidator, INotifyPropertyChanged,
INotifyPropertyChanging, INotifyDataErrorInfo
```

Inheritance

object ← ObservableObject ← ObservableValidator ← EquationInputViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging, INotifyDataErrorInfo

Inherited Members

ObservableValidator.SetProperty<T>(ref T, T, bool, string) ,
ObservableValidator.SetProperty<T>(ref T, T, IEqualityComparer<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, Action<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, IEqualityComparer<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.ClearErrors(string) , ObservableValidator.GetErrors(string) ,
ObservableValidator.ValidateAllProperties() , ObservableValidator.ValidateProperty(object, string) ,
ObservableValidator.HasErrors , ObservableValidator.ErrorsChanged ,
ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,

ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) , ObservableObject.SetProperty<T>(ref T, T, string) , ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) , ObservableObject.SetProperty<T>(T, T, Action<T>, string) , ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, Action<Task>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, Action<Task<T>>, string) , ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

EquationInputViewModel(LinearProgramInputStore, INavigator<ResultsViewModel, LinearProgramResultDto>, INavigator<StartViewModel>)

Initializes a new instance of EquationInputViewModel using data from an existing store. Configures navigation services and restores previous data from storage.

```
public EquationInputViewModel(LinearProgramInputStore lpStore, INavigator<ResultsViewModel,
LinearProgramResultDto> navigationService, INavigator<StartViewModel> backNavigationService)
```

Parameters

`lpStore` LinearProgramInputStore

The store containing the current LinearProgramInput data.

`navigationService` INavigator<ResultsViewModel, LinearProgramResultDto>

Service for navigating to the ResultsViewModel.

`backNavigationService` INavigator<StartViewModel>

Service for navigating back to the StartViewModel.

EquationInputViewModel((int variables, int constraints), LinearProgramInputStore, INavigator<ResultsViewModel, LinearProgramResultDto>, INavigator<StartViewModel>)

Initializes a new instance of EquationInputViewModel with a specified number of variables and constraints.

```
public EquationInputViewModel((int variables, int constraints) parameters,
LinearProgramInputStore lpStore, INavigator<ResultsViewModel, LinearProgramResultDto>
navigationService, INavigator<StartViewModel> backNavigationService)
```

Parameters

`parameters` (int variables, int constraints)

Tuple containing the number of variables and constraints.

`lpStore` LinearProgramInputStore

The store containing the current LinearProgramInput data.

`navigationService` INavigator<ResultsViewModel, LinearProgramResultDto>

Service for navigating to the ResultsViewModel.

`backNavigationService` INavigator<StartViewModel>

Service for navigating back to the StartViewModel.

Remarks

Configures navigation services and initializes ObjectiveFunctionValues and ConstraintValues based on the provided values.

Properties

ConstraintValues

A collection of constraints.

```
public ObservableCollection<ConstraintViewModel>? ConstraintValues { get; }
```

Property Value

ObservableCollection <ConstraintViewModel>

IntegerCheck

Indicates whether the problem has integer constraint.

```
public bool IntegerCheck { get; set; }
```

Property Value

bool

IntegerVariablesText

Formatted text for displaying an integer constraint.

```
public string IntegerVariablesText { get; }
```

Property Value

string

IsMaximization

Indicates whether the problem is a maximization.

```
public bool IsMaximization { get; set; }
```

Property Value

bool

NewProblemCommand

Gets an IRelayCommand instance wrapping NewProblem().

```
public IRelayCommand NewProblemCommand { get; }
```

Property Value

IRelayCommand

ObjectiveFunctionValues

List of coefficients for the variables in the objective function.

```
public ObservableCollection<FieldViewModel>? ObjectiveFunctionValues { get; }
```

Property Value

ObservableCollection<FieldViewModel>

SolveCommand

Gets an IRelayCommand instance wrapping Solve().

```
public IRelayCommand SolveCommand { get; }
```

Property Value

IRelayCommand

Methods

NewProblem()

Command for navigating to the StartViewModel for entering the number of variables and constraints.

```
[RelayCommand]
public void NewProblem()
```

Solve()

Command that solves the problem using the entered data and navigates to the ResultsViewModel.

```
[RelayCommand]
public void Solve()
```

Class FieldViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represents a ViewModel that contains a single field for user input.

```
public class FieldViewModel : ObservableValidator, INotifyPropertyChanged,
INotifyPropertyChanging, INotifyDataErrorInfo
```

Inheritance

object ← ObservableObject ← ObservableValidator ← FieldViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging, INotifyDataErrorInfo

Inherited Members

ObservableValidator.SetProperty<T>(ref T, T, bool, string) ,
ObservableValidator.SetProperty<T>(ref T, T, IEqualityComparer<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, Action<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, IEqualityComparer<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.ClearErrors(string) , ObservableValidator.GetErrors(string) ,
ObservableValidator.ValidateAllProperties() , ObservableValidator.ValidateProperty(object, string) ,
ObservableValidator.HasErrors , ObservableValidator.ErrorsChanged ,
ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,

ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) , ObservableObject.SetProperty<T>(ref T, T, string) , ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) , ObservableObject.SetProperty<T>(T, T, Action<T>, string) , ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, Action<Task>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, Action<Task<T>>, string) , ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Properties

Label

The label displayed for the field.

```
public required string Label { get; set; }
```

Property Value

string

Value

The numeric value of the field as a string.

```
[NumericOnly]
public string Value { get; set; }
```

Property Value

string

Class GomoryViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represents a ViewModel with the single step of the Gomory cutting-plane method.

```
public class GomoryViewModel : ObservableObject, INotifyPropertyChanged,
INotifyPropertyChanging
```

Inheritance

object ← ObservableObject ← GomoryViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging

Inherited Members

ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,
ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) ,
ObservableObject.SetProperty<T>(ref T, T, string) ,
ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) ,
ObservableObject.SetProperty<T>(T, T, Action<T>, string) ,
ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>,
string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string)
,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task,
Action<Task>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, Action<Task<T>>, string) ,
ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) ,
object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

GomoryViewModel(GomoryHistory, List<string>, IOptimalResultSummaryService, IGomoryCutFormatterService)

Represents a ViewModel with the single step of the Gomory cutting-plane method.

```
public GomoryViewModel(GomoryHistory gomoryStep, List<string> objFuncCoeff,
IOptimalResultSummaryService summaryService, IGomoryCutFormatterService cutFormatterService)
```

Parameters

`gomoryStep` GomoryHistory

The current step of Gomory history containing the cut and fractional values.

`objFuncCoeff` List<string>

List of objective function coefficients for reference in formatting.

`summaryService` IOptimalResultSummaryService

Service for generating summaries of optimal results.

`cutFormatterService` IGomoryCutFormatterService

Service for formatting Gomory cuts into readable strings.

Properties

GomoryCutLines

List of formatted strings representing the Gomory cut for this step.

```
public List<string> GomoryCutLines { get; }
```

Property Value

List<string>

GomoryStepSummary

A summary string describing this Gomory step.

```
public string GomoryStepSummary { get; }
```

Property Value

string

GomoryTables

List of SimplexViewModel representing simplex tables associated with this Gomory step.

```
public List<SimplexViewModel>? GomoryTables { get; }
```

Property Value

List<SimplexViewModel>

MaxFractionDisplayText

Text displaying the variable with the maximum fractional part in this Gomory step.

```
public string MaxFractionDisplayText { get; }
```

Property Value

string

Class MainViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represents a base ViewModel for storing and handling navigation between ViewModels.

```
public class MainViewModel : ObservableObject, INotifyPropertyChanged,
INotifyPropertyChanging
```

Inheritance

object ← ObservableObject ← MainViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging

Inherited Members

ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,
ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) ,
ObservableObject.SetProperty<T>(ref T, T, string) ,
ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) ,
ObservableObject.SetProperty<T>(T, T, Action<T>, string) ,
ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>,
string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string)
,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task,
Action<Task>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, Action<Task<T>>, string) ,
ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) ,
object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

MainViewModel(NavigationStore)

Subscribes to event to update the current view model.

```
public MainViewModel(NavigationStore navigationStore)
```

Parameters

`navigationStore` NavigationStore

Navigation store containing the current view model.

Properties

CurrentViewModel

Current ViewModel.

```
public ObservableObject CurrentViewModel { get; }
```

Property Value

ObservableObject

Class ResultsViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represents a ViewModel that displays the results of solving a LPP.

```
public class ResultsViewModel : ObservableObject, INotifyPropertyChanged,
INotifyPropertyChanging
```

Inheritance

object ← ObservableObject ← ResultsViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging

Inherited Members

ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,
ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) ,
ObservableObject.SetProperty<T>(ref T, T, string) ,
ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) ,
ObservableObject.SetProperty<T>(T, T, Action<T>, string) ,
ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>,
string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string)
,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task,
Action<Task>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, Action<Task<T>>, string) ,
ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) ,
object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

ResultsViewModel(LinearProgramResultDto, INavigator<StartViewModel>, INavigator<EquationInputViewModel>, IProblemFormatterService, IOptimalResultSummaryService, IGomoryCutFormatterService)

Initializes all services and formats all mathematical block steps.

```
public ResultsViewModel(LinearProgramResultDto resultDto, INavigator<StartViewModel>
backNavigator, INavigator<EquationInputViewModel> editNavigator, IProblemFormatterService
problemFormatter, IOptimalResultSummaryService resultSummaryService,
IGomoryCutFormatterService gomoryCutFormatterService)
```

Parameters

`resultDto` LinearProgramResultDto

Results of the solved LPP.

`backNavigator` INavigator<StartViewModel>

Navigator for returning to the StartViewModel.

`editNavigator` INavigator<EquationInputViewModel>

Navigator for returning to the equation EquationInputViewModel.

`problemFormatter` IProblemFormatterService

Service for formatting the problem for display.

`resultSummaryService` IOptimalResultSummaryService

Service for generating the optimal result summary.

`gomoryCutFormatterService` IGomoryCutFormatterService

Service for formatting Gomory cut steps for display.

Properties

ArtificialVariableBlock

Formatted representation of the problem after introducing artificial variables, if applicable.

```
public FormattedLinearProblem? ArtificialVariableBlock { get; }
```

Property Value

FormattedLinearProblem

EditProblemCommand

Gets an IRelayCommand instance wrapping EditProblem().

```
public IRelayCommand EditProblemCommand { get; }
```

Property Value

IRelayCommand

ErrorMessage

Error message describing any issues encountered during solving.

```
public string? ErrorMessage { get; }
```

Property Value

string

GomoryHistory

History of Gomory cut method steps, if needed.

```
public List<GomoryViewModel>? GomoryHistory { get; }
```

Property Value

List <GomoryViewModel>

InitialBasisText

Text description of the initial basis for the problem.

```
public string InitialBasisText { get; }
```

Property Value

string

MathModelBlock

Formatted representation of the original mathematical model of the problem.

```
public FormattedLinearProblem MathModelBlock { get; }
```

Property Value

FormattedLinearProblem

NewProblemCommand

Gets an IRelayCommand instance wrapping NewProblem().

```
public IRelayCommand NewProblemCommand { get; }
```

Property Value

IRelayCommand

OptimalResult

Final optimal simplex table after solving.

```
public SimplexViewModel OptimalResult { get; }
```

Property Value

SimplexViewModel

SimplexSolutionSummary

Summary of the solution obtained by the simplex method.

```
public string SimplexSolutionSummary { get; }
```

Property Value

string

SimplexTables

List of formatted simplex tables for each iteration step.

```
public List<SimplexViewModel> SimplexTables { get; }
```

Property Value

List<SimplexViewModel>

SlackVariableBlock

Formatted representation of the problem after introducing slack variables.

```
public FormattedLinearProblem SlackVariableBlock { get; }
```

Property Value

FormattedLinearProblem

Methods

EditProblem()

Navigates to the view for editing the current LPP.

```
[RelayCommand]
public void EditProblem()
```

NewProblem()

Navigates to the view for creating a new LPP.

```
[RelayCommand]
public void NewProblem()
```

Class SimplexViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represents a ViewModel holding the table content and related information displayed in the View.

```
public class SimplexViewModel : ObservableObject, INotifyPropertyChanged,
INotifyPropertyChanging
```

Inheritance

object ← ObservableObject ← SimplexViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging

Inherited Members

ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,
ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) ,
ObservableObject.SetProperty<T>(ref T, T, string) ,
ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) ,
ObservableObject.SetProperty<T>(T, T, Action<T>, string) ,
ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) ,
ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>,
string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string)
,
ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task,
Action<Task>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, string) ,
ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>,
Task<T>, Action<Task<T>>, string) ,
ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) ,
object.Equals(object, object) , object.GetHashCode() , object.GetType() ,
object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

SimplexViewModel(SimplexStep)

Represents a ViewModel holding the table content and related information displayed in the View.

```
public SimplexViewModel(SimplexStep simplexStep)
```

Parameters

`simplexStep` SimplexStep

Current step of the solved problem.

Properties

Cells

Collection of cells representing the current table.

```
public ObservableCollection<SimplexCell> Cells { get; }
```

Property Value

ObservableCollection<SimplexCell>

PivotRowKey

Gets the key of the pivot row in the current table.

```
public string PivotRowKey { get; }
```

Property Value

string

Step

Current step of the solved problem.

```
public SimplexStep Step { get; }
```

Property Value

SimplexStep

TotalCols

Total number of columns in the table.

```
public int TotalCols { get; set; }
```

Property Value

int

TotalRows

Total number of rows in the table.

```
public int TotalRows { get; set; }
```

Property Value

int

Methods

LoadFromTable()

Loads the table data into a new SimplexViewModel instance.

```
public SimplexViewModel LoadFromTable()
```

Returns

SimplexViewModel

A new SimplexViewModel populated with the current table data.

Class StartViewModel

Namespace: Linear_Programming_Calculator_Desktop.ViewModels

Assembly: Linear Programming Calculator Desktop.dll

Represents a ViewModel for entering the number of variables and constraints.

```
public class StartViewModel : ObservableValidator, INotifyPropertyChanged,
INotifyPropertyChanging, INotifyDataErrorInfo
```

Inheritance

object ← ObservableObject ← ObservableValidator ← StartViewModel

Implements

INotifyPropertyChanged, INotifyPropertyChanging, INotifyDataErrorInfo

Inherited Members

ObservableValidator.SetProperty<T>(ref T, T, bool, string) ,
ObservableValidator.SetProperty<T>(ref T, T, IEqualityComparer<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, Action<T>, bool, string) ,
ObservableValidator.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, bool, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(ref T, T, IEqualityComparer<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.TrySetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, out IReadOnlyCollection<ValidationResult>, string) ,
ObservableValidator.ClearErrors(string) , ObservableValidator.GetErrors(string) ,
ObservableValidator.ValidateAllProperties() , ObservableValidator.ValidateProperty(object, string) ,
ObservableValidator.HasErrors , ObservableValidator.ErrorsChanged ,
ObservableObject.OnPropertyChanged(PropertyChangedEventArgs) ,
ObservableObject.OnPropertyChanging(PropertyChangingEventArgs) ,

ObservableObject.OnPropertyChanged(string) , ObservableObject.OnPropertyChanging(string) , ObservableObject.SetProperty<T>(ref T, T, string) , ObservableObject.SetProperty<T>(ref T, T, IEqualityComparer<T>, string) , ObservableObject.SetProperty<T>(T, T, Action<T>, string) , ObservableObject.SetProperty<T>(T, T, IEqualityComparer<T>, Action<T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, TModel, Action<TModel, T>, string) , ObservableObject.SetProperty<TModel, T>(T, T, IEqualityComparer<T>, TModel, Action<TModel, T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, string) , ObservableObject.SetPropertyAndNotifyOnCompletion(ref ObservableObject.TaskNotifier, Task, Action<Task>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, string) , ObservableObject.SetPropertyAndNotifyOnCompletion<T>(ref ObservableObject.TaskNotifier<T>, Task<T>, Action<Task<T>>, string) , ObservableObject.PropertyChanged , ObservableObject.PropertyChanging , object.Equals(object) , object.Equals(object, object) , object.GetHashCode() , object.GetType() , object.MemberwiseClone() , object.ReferenceEquals(object, object) , object.ToString()

Constructors

StartViewModel(INavigator<EquationInputViewModel, (int variables, int constraints)>, LinearProgramInputStore)

Represents a ViewModel for entering the number of variables and constraints.

```
public StartViewModel(INavigator<EquationInputViewModel, (int variables, int constraints)>
navigationService, LinearProgramInputStore linearProgramInputStore)
```

Parameters

`navigationService` INavigator<EquationInputViewModel, (int variables, int constraints)>

Navigation service used to navigate to the EquationInputViewModel.

`linearProgramInputStore` LinearProgramInputStore

Store containing the current LinearProgramInput.

Properties

Constraints

Number of constraint.

```
[ValidVariableCount(2)]
public string Constraints { get; set; }
```

Property Value

string

GenerateProblemCommand

Gets an IRelayCommand instance wrapping GenerateProblem().

```
public IRelayCommand GenerateProblemCommand { get; }
```

Property Value

IRelayCommand

Variables

Number of variables.

```
[ValidVariableCount(2)]
public string Variables { get; set; }
```

Property Value

string

Methods

GenerateProblem()

Clears the previous data in LinearProgramInputStore and navigates to the parameter entry view.

```
[RelayCommand(CanExecute = "CanGenerateProblem")]
public void GenerateProblem()
```

Remarks

The command can only execute if Linear_Programming_Calculator_Desktop.ViewModels.StartViewModel.CanGenerateProblem() returns `true`.