UC003 Methods

Juan Parada Claro and Adrián Guirado Luna

Use case Number	UC003
Use case Name	User requests to change the content of a cell
Goal in context	-
Actors and their interest	● User. ● System being developed.
Preconditions	● The program shall be opened before. ● The user sees the options menu displayed by the system. ● The user introduces the option of the menu which corresponds to change the content of a cell.
Postconditions	• The system is able to change the content of the cells. • The system shows the menu.
Basic Flow	1. The user introduces the cell and the content of it, in this case a formula. 2. The system tokenizes differentiating the string between number, cell identifier, opening round bracket, closing round bracket, colon character, semi-colon character, comma, function name and range with the operators (+, -, *, /, ^, √). 3. The system parses the tokens and checks that the sequence meets the syntactical rules of regular arithmetic expressions. 4. The system generates the postfix deleting the parentheses that are not part of a function and consequently order the list of tokens. 5. The system converts the list of tokens to of components of formula. 6. The system checks if there are any circular dependences. 7. The system evaluates the postfix expression (see UC006). 8. The system recomputes the content of the cells that are referencing this cell.
Extension	1.b The user introduces a number: 1. The system recomputes the content of the cells if necessary. 1.c The user introduces a string: 1. The system checks if the content of this cell was referenced in a formula. 2. If the cell were referenced the system shall show an error and to not update the content of the cell. 3.a The user introduces an incorrect formula, and the parser notices it:

	1. The system will display an error message. 6.a The user introduces a formula with circular dependencies. 1. The system will display an error message. 2. The system deletes the objects that has been created. 7.a The user has introduced a division by 0. 1. The system will display an error message. 2. The system deletes the objects that has been created.
Special Requirements	All the data requirements shall be developed.

Step in the Use Case	1, and execute all the other methods steps
Name of the method	editCell
Initial list of arguments	content: A String object which could contain whatever content. key: A String object which includes the key
Returned Value	The method does not return any value.
Exceptions	BadFormulaException: Thrown if the sequence of tokens fails to meet the syntactical rules of regular arithmetic expressions, indicating an error in the formula's structure or logic. TokenNotMatchedException: Thrown if the tokenizer is not working. CircularDependenciesException: Thrown if the analysis reveals that the formula contains circular references, where a formula or cell indirectly or directly refers to itself, creating a loop that prevents resolution. ZeroDivisionException: Thrown if the stack detects a zero division. BadPostfixerImplementationException: Thrown if the Postfixer is not working.
Description	Retrieves the Cell associated with the provided TupleKey from a spreadsheet. The string on the content may not be the definitive one.
Assigned to class	SpreadsheetController: because it is an action requested by the user. GRASP Controller pattern.

Step in the Use Case	1
Name of the method	isFormula
Initial list of arguments	content: A String object which could contain whatever content.
Returned Value	The method does not return any value.
Exceptions	1.b IsNumberException: thrown when content is only a number. 1.c IsStringException: thrown when content is only a string.
Description	Validates the content introduced by the user, checking if it is a formula (starts with =), a number or a string.
Assigned to class	Formula: because is a little check that doesn't break the SRP pf Formula. Expert GRASP pattern.

Step in the Use Case	2
Name of the method	tokenizeFormula
Initial list of arguments	String formula: a String representing the mathematical or logical formula to be tokenized.
Returned Value	The method returns a List<Token>, which contains the individual tokens derived from the formula.
Exceptions	TokenNotMatchedException: Thrown if the tokenizer is not working.
Description	Analyzes and splits the input formula into discrete components or tokens.
Assigned to class	Tokenizer: because the FormulaCalculator Class has already the responsibility of evaluating the formula. SRP of FormulaCalculator. Pure fabrication GRASP pattern.

Step in the Use Case	3
Name of the method	parseFormula
Initial list of arguments	A List<Token> of tokens derived from a mathematical or logical formula.
Returned Value	The List<Token> of the argument but checked.
Exceptions	BadFormulaException: Thrown if the sequence of tokens fails to meet the syntactical rules of regular arithmetic expressions, indicating an error in the formula's structure or logic.
Description	Examines the tokenized version of a formula to verify its syntactic validity and organizes it into a structured expression suitable for further evaluation or execution. If the tokens do not properly align with established syntactical rules, resulting in an invalid expression, the method throws a BadFormulaException.
Assigned to class	Parser: because the FormulaCalculator Class has already the responsibility of evaluating the formula. SRP of FormulaCalculator. Pure fabrication GRASP pattern.

Step in the Use Case	4
Name of the method	generatePostfix
Initial list of arguments	A List<Token> representing the mathematical or logical formula to be converted into postfix notation.
Returned Value	The method returns a List<Token> without the parentheses that are not part of functions and in postfix form.
Exceptions	BadFormulaException: Thrown if the sequence of tokens fails to meet the syntactical rules of regular arithmetic expressions, indicating an error in the formula's structure or logic.
Description	Processes the input formula and converts it from its standard infix notation to postfix notation.
Assigned to class	Postfixer: because the FormulaCalculator Class has already the responsibility of evaluating the formula. SRP of FormulaCalculator. Pure fabrication GRASP pattern.

Step in the Use Case	5
Name of the method	generateFormula
Initial list of arguments	List<Token> formula: already syntactically checked and ordered properly. Spreadsheet spreadsheet: Spreadsheet being used. String expression: Expression to be treated. Cell currentCell: The cell that is equal the expression.
Returned Value	A Formula object complete with all the operands and operators inside.
Exceptions	This method does not have any exception.
Description	Examines the tokenized version of a formula to verify its syntactic validity and organizes it into a structured expression suitable for further evaluation or execution. If the tokens do not properly align with established syntactical rules, resulting in an invalid expression, the method throws a BadFormulaException.
Assigned to class	FormulaGenerator: because the Formula Class has already the responsibility of keeping in memory all the data about the formula. SRP of Formula. GRASP creation pattern. Pure fabrication GRASP pattern.

Step in the Use Case	6
Name of the method	checkCircularDependencies
Initial list of arguments	Formula formula: object that represents a mathematical or logical expression potentially involving references to other cells or formulas.
Returned Value	The method does not have any returned value (void).
Exceptions	CircularDependenciesException: Thrown if the analysis reveals that the formula contains circular references, where a formula or cell indirectly or directly refers to itself, creating a loop that prevents resolution.
Description	Examines a given formula to detect any circular dependencies among referenced cells or formulas. If there is any loop, throws CircularDependenciesException.
Assigned to class	FormulaCalculator: because the Formula Class has already the responsibility of keeping in memory all the data about the formula. SRP of Formula. Pure fabrication GRASP pattern.

Step in the Use Case	7
Name of the method	evaluatePostfix
Initial list of arguments	Formula formula: object that represents a mathematical or logical expression in postfix notation.
Returned Value	The method returns a Number object, which contains the result of the evaluated expression.
Exceptions	ZeroDivisionException: Thrown if the stack detects a zero division. BadPostfixerImplementationException: Thrown if Postfixer is not working.
Description	Takes a formula expressed in postfix notation and computes its value.

Assigned to class	FormulaCalculator: because the Formula Class has already the responsibility of keeping in memory all the data about the formula. SRP of Formula. Pure fabrication GRASP pattern.

Step in the Use Case	14
Name of the method	recalculateCells
Initial list of arguments	Cell cell: the cell has been changed and it has the information about their usage.
Returned Value	The method does not return any value.
Exceptions	CircularDependenciesException: Thrown if the analysis reveals that the formula contains circular references, where a formula or cell indirectly or directly refers to itself, creating a loop that prevents resolution. ZeroDivisionException: Thrown if the stack detects a zero division. BadPostfixerImplementationException: Thrown if Postfixer is not working.
Description	Triggers a recalculation of the values in the provided list of cells that it is contained in the changed Cell. This method ensures that all cells in the list are updated based on their current dependencies and formulas.
Assigned to class	FormulaCalculator: because the Formula Class has already the responsibility of keeping in memory all the data about the formula. SRP of Formula. Pure fabrication GRASP pattern.