CTTJFrame

Import Text File

Read text in

Pass to CTTLogic.AllPairs(String), receive back ArrayList<TestCaseObject> for test output

Print test output

Import Excel File

Read From Text Box

Radial Options

* All Pairs (set n to 2) – only option to start
* All N-th
  + Text Field for N value
* Complete
* Text output
* Excel output

CTTLogic

Creation()

AllNth(String, Int nthValue)

Split String Input at “\n” – receive Array [Strings] (if array.length <= 1 return error to user)

Iterate over Array [Strings] filling Array[CTTVariableObject(String)] cttVariableObjectsArray

sortArrray(cttVariableObjectsArray)

ArrayList<TestCaseObject(cttVariableObjectsArray .length)> testCasesFullList

FillVariableNames(cttVariableObjectsArray, testCasesFullList);

fillNthColumns(nth, testCasesFullList);

Int caseLocation, #MatchesRequired;

ArrayList<VariableValuesToMatch> matchesNeededList;

Int[] matchLocationsArray;

For(int variableLocation = nthValue; variableLocation < cttVariableObjectsArray.length; variableLocation ++)

{

For(int valueLocation = 0; valueLocation < cttVariableObjectsArray[variableLocation].GetNumberOfValues; valueLocation ++)

{

fillMatchesNeeded(matchesNeededList, cttVariableObjectsArray, variableLocation);

#matchesRequired = variableLocation;

caseLocatoin = 1;

matchLocationsArray = new Int[variableLocation];

While(!matchesNeededList.isEmpty())

If(caseLocation ==1)

While(answerFound(caseLocation, matchLocationsArray, matchesNeededList, #matchesRequired, testCasesFullList.length)!)

Make easiser

Else if(answerfound(caseLocation)!)

While(answerFound(caseLocation)!)

Make easiser

AddTestValue()

CleanMatchesNeeded();

}

}

Complete(String)

sortArray(Array needSorting)

sort by GetNumberofValues – largest first

fillMatchesNeeded(matchesNeededList, cttVariableObjectsArray, int VariableLocation)

matchesNeededList = new ArrayList<>();

for(int x = 0; x < VariableLocation; x++)

{

matchesNeededList.add(VariableValuesToMatch());

For(int y = 0; y < cttVariableObjectsArray[x].GetNumberOfValues; y ++)

{

matchesNeededList[x].Add(cttVariableObjectsArray[x].GetValue(y);

}

}

fillVariableNames(cttVariableObjects, testCasesFullList)

For(int x = 0; x < cttVariableObjects.length; x++)

{

testCasesFullList.get(0).SetValue(x , cttVariableObjects.get(x).GetName())

}

fillNthColumns(nth, testCasesFullList)

boolean AnswerFound(caseLocation, matchLocationsArray, matchesNeededList, #MatchesRequired, #TestCasesCurrently)

matchLocationsArray = Int[#MatchesRequired];

while(caseLocation<# of Tests && GetCount(matchLocationsArray) < #TestCasesCurrently)

if(TestCaseAlreadyFilled !)

if(matchesNeeded[last].get(0) == testcases(location in test array).get(last) || “ “ == testcases(location in test array).get(last))

for(each other variable already filled)

if(matchesNeeded[current].contains(testcases(location in Test Array).get(current) || testcases(location in Test Array.get(current) == “ “)

add to locations[]

return (!GetCount(locations)<#need to match)

int GetCount(matchLocationsArray)

int count = 0;

for(int x = 0; x <matchLocationsArray.length; x++)

if(matchlocationsArray[x] != null)

count +1

MakeEasier()

If(needToMatch != 1)

needToMatch -1

else

addRowToTestCases()

CleanMatchesNeeded()

For(each object from right)

If(empty)

Remove from list;

Else

Break;

addRowToTestCases()

CTTVariableObject

Creation(String input)

Split input at “:” – receive array [Strings]

String variableName = array[0]

Split array[1] at “,” – receive array [strings] = Array variableValues

Iterate over variableValues to remove extra “ “s

GetName()

Return variableName

GetNumberOfValues()

Return variableValues.length()

GetValue(Int location)

Return varaibleValues[location]

TestCaseObject

Creation(Int length)

Fill String[length] testCase with “ “

SetValue(Int location, String value)

testCase[location] = value;

GetValue(Int location)

Return testCase [location]

VariableValuesToMatch

Creation()

Initialize List values

Add(String value)

Values.add(value);

Remove(int location)

Values.remove(location);

Boolean Contains(Sting value)

Return values.contains(value);