

Portfolio: VBA Learning Project

Kaan Bora Karapınar

A look at the project

The problem set that I solved with VBA :

Level 3

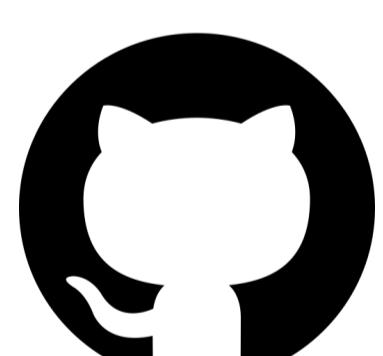
Catalysts

- Da John seine Mittagessen aus seinen täglichen Einnahmen bezahlt kann, muss er das gesamte Geld nicht sofort einzahlen. Er darf den Betrag in bis zu 4 Teilen bei der Bank einzahlen. Die Einzahlungen können auch an Folgetagen erfolgen.
- Eingabe:
 - {F <Tag> <Tagesumsatz>} {B <Tag> <Bankeinzahlung>}
 - Tag ... Tag im Jahr (1...365)
 - Tagesumsatz ... positive ganze Zahl
 - Bankeinzahlung ... positive ganze Zahl (eine Bankeinzahlung kann nur für max. einen Tag verwendet werden!)
- Ausgabe:
 - {Tag}
 - Tag ... Tag an dem der Tagesumsatz nicht komplett eingezahlt wurde.
 - Die Tage müssen aufsteigend sortiert sein.
- Beispiel:
 - Eingabe: F 1 200 F 2 170 B 1 100 B 2 80 B 2 15 B 2 100 B 3 70
 - Ausgabe: 2
- Since John is allowed to use parts of his daily revenue to buy lunch, he can pay the earnings divided in up to 4 parts into the bank. John is allowed to pay the parts also on the days after he got it from his customers.
- Input:
 - {F <day> <daily sales>} {B <day> <payment at the bank>}
 - Day... Day in the year (1...365)
 - Daily sales ... Positive integer
 - Payment at the bank ... Positive integer (one payment can NOT be used for two different daily sales!)
- Output:
 - {day}
 - day ... Day where the daily sales were not completely paid into the bank account.
 - The days must be in ascending order.
- Example:
 - Input: F 1 200 F 2 170 B 1 100 B 2 80 B 2 15 B 2 100 B 3 70
 - Output: 2

Day	Payment
1	100
2	80
2	15
2	100
3	70

© Catalysts, 2011

5



Github Repo for better code review:

KaanBoraKarapinar/VBABakeryAuditScripts

Project Details

Level 1

- Die Bäckerei Semmel King möchte nun überprüfen, ob John auch wirklich alle Einnahmen auf der Bank eingezahlt hat.
- Eingabe:**
 - {F <Tag> <Tagesumsatz>} {B <Tag> <Bankenzahlung>}
 - Tag ... Tag im Jahr (1...365)
 - Tagesumsatz ... positive ganze Zahl
 - Bankenzahlung ... positive ganze Zahl
- Ausgabe:**
 - {Tag}
 - Tag ... Nummer des Tages an dem der Tagesumsatz nicht komplett eingezahlt wurde.
 - Die Tage müssen aufsteigend sortiert sein, getrennt durch Leerzeichen
- Beispiel:**
 - Eingabe: F 1 200 F 2 170 B 1 200 B 2 100
 - Ausgabe: 2

The bakery wants to check if John really delivered all his earnings to the bank.

Input:

- {F <day> <daily sales>} {B <day> <payment at the bank>}
- Day... Day in the year (1...365)
- Daily sales ... Positive Integer
- Payment at the bank ... Positive Integer

Output:

- {day}
- day ... Number of the day where the daily sales were not completely paid into the bank account.
- The days must be in ascending order, separated by blanks

Example:

- Input: F 1 200 F 2 170 B 1 200 B 2 100
- Output: 2

© Catalysts, 2011 Catalysts Spring Challenge 2011 3

2

Catalysts

Level 2 ist wie Level 1, es kommt keine Funktionalität dazu. Nur die Eingaben sind deutlich länger.

Eingabe:

- {F <Tag> <Tagesumsatz>} {B <Tag> <Bankenzahlung>}
- Tag ... Tag im Jahr (1...365)
- Tagesumsatz ... positive ganze Zahl
- Bankenzahlung ... positive ganze Zahl

Ausgabe:

- {Tag}
- Tag ... Nummer des Tages an dem der Tagesumsatz nicht komplett eingezahlt wurde.
- Die Tage müssen aufsteigend sortiert sein, getrennt durch Leerzeichen

- The level 2 is like the first level, no additional functionality is added. The inputs are much larger.
- Input:**
 - {F <day> <daily sales>} {B <day> <payment at the bank>}
 - Day... Day in the year (1...365)
 - Daily sales ... Positive Integer
 - Payment at the bank ... Positive Integer
- Output:**
 - {day}
 - day ... Number of the day where the daily sales were not completely paid into the bank account.
 - The days must be in ascending order, separated by blanks

© Catalysts, 2011 Catalysts Spring Challenge 2011 3

Level 3

Catalysts

- Da John seine Mittagessen aus seinen täglichen Einnahmen bezahlt kann, muss er das gesamte Geld nicht einzahlen. Er darf den Betrag in bis zu 4 Teilen bei der Bank einzahlen. Die Einzahlungen können auch an Folgetagen erfolgen.
- Eingabe:**
 - {F <Tag> <Tagesumsatz>} {B <Tag> <Bankenzahlung>}
 - Tag ... Tag im Jahr (1...365)
 - Tagesumsatz ... positive ganze Zahl
 - Bankenzahlung ... positive ganze Zahl (eine Bankenzahlung kann nur für max. einen Tag verwendet werden!)
- Ausgabe:**
 - {Tag}
 - Tag ... Tag an dem der Tagesumsatz nicht komplett eingezahlt wurde.
 - Die Tage müssen aufsteigend sortiert sein.
- Beispiel:**
 - Eingabe: F 1 200 F 2 170 B 1 100 B 2 80 B 2 15 B 2 100 B 3 70
 - Ausgabe: 2

Day	Salary
1	200
2	170

Day	Payment
1	100
2	80
2	15
2	100
3	70

Since John is allowed to use parts of his daily revenue to buy lunch, he can pay the earnings divided in up to 4 parts into the bank. John is allowed to pay the parts also on the days after he got it from his customers.

Input:

- {F <day> <daily sales>} {B <day> <payment at the bank>}
- Day... Day in the year (1...365)
- Daily sales ... Positive Integer
- Payment at the bank ... Positive Integer (one payment can NOT be used for two different daily sales!)

Output:

- {day}
- day ... Day where the daily sales were not completely paid into the bank account.
- The days must be in ascending order.

Example:

- Input: F 1 200 F 2 170 B 1 100 B 2 80 B 2 15 B 2 100 B 3 70
- Output: 2

© Catalysts, 2011 Catalysts Spring Challenge 2011 3

Catalyst / Cloudflight Catcoder has released several algorithmic and data management quizzes. To build my VBA skills, I selected a finance-focused quiz and implemented a solution.

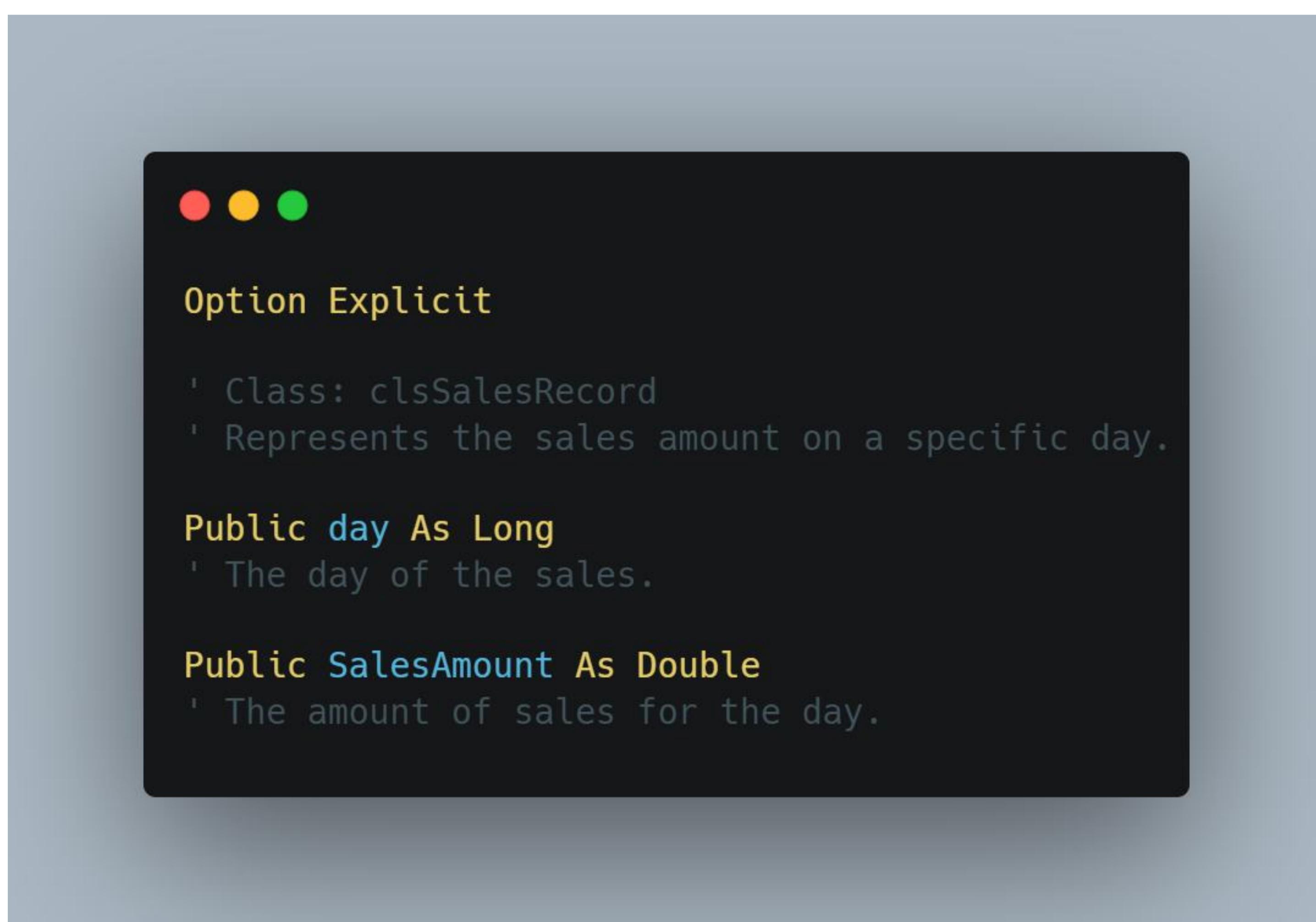
My task was to identify discrepancies in gross amounts and bank payments by comparing them using manual records.

In level two, while the core logic remained the same, the dataset became significantly larger, increasing the complexity of the task.

In level three, the challenge was to identify matching dates for untagged late-day payments, ensuring accuracy in tracking delays across each day.

Source Code

Initially, each daily record contained a single `clsSalesRecord` and a single `clsPaymentRecord`. In level three, however, up to four `clsPaymentRecord` entries can now be assigned per day. Additionally, it's no longer assumed that all payments correspond directly to the gross sale of the payment day, adding another layer of complexity to tracking and validation.



```
Option Explicit

' Class: clsSalesRecord
' Represents the sales amount on a specific day.

Public day As Long
' The day of the sales.

Public SalesAmount As Double
' The amount of sales for the day.
```



```
Option Explicit

' Class: clsPaymentRecord
' Represents a payment made on a specific day.

Public day As Long
' The day the payment was made.

Public Payment As Double
' The amount of the payment.

Public IsUsed As Boolean
' Indicates if the payment has been used.

Public PaymentID As String
' A unique identifier for the payment.

Private Sub Class_Initialize()
    ' Initializes IsUsed to False when the class is created.
    IsUsed = False
End Sub
```

Source Code: clsDailyRecord

The clsDailyRecord class contains the primary strategy for identifying basic discrepancies, as discrepancies are initially considered on a daily basis. However, for more complex calculations, such as those in level three, the operations are handled in a separate class, clsDays, which represents and manages all daily records and their corresponding days. This structure allows for more advanced, cross-day analysis of discrepancies.

```
1 Option Explicit
2
3 ' Class: clsDailyRecord
4 ' Represents a daily record containing payments and sales.
5 ' Calculates discrepancies between sales and payments.
6
7 Public day As Long
8 ' The day number (e.g., 1, 2, 3, etc.)
9
10 Private PaymentRecord() As clsPaymentRecord
11 ' Array of PaymentRecords for the day.
12
13 Private EffectivePaymentRecord() As clsPaymentRecord
14 ' Array of Effective PaymentRecords used for discrepancy calculation.
15
16 Public TotalNumberOfEffectivePayments As Long
17 ' Total number of effective payments for the day.
18
19 Public TotalEffectivePaymentAmount As Long
20 ' Total amount of effective payments for the day.
21
22 Public SalesRecord As clsSalesRecord
23 ' SalesRecord for the day.
24
25 Public Discrepancy As Double
26 ' The discrepancy amount for the day.
27
28 Public IsDiscrepant As Boolean
29 ' Indicates if there is a discrepancy on this day.
30
31 Public ThereIsSavedPayment As Boolean
32 ' Indicates if there are any saved payments for the day.
33
34 Public TotalPaymentAmount As Double
35 ' Total amount of payments for the day.
36
37 Private TotalNumberOfPayment As Long
38 ' Total number of payments for the day.
39
40 Sub ToString()
41 ' Displays information about the day's record.
42 On Error Resume Next
43 MsgBox "In day: " & day & " payment: " & PaymentRecord.Payment & " sales: " &
44 SalesRecord.SalesAmount &
45 " isDiscrepant and Amount: " & IsDiscrepant & " " & Discrepancy
46 End Sub
47
48 Public Property Get getReadOnlyPaymentRecords() As Long
49 ' Getter for PaymentRecord array (read-only).
50 paymentrecords = PaymentRecord
51 End Property
52
53 Public Sub addToPaymentRecordsArray(value As clsPaymentRecord)
54 ' Adds a PaymentRecord to the PaymentRecord array.
55 ThereIsSavedPayment = True
56 TotalNumberOfPayment = TotalNumberOfPayment + 1
57 ReDim Preserve PaymentRecord(0 To TotalNumberOfPayment - 1)
58 Set PaymentRecord(TotalNumberOfPayment - 1) = value
59 TotalPaymentAmount = TotalPaymentAmount + value.Payment
60 End Sub
61
62 Public Sub AddEffectivePayment(effectivePayment As clsPaymentRecord)
63 ' Adds a PaymentRecord to the EffectivePaymentRecord array if it hasn't been used
64 yet.
65 If effectivePayment.IsUsed = False Then
66 TotalNumberOfEffectivePayments = TotalNumberOfEffectivePayments + 1
67 Dim lastLegalIndex As Long
68 lastLegalIndex = TotalNumberOfEffectivePayments - 1
69 ReDim Preserve EffectivePaymentRecord(0 To lastLegalIndex)
70 Set EffectivePaymentRecord(lastLegalIndex) = effectivePayment
71 TotalEffectivePaymentAmount = TotalEffectivePaymentAmount + effectivePayment.Payment
72 effectivePayment.IsUsed = True
73
74 Else
75 ' MsgBox "Used payments cannot be effective"
76 End If
77 End Sub
78
79 Public Sub ApplyAllPaymentsOfDayAsEffective()
80 ' Makes all payments of the day effective.
81 Dim current As Variant
82 Dim tempRecord As clsPaymentRecord
83 For Each current In PaymentRecord
84 Set tempRecord = current
85 AddEffectivePayment tempRecord
86 Next
87 End Sub
88
89 Public Sub ApplyAllRemainingPaymentsOfDayAsEffective()
90 ' Makes all unused payments of the day effective.
91 Dim current As Variant
92 Dim tempRecord As clsPaymentRecord
93 For Each current In PaymentRecord
94 If current.IsUsed = False Then
95 Set tempRecord = current
96 AddEffectivePayment tempRecord
97 End If
98 Next
99 End Sub
100
101 Public Sub makeBiggestPaymentEffective()
102 ' Makes the biggest unused payment effective.
103 Dim tempIndex As Long
104 tempIndex = IndexOfBiggestNonUsedPayment
105 If tempIndex <> -1 Then
106 AddEffectivePayment PaymentRecord(tempIndex)
107 End If
108 End Sub
109
110 Public Sub CalculateDiscrepancyWithCurrentEffectivePayments()
111 ' Calculates discrepancy based on current effective payments and sales record.
112 If SalesRecord Is Nothing Then
113 IsDiscrepant = True
114 Discrepancy = -TotalEffectivePaymentAmount
115 If TotalEffectivePaymentAmount = 0 Then
116 IsDiscrepant = False
117 Discrepancy = -TotalEffectivePaymentAmount
118 End If
119 Exit Sub
120 End If
121
122 If SalesRecord.SalesAmount = TotalEffectivePaymentAmount Then
123 Discrepancy = 0
124 IsDiscrepant = False
125 Else
126 IsDiscrepant = True
127 Discrepancy = SalesRecord.SalesAmount - TotalEffectivePaymentAmount
128 End If
129 End Sub
130
131 Public Function IndexOfBiggestNonUsedPayment() As Long
132 ' Returns the index of the biggest non-used payment.
133 Dim i As Long
134 Dim maxIndex As Long
135 Dim maxValue As Double
136 maxIndex = -1
137 maxValue = -1
138 For i = 0 To UBound(PaymentRecord)
139 If Not PaymentRecord(i).IsUsed And PaymentRecord(i).Payment > maxValue Then
140 maxValue = PaymentRecord(i).Payment
141 maxIndex = i
142 End If
143 End Function
144
145 Next i
146
147 IndexOfBiggestNonUsedPayment = maxIndex
148 End Function
149
150 Public Function getNumberOfNotUsedPayments() As Long
151 ' Returns the number of payments that have not been used.
152 Dim count As Long
153 count = 0
154 For Each Payment In PaymentRecord
155 If Payment.IsUsed = False Then
156 count = count + 1
157 End If
158 Next
159 getNumberOfNotUsedPayments = count
160 End Function
161
162 Sub BasicCheckDiscrepancy()
163 ' Performs a basic discrepancy check between sales and total payments.
164 If SalesRecord Is Nothing And Not ThereIsSavedPayment Then
165 IsDiscrepant = False
166 Exit Sub
167 ElseIf SalesRecord Is Nothing Then
168 IsDiscrepant = True
169 Discrepancy = -TotalPaymentAmount
170 Exit Sub
171 ElseIf Not ThereIsSavedPayment Then
172 IsDiscrepant = True
173 Discrepancy = SalesRecord.SalesAmount
174 Exit Sub
175 End If
176
177 If SalesRecord.SalesAmount = TotalPaymentAmount Then
178 IsDiscrepant = False
179 Else
180 IsDiscrepant = True
181 Discrepancy = SalesRecord.SalesAmount - TotalPaymentAmount
182 End If
183 End Sub
184
185 Sub ComplexCheckDiscrepancy(effectivePayment As Double)
186 ' Performs a discrepancy check using effective payments.
187 If SalesRecord Is Nothing And Not ThereIsSavedPayment Then
188 IsDiscrepant = False
189 Exit Sub
190 ElseIf SalesRecord Is Nothing Then
191 IsDiscrepant = True
192 Discrepancy = -effectivePayment
193 Exit Sub
194 ElseIf Not ThereIsSavedPayment Then
195 IsDiscrepant = True
196 Discrepancy = SalesRecord.SalesAmount
197 Exit Sub
198 End If
199
200 If SalesRecord.SalesAmount = effectivePayment Then
201 IsDiscrepant = False
202 Else
203 IsDiscrepant = True
204 Discrepancy = SalesRecord.SalesAmount - effectivePayment
205 End If
206 End Sub
207
208 Sub CheckDiscrepancy()
209 ' Empty Sub (possibly for future use).
210 End Sub
211
212 Private Sub Class_Initialize()
213 ' Initializes arrays and counters.
214 ReDim PaymentRecord(0 To 0)
215 TotalNumberOfPayment = 0
216 TotalPaymentAmount = 0
```

Source Code: clsDays

clsDays contains the main business logic implementation. It manages data using Scripting.Dictionary objects. The mainDict holds all clsDailyRecord objects, which encapsulate originalclsSalesRecord and clsPaymentRecord objects. The centralPaymentManagement dictionary references all created payment objects, while freePaymentMethod keeps track of unused payment methods for the day-payment assignment task introduced in Level 3.

Dictionary objects maintain data in the given order, and the data provided by CCC was chronologically ordered. **However, I implemented a QuickSort algorithm to handle cases where the data might not be ordered.** Later, I enhanced and migrated the sorting logic to HelperMethods to include additional functionality, such as reversing functions and **distinguishing between shallow and deep copies**. Despite this migration, a basic sort method remains within the class.

```
1 Option Explicit
2
3 ' Class: DAYS
4 ' Manages daily records, payments, and discrepancies across multiple days.
5
6 Private mainDict As Scripting.Dictionary
7 ' Dictionary to hold clsDailyRecord objects, keyed by day.
8
9 Private centralPaymentManagement As Scripting.Dictionary
10 ' Central management of all payments, keyed by PaymentID.
11
12 Private freePaymentMethods As Scripting.Dictionary
13 ' Dictionary of payments that are not yet used, keyed by PaymentID.
14
15 Public Sub AddPaymentRecordToDay(day As Long, PaymentRecord As clsPaymentRecord)
16     ' Adds a PaymentRecord to a day's record.
17     If Not mainDict.Exists(day) Then
18         Dim DRtoSave As clsDailyRecord
19         Set DRtoSave = New clsDailyRecord
20         DRtoSave.day = day
21         DRtoSave.addToPaymentRecordsArray PaymentRecord
22         mainDict.Add day, DRtoSave
23         AddReferenceOfPaymentRecordToTheCentralManagement PaymentRecord
24     Else
25         mainDict(day).addToPaymentRecordsArray PaymentRecord
26         AddReferenceOfPaymentRecordToTheCentralManagement PaymentRecord
27     End If
28 End Sub
29
30 Public Sub AddReferenceOfPaymentRecordToTheCentralManagement(PaymentRecord As
31     clsPaymentRecord)
32     ' Adds a PaymentRecord to central management and free payments.
33     PaymentRecord.PaymentID = "D" & PaymentRecord.day & "A" & PaymentRecord.Payment
34     centralPaymentManagement.Add PaymentRecord.PaymentID, PaymentRecord
35     freePaymentMethods.Add PaymentRecord.PaymentID, PaymentRecord
36 End Sub
37
38 Public Sub AddSalesRecordToDay(day As Long, SalesRecord As clsSalesRecord)
39     ' Adds a SalesRecord to a day's record.
40     If Not mainDict.Exists(day) Then
41         Dim DRtoSave As clsDailyRecord
42         Set DRtoSave = New clsDailyRecord
43         DRtoSave.day = day
44         Set DRtoSave.SalesRecord = SalesRecord
45         mainDict.Add day, DRtoSave
46     Else
47         If Not mainDict(day).SalesRecord Is Nothing Then
48             MsgBox ("Wrong Raw Data: Two Sales Records for same day: " & day)
49             Exit Sub
50         End If
51         Set mainDict(day).SalesRecord = SalesRecord
52     End Sub
53
54 Property Get Dictionary() As Scripting.Dictionary
55     ' Getter for the main dictionary of days.
56     Set Dictionary = mainDict
57 End Property
58
59 Public Sub CheckAllDiscrepancy()
60     ' Checks discrepancies for all days.
61     Dim current As Variant
62     Dim currentDay As clsDailyRecord
63
64     For Each current In mainDict.Keys
65         Set currentDay = mainDict(current)
66
67         If current = 1 Then
68             currentDay.ApplyAllPaymentsOfDayAsEffective
69             currentDay.CalculateDiscrepancyWithCurrentEffectivePayments
70             ForceRefreshFreePaymentRecords
71         Else
72             If Not currentDay.SalesRecord Is Nothing Then
```

```
73             currentDay.ApplyAllRemainingPaymentsOfDayAsEffective
74             ForceRefreshFreePaymentRecords
75         End If
76
77         currentDay.CalculateDiscrepancyWithCurrentEffectivePayments
78
79         If currentDay.IsDiscrepant Then
80             Dim discrepancyFiller As clsPaymentRecord
81             Set discrepancyFiller = FindPaymentWithPaymentAmount(currentDay,
82             Discrepancy)
83
84             If Not discrepancyFiller Is Nothing Then
85                 currentDay.AddEffectivePayment discrepancyFiller
86                 currentDay.CalculateDiscrepancyWithCurrentEffectivePayments
87                 ForceRefreshFreePaymentRecords
88             End If
89         End If
90     Next
91 End Sub
92
93 Public Sub applyAllRemaining()
94     ' Applies all remaining payments as effective.
95     Dim current As Variant
96     Dim currentDay As clsDailyRecord
97
98     For Each current In mainDict.Keys
99         Set currentDay = mainDict(current)
100        currentDay.ApplyAllRemainingPaymentsOfDayAsEffective
101    Next
102 End Sub
103
104 Public Sub ForceRefreshFreePaymentRecords()
105     ' Refreshes the list of free payment methods.
106     Set freePaymentMethods = New Scripting.Dictionary
107     Dim currentID As Variant
108
109     For Each currentID In centralPaymentManagement.Keys
110         If centralPaymentManagement(currentID).IsUsed = False Then
111             freePaymentMethods.Add currentID, centralPaymentManagement(currentID)
112         End If
113     Next
114 End Sub
115
116 Public Function FindPaymentWithPaymentAmount(Amount As Long) As clsPaymentRecord
117     ' Finds a free payment record with a specific payment amount.
118     On Error Resume Next
119     Dim currentID As Variant
120     Dim returnRecord As clsPaymentRecord
121
122     For Each currentID In freePaymentMethods.Keys
123         If freePaymentMethods(currentID).IsUsed = False And freePaymentMethods(
124             currentID).Payment = Amount Then
125             Set returnRecord = freePaymentMethods(currentID)
126         End If
127     Next
128
129     Set FindPaymentWithPaymentAmount = returnRecord
130 End Function
131
132 Public Sub BasicCheckAllDiscrepancy()
133     ' Performs basic discrepancy checks for all days.
134     Dim currentDay As Variant
135
136     For Each currentDay In mainDict.Keys
137         mainDict(currentDay).BasicCheckDiscrepancy
138     Next
139 End Sub
140
141 Public Sub checkAllDiscrepancyAllowOnlyNextDay()
142     ' Checks discrepancies, allowing payments from the next day.
143     Dim currentDay As Variant
```

Source Code: clsDays

```
144  
145  For Each currentDay In mainDict.Keys  
146      On Error Resume Next  
147      Dim previousDay As Long  
148      Dim effectivePaymentToday As Double  
149      Dim calculativePaymentToday As Double  
150  
151      calculativePaymentToday = mainDict(currentDay).TotalPaymentAmount  
152      previousDay = currentDay - 1  
153  
154      If mainDict.Exists(previousDay) Then  
155          If mainDict(previousDay).IsDiscrepant Then  
156              Dim discrepancyPreviousDay As Long  
157              discrepancyPreviousDay = mainDict(previousDay).Discrepancy  
158  
159          If calculativePaymentToday >= discrepancyPreviousDay Then  
160              mainDict(previousDay).Discrepancy = 0  
161              mainDict(previousDay).IsDiscrepant = False  
162              effectivePaymentToday = calculativePaymentToday -  
163                  discrepancyPreviousDay  
164          Else  
165              mainDict(previousDay).Discrepancy = discrepancyPreviousDay -  
166                  calculativePaymentToday  
167              effectivePaymentToday = 0  
168      End If  
169  Else  
170      effectivePaymentToday = calculativePaymentToday  
171  End If  
172  
173  mainDict(currentDay).ComplexCheckDiscrepancy effectivePaymentToday  
174  
175  Next  
176 End Sub  
177  
178 Function getAnArrayOfSortedKeys() As Long  
179  ' Returns an array of sorted keys (days).  
180  Dim k As Variant  
181  Dim i0 As Long  
182  i0 = 0  
183  Dim SortedArray() As Long  
184  
185  ReDim SortedArray(0 To mainDict.Count - 1)  
186  
187  For Each k In mainDict.Keys()  
188      SortedArray(i0) = k  
189      i0 = i0 + 1  
190  Next  
191  
192  quicksort SortedArray, 0, UBound(SortedArray)  
193  getAnArrayOfSortedKeys = SortedArray  
194 End Function  
195  
196 Sub quicksort(a() As Long, left As Long, right As Long)  
197  ' QuickSort algorithm.  
198  Dim P As Long  
199  
200  If left < right Then  
201      P = partition(a, left, right)  
202      quicksort a, left, P  
203      quicksort a, P + 1, right  
204  End If  
205 End Sub  
206  
207 Function partition(a() As Long, left As Long, right As Long) As Long  
208  ' Partition function for QuickSort.  
209  Dim Pivot As Long  
210  Pivot = a(left)  
211  Dim pl As Long  
212  Dim pr As Long  
213  pl = left  
214  pr = right  
215  
216  Do While pl < pr  
217      Do While a(pl) <= Pivot And pl < right  
218          pl = pl + 1  
219      Loop  
220  
221      Do While a(pr) > Pivot And pr > left  
222          pr = pr - 1  
223      Loop  
224  
225      If pl < pr Then  
226          swap a, pl, pr  
227      End If  
228  Loop  
229  
230  swap a, left, pr  
231  partition = pr  
232 End Function  
233  
234 Sub swap(a() As Long, i As Long, j As Long)  
235  ' Swaps two elements in an array.  
236  Dim temp As Long  
237  temp = a(i)  
238  a(i) = a(j)  
239  a(j) = temp  
240 End Sub  
241  
242 Sub WriteData()  
243  ' Writes data to the worksheet starting at cell A10.  
On Error Resume Next  
244  Dim startIndex As Range  
245  Set startIndex = Range("A10")  
246  Dim ColumnPointer As Long  
247  ColumnPointer = 1  
248  Dim sortedKeys As Variant  
249  sortedKeys = Me.getAnArrayOfSortedKeys  
250  Dim Key As Variant  
251  
Range("A10", "D100").Value = ""  
252  
253 For Each Key In sortedKeys  
254     startIndex.Cells(ColumnPointer, 1).Value = mainDict(Key).day  
255     startIndex.Cells(ColumnPointer, 2).Value = mainDict(Key).SalesRecord  
256     SalesAmount  
257     startIndex.Cells(ColumnPointer, 3).Value = mainDict(Key).TotalPaymentAmount  
258     startIndex.Cells(ColumnPointer, 4).Value = mainDict(Key).Discrepancy  
259     ColumnPointer = ColumnPointer + 1  
260 Next Key  
261 End Sub  
262  
263 Sub WriteAuditResult()  
264  ' Writes the audit results to cell B4.  
265  Dim Index As Range  
266  Set Index = Range("B4")  
267  Dim Result As String  
268  Dim sortedKeys As Variant  
269  sortedKeys = Me.getAnArrayOfSortedKeys  
270  Dim Key As Variant  
271  
272 For Each Key In sortedKeys  
273     If mainDict(Key).IsDiscrepant Then  
274         Result = Result & " " & mainDict(Key).day  
275     End If  
276 Next Key  
277  
278 Index.Value = Result  
279 End Sub  
280  
281 Sub ReverseArray(a() As Long)  
282  ' Reverses the elements of an array.  
283  Dim left As Long  
284  Dim right As Long  
285  left = LBound(a)  
286  
287  right = UBound(a)  
288  
289  Do While left < right  
290      swap a, left, right  
291      left = left + 1  
292      right = right - 1  
293  Loop  
294 End Sub  
295  
296 Private Sub Class_Initialize()  
297  ' Initializes the dictionaries.  
298  Set mainDict = New Scripting.Dictionary  
299  Set centralPaymentManagement = New Scripting.Dictionary  
300  Set freePaymentMethods = New Scripting.Dictionary  
301 End Sub  
302  
303 Private Sub Class_Terminate()  
304  ' Cleanup code if necessary.  
305 End Sub  
306
```

Source Code: clsDays

```
144  
145  For Each currentDay In mainDict.Keys  
146      On Error Resume Next  
147      Dim previousDay As Long  
148      Dim effectivePaymentToday As Double  
149      Dim calculativePaymentToday As Double  
150  
151      calculativePaymentToday = mainDict(currentDay).TotalPaymentAmount  
152      previousDay = currentDay - 1  
153  
154      If mainDict.Exists(previousDay) Then  
155          If mainDict(previousDay).IsDiscrepant Then  
156              Dim discrepancyPreviousDay As Long  
157              discrepancyPreviousDay = mainDict(previousDay).Discrepancy  
158  
159          If calculativePaymentToday >= discrepancyPreviousDay Then  
160              mainDict(previousDay).Discrepancy = 0  
161              mainDict(previousDay).IsDiscrepant = False  
162              effectivePaymentToday = calculativePaymentToday -  
163                  discrepancyPreviousDay  
164          Else  
165              mainDict(previousDay).Discrepancy = discrepancyPreviousDay -  
166                  calculativePaymentToday  
167              effectivePaymentToday = 0  
168      End If  
169  Else  
170      effectivePaymentToday = calculativePaymentToday  
171  End If  
172  
173  mainDict(currentDay).ComplexCheckDiscrepancy effectivePaymentToday  
174  
175  Next  
176 End Sub  
177  
178 Function getAnArrayOfSortedKeys() As Long  
179  ' Returns an array of sorted keys (days).  
180  Dim k As Variant  
181  Dim i0 As Long  
182  i0 = 0  
183  Dim SortedArray() As Long  
184  
185  ReDim SortedArray(0 To mainDict.Count - 1)  
186  
187  For Each k In mainDict.Keys()  
188      SortedArray(i0) = k  
189      i0 = i0 + 1  
190  Next  
191  
192  quicksort SortedArray, 0, UBound(SortedArray)  
193  getAnArrayOfSortedKeys = SortedArray  
194 End Function  
195  
196 Sub quicksort(a() As Long, left As Long, right As Long)  
197  ' QuickSort algorithm.  
198  Dim P As Long  
199  
200  If left < right Then  
201      P = partition(a, left, right)  
202      quicksort a, left, P  
203      quicksort a, P + 1, right  
204  End If  
205 End Sub  
206  
207 Function partition(a() As Long, left As Long, right As Long) As Long  
208  ' Partition function for QuickSort.  
209  Dim Pivot As Long  
210  Pivot = a(left)  
211  Dim pl As Long  
212  Dim pr As Long  
213  pl = left  
214  pr = right  
215  
216  Do While pl < pr  
217      Do While a(pl) <= Pivot And pl < right  
218          pl = pl + 1  
219      Loop  
220  
221      Do While a(pr) > Pivot And pr > left  
222          pr = pr - 1  
223      Loop  
224  
225      If pl < pr Then  
226          swap a, pl, pr  
227      End If  
228  Loop  
229  
230  swap a, left, pr  
231  partition = pr  
232 End Function  
233  
234 Sub swap(a() As Long, i As Long, j As Long)  
235  ' Swaps two elements in an array.  
236  Dim temp As Long  
237  temp = a(i)  
238  a(i) = a(j)  
239  a(j) = temp  
240 End Sub  
241  
242 Sub WriteData()  
243  ' Writes data to the worksheet starting at cell A10.  
On Error Resume Next  
244  Dim startIndex As Range  
245  Set startIndex = Range("A10")  
246  Dim ColumnPointer As Long  
247  ColumnPointer = 1  
248  Dim sortedKeys As Variant  
249  sortedKeys = Me.getAnArrayOfSortedKeys  
250  Dim Key As Variant  
251  
Range("A10", "D100").Value = ""  
252  
253 For Each Key In sortedKeys  
254     startIndex.Cells(ColumnPointer, 1).Value = mainDict(Key).day  
255     startIndex.Cells(ColumnPointer, 2).Value = mainDict(Key).SalesRecord  
256     SalesAmount  
257     startIndex.Cells(ColumnPointer, 3).Value = mainDict(Key).TotalPaymentAmount  
258     startIndex.Cells(ColumnPointer, 4).Value = mainDict(Key).Discrepancy  
259     ColumnPointer = ColumnPointer + 1  
260 Next Key  
261 End Sub  
262  
263 Sub WriteAuditResult()  
264  ' Writes the audit results to cell B4.  
265  Dim Index As Range  
266  Set Index = Range("B4")  
267  Dim Result As String  
268  Dim sortedKeys As Variant  
269  sortedKeys = Me.getAnArrayOfSortedKeys  
270  Dim Key As Variant  
271  
272 For Each Key In sortedKeys  
273     If mainDict(Key).IsDiscrepant Then  
274         Result = Result & " " & mainDict(Key).day  
275     End If  
276 Next Key  
277  
278 Index.Value = Result  
279 End Sub  
280  
281 Sub ReverseArray(a() As Long)  
282  ' Reverses the elements of an array.  
283  Dim left As Long  
284  Dim right As Long  
285  left = LBound(a)  
286  
287  right = UBound(a)  
288  
289  Do While left < right  
290      swap a, left, right  
291      left = left + 1  
292      right = right - 1  
293  Loop  
294 End Sub  
295  
296 Private Sub Class_Initialize()  
297  ' Initializes the dictionaries.  
298  Set mainDict = New Scripting.Dictionary  
299  Set centralPaymentManagement = New Scripting.Dictionary  
300  Set freePaymentMethods = New Scripting.Dictionary  
301 End Sub  
302  
303 Private Sub Class_Terminate()  
304  ' Cleanup code if necessary.  
305 End Sub  
306
```

Source Code: HelperMethods

```
1 Option Explicit
2
3 ' Module: HelperMethods
4 ' Contains helper functions for array sorting and manipulation.
5
6 Function getSortedArrayAscending(ArrayToSort() As Long) As Long
7     ' Returns a sorted array in ascending order.
8     Dim SortedArray() As Long
9     SortedArray = CloneArray(ArrayToSort)
10    quicksort SortedArray, 0, UBound(SortedArray)
11    getSortedArrayAscending = SortedArray
12 End Function
13
14 Function getSortedArrayDescending(ArrayToSort() As Long) As Long
15     ' Returns a sorted array in descending order.
16     Dim SortedArray() As Long
17     SortedArray = CloneArray(ArrayToSort)
18     quicksort SortedArray, 0, UBound(SortedArray)
19     ReverseArray SortedArray
20     sortArray = SortedArray
21 End Function
22
23 Sub sortGivenArrayAscending(ArrayToSort() As Long)
24     ' Sorts the given array in ascending order.
25     quicksort ArrayToSort, 0, UBound(ArrayToSort)
26 End Sub
27
28 Sub sortGivenArrayDescending(ArrayToSort() As Long)
29     ' Sorts the given array in descending order.
30     quicksort ArrayToSort, 0, UBound(ArrayToSort)
31     ReverseArray ArrayToSort
32 End Sub
33
34 Sub quicksort(a() As Long, left As Long, right As Long)
35     ' QuickSort algorithm.
36     Dim P As Long
37
38     If left < right Then
39         P = partition(a, left, right)
40         quicksort a, left, P
41         quicksort a, P + 1, right
42     End If
43 End Sub
44
45 Function partition(a() As Long, left As Long, right As Long) As Long
46     ' Partition function for QuickSort.
47     Dim Pivot As Long
48     Pivot = a(left)
49     Dim pl As Long
50     Dim pr As Long
51     pl = left
52     pr = right
53
54     Do While pl < pr
55         Do While a(pl) <= Pivot And pl < right
56             pl = pl + 1
57         Loop
58
59         Do While a(pr) > Pivot And pr > left
60             pr = pr - 1
61         Loop
62
63         If pl < pr Then
64             swap a, pl, pr
65         End If
66     Loop
67
68     swap a, left, pr
69     partition = pr
70 End Function
71
72 Sub swap(a() As Long, i As Long, j As Long)
73     ' Swaps two elements in an array.
74     Dim temp As Long
75     temp = a(i)
76     a(i) = a(j)
77     a(j) = temp
78 End Sub
79
80 Sub ReverseArray(a() As Long)
81     ' Reverses the elements of an array.
82     Dim left As Long
83     Dim right As Long
84     left = LBound(a)
85     right = UBound(a)
86
87     Do While left < right
88         swap a, left, right
89         left = left + 1
90         right = right - 1
91     Loop
92 End Sub
93
94 Function CloneArray(arr() As Long) As Long
95     ' Clones an array.
96     Dim newArray() As Long
97     ReDim newArray(LBound(arr) To UBound(arr))
98     Dim i As Long
99
100    For i = LBound(arr) To UBound(arr)
101        newArray(i) = arr(i)
102    Next i
103
104    CloneArray = newArray
105
106 End Function
```

Thank you for your
attention.

Vielen Dank für
Ihre Aufmerksamkeit.

I look forward to your feedback.
Ich freue mich auf ihre Rückmeldung.

Kaan Bora Karapınar