

あみすき堂

競馬の統計・データ分析を  
したい人のための  
JRA-VANデータラボ講座



著 万灯あお

# 目次

はじめに	3
8月上旬に行われるJV-Dataの仕様変更について (Ver.4.9.0)	4
第一章 事前知識	5
1-1: プログラミングについて	5
・アルゴリズム	5
・ライブラリ	6
・変数と配列	7
・データ型と宣言	8
・関数とプロシージャ	9
・クラスと New 演算子	10
・スコープ	11
・命名規則	12
・イベント	12
・Visual Studio、vb.net について	12
1-2: ソフト開発について	14
・JRA-VAN とは	14
・蓄積系ソフトか非蓄積系ソフトか	14
・蓄積系データか速報系 (リアルタイム系) データか	15
・蓄積系ソフトのデータ取得	15
1-3: 今回の開発で使う機能	16
・コントロール	16
・プロパティ	16
・ボタン	16
・メッセージボックス	17
・プログレスバー	17
・デバッグ	18
・コメントアウト	18

第二章 ソフト開発	2 0
2-1：全体像の説明	2 0
・ Visual Studio のインストール	2 0
・ ダウンロードしておくもの	2 0
・ 簡単な流れの説明	2 1
・ 全体のソースコード	2 2
・ プロジェクトの作成	3 0
・ 適宜確認するもの	3 0
・ 質問掲示板について	3 1
2-2：諸々の機能	3 2
・ Access 2016 の追加	3 2
・ JV-Link コントロールの追加	3 2
・ JV-Link の設定変更機能 (JVSetUIProperties)	3 4
・ JV-Link の初期化機能 (JVInit)	3 5
・ DB 作成機能 (CreateDB)	3 7
・ DB 削除機能 (KillDB)	4 0
・ DB 最適化機能 (CompactDB)	4 2
・ DB との接続機能 (ConnectDBF)	4 3
・ DB との切断機能 (CloseDBF)	4 5
・ JV-Link 終了機能 (JVClose)	4 7
・ ボタンの停止・再開機能 (BtnClose, BtnOpen)	4 8
・ 開始・終了機能 (Kaishi, Syuryou)	4 9
・ ダウンロード機能+DB への登録機能：蓄積系データの場合 (JVOpen→JVGets→JVSkip→JVClose)	5 0
・ DB からデータを取得する機能 (SQL)	5 9
・ 実行ファイル (.exe) を作成 (ビルド) する方法	6 5
おわりに	6 6
参考文献	6 6
見ておくと良いサイト	6 7
装丁情報	6 7
奥付	6 8

## はじめに

競馬の過去データを使って統計・分析を行いたい方には是非 JRA-VAN というサービスを知っていただきたいです。月額 2090 円掛かりますが、競馬の過去データ（レース情報やオッズから動画など様々なデータ）を取得できますし、無料でついてくるソフトを使えば自分のやりたいデータ分析ができる可能性があります。そんなソフトがなくとも、自分でプログラミングを行えば自分好みの分析ソフトを作成することができるサービスです。そして、本同人誌を参考にすれば自分がやりたい分析に必要なデータの取得がスムーズに行くはずです。

本同人誌の内容を簡潔に述べると、JRA から競馬の過去データをダウンロードする方法を紹介するものです。詳しく述べるならば、JRA が提供している競馬の過去データを入手するために、JRA-VAN データラボというサービスで配布されている JV-Link モジュールをどう使えば過去データを入手できるのかの方法の紹介になります。

本同人誌では、プログラミングが初めてという人でも、見よう見まねでソースコードを改変し、目的の情報を入手できることを目指しております。そのため、前半部では今回のプログラミングで必要となる最低限のプログラミングの知識をざっくりと説明しております。登場した用語をキーワードとして検索してさらに知識を深めて頂ければ幸いです。後半では私が作成したソースコードを公開し、一行ずつどういう意味があるコードなのかの説明を行い、改変ポイントを示すことで改変しやすくしました。

質問掲示板を覗いていると、Visual Studio 2022 の C# で開発すると将来性があるとのことですが、では、2019 の vb.net の本同人誌に立場がないのかということ、別にそんなことはありません。とにかく手軽にデータを入手したいならばコピペでどうにかなる本同人誌は役に立ちますし、2022 や C# へ移行するにせよ手直しの手本となる完成したソースコードを提供しているという点で役に立つ余地はあります。

注意事項としては、本同人誌では蓄積系ソフトのソースコードを載せていますが、非蓄積系ソフトのソースコードは載せておりません。また、JRA-VAN においては動画などのデータの取得も可能ですが、本同人誌ではレース情報などのテキスト情報しか扱っておりません。

また、私が疑問に思ったところは調べて載せておきましたが、理解不足なままの点やそもそも私が触れていない部分などがあるかもしれません。そういった場合は JRA-VAN が用意している三種類ある質問掲示板に質問をしていただければ公式や親切な方からの回答を得ることができると思います。

なお、本同人誌で作成するソフトは Visual Studio2019 にて作成し、使用言語は Visual Studio、対象 OS は Windows となっております。

本同人誌を作成した理由としては、JRA が公開しているチュートリアルでは微妙に痒い所に手が届いていないと感じたためです。また、正直なところチュートリアル以外の部分も不親切な部分が多いです。そのため、ソフト開発にこれだけ苦労したのだから何かしらの形で残したいという気持ちが沸き上がったのも一つの理由です。

競馬の分析をするぞ！という時によく挙げられる手法は、netkeiba のサイトをスクレイピングしてデータを集めるというものです。では、なぜわざわざ金（月額 2090 円）を払って JRA-VAN を利用するのでしょうか。正直なところ私はどちらの手法でもいいと思っております。が、なぜ私は JRA-VAN を利用したのかの理由を挙げると、

- ・公式のツールがあるのだから公式を使ってみることにした。
- ・netkeiba のスクレイピングをするプログラムの作成が面倒くさく感じた。
- ・vb.net でソフト開発するのはスキル向上になると思った。

あたりになります。

振り返ってみて実際どうだったかというと、勉強にはなったけど普通に面倒くさかったという感想になります。あ、本同人誌を参考にすれば私が経験した面倒くさは大幅に減らせることは保証します。

## 8 月上旬に行われる JV-Data の仕様変更について (Ver.4.9.0)

2023 年 8 月上旬から JV-Data の仕様が変更されます。幾つかの変更点がありますが、大まかに分けると、

- ・JV-Data の幾つかのレコードのバイトの長さが変更された。
- ・新しいデータ種別が追加された。

の二つになります。

これによって、本同人誌の内容はどういった影響を受けるのでしょうか。

まず、レコードのバイトの長さは JVGets の size パラメータに関わってきます。size で設定した大きさよりも長いデータが渡された場合、はみ出したデータは捨てられるためです。しかしながら、それぞれのレコード長を確認したところ本同人誌で指定している size の大きさを超えるものはなかったため、こちらの変更で影響はないと考えられます。

次に、新しいデータ種別についてですが、変更日以降のデータは下に表示されている新しいデータ種別 ID (DIFN,BLDN など) しか利用できないそうです。一方で、旧データにも新しいデータ種別 ID は適応されるそうですので、基本的に新しいデータ種別 ID を使えば問題はなさそうです。(→JV-Link 質問箱 No.7070)

## 第一章 事前知識

今回のソフトを作成するにあって必要になるプログラミングの知識をざっくり紹介していきます。詳しく知りたい場合は紹介している単語を頼りに各自で調べるようお願いします。(プログラミング能力には検索能力も含まれる……ハズ)

### 1-1：プログラミングについて

プログラミングを学ぶという言葉があります。私は、この言葉には2段階あると考えています。「アルゴリズム」を学ぶ段階と「ライブラリ」を学ぶ段階です。

プログラミングに触れたことのない人にどういうことかと説明すると、こうなります。大雑把に言えばプログラミングとは、こういう順番でこういう動作(処理)をしてほしいというものを記述したものです。これを分解すると、「動作」とそれをどういう「順番」で行うかに分けられます。例えば、料理においては「切る」、「炒める」、「煮る」などの動作があり、それらをどういう順番で行うのか考える必要があります(野菜炒めでは「炒める」前に「切る」必要がありますよね)。ここにおける順番がアルゴリズムであり、「切る」などの動作一つ一つがライブラリという理解をお願いします。

アルゴリズムはどのプログラミング言語でも共通しています。これが、一つのプログラミング言語が扱えれば他のプログラミング言語も簡単に学べるという言説の根っこだと思います。

一方で、ライブラリは言語や開発環境によって変わってくるため、他のプログラミング言語の知識をそのまま使えるかは怪しいものがあります。そして、そもそもライブラリをすべて覚えることは不可能なので、実際に手を動かす過程で必要なライブラリを調べ、覚えていくことになります。プログラミングの勉強をするなら、何か作った方がいいよ！ という言説はこの辺りから来ると思います。

プログラミング教室には6年以上前に少し触れた(サイト形式でした)だけなので流石にもうだいぶ変化していると思いますが、その時にやったIf文100回ノックなどはアルゴリズムの勉強であり、ライブラリの勉強ではないので、大事ですがそんなに繰り返しやっても仕方ないかなと思う次第です。

### アルゴリズム

まず、実現したい動作を行うソフトを作るには、どういう順番でどんな動作を行えばいいのかを考えなくてはなりません。どの動作をどういう順番で行うのかを示すのがアルゴリズムであり、それを表現する方法として「フローチャート」や「アクティビティ図」があります。

プログラミングで使うアルゴリズムの動作(処理)は三つあり、この三つを覚えるだけで今回のソフトを組むことができる(し、基本的にどんなソフトも組めるはずです)。その三つの処理はそれぞれ、「順次構造」、「選択構造(分岐処理)」、「反復構造(繰り返し処理)」と言います。

- ・順次構造

アルゴリズムの大前提です。記述した順番に処理を実行していくことを指します。プログラミングで言えば上の行から下の行へ順番に処理していきますよということです。

- ・選択構造

条件によって処理の流れを切り替えることを指します。いくつかの種類があるので、以下のキーワードで検索すると良いでしょう。ちなみに Select Case 文は python で使えないなど言語によって微妙に違いがあるので「vb.net If」といった形で検索するとより確実です。

キーワード)

- ・If 文

- ・Select Case 文

- ・反復構造

条件が成立している間はある範囲の処理を繰り返すことを指します。条件の判定を最初にやるのか最後にやるのかなどで構文が変化するので、注意が必要です。選択構造と同じく言語によって違いがあります。

キーワード)

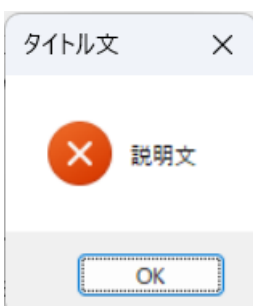
- ・Do Loop 文

## ライブラリ

先人達が作成してくれたもので、ある処理・機能を実行したい時に（だいたい）1行書くだけでその処理が実行できるもののことです。

その威力をぼんやりとでも理解してもらうために一つの例を挙げます。MessageBox.Show() メソッドです。このメソッド（→クラスと New 演算子）は以下のように書くだけで、メッセージボックスを表示することができるのです。

1: MessageBox.Show("説明文", "タイトル文", MessageBoxButtons.OK, MessageBoxIcon.Error)



写真：メッセージボックスの一例

これの何が凄いのかというと、たった一行でメッセージボックスの表示だけでなく、ユーザーが OK ボタンを押したらメッセージボックスが消える処理まで実行してくれるのです。つまり、

これぐらいの新しいウィンドウを作成して……、この辺にテキストを表示させて……、この辺にアイコンを表示させて……、ユーザーが OK ボタンを押したらウィンドウを閉じて……、などの細かいプログラムを書く必要がなく、たった一行で「メッセージボックスを表示させる」ことができる凄さが伝われば幸いです。

また、引数 (→関数とプロシージャ) である、「MessageBoxButtons.OK」や「MessageBoxIcon.Error」は提供側が指定したもの、例えば「MessageBoxButtons.OKCancel」(「OK」と「キャンセル」ボタンが表示される) や「MessageBoxIcon.Exclamation」(黄色い三角に感嘆符記号が表示される) に変更するだけでメッセージボックスを簡単に変更できる点も非常に便利です。

ライブラリを可能な限り利用することで、プログラミングに割く時間をより重要な場所にあてることができるようになります。

## 変数と配列

変数とは文字や数字を入れる入れ物のようなもののことです。実際は箱ではないのですが大雑把に箱という認識で今回は大丈夫です。そこら辺を詳しく知りたい場合は基本型と参照型というキーワードで調べて頂きたいです。

※以下のコードは vb.net の規則に準じたものではなく、抽象的なコードです。

A = 100

で A という変数に 100 という数字を入れられます。

変数を操作することで色々できますが、とりあえず以下のようなことができます。

A = 100   B = 50

C = A + B   C には 150 が入る

C = A - B   C には 50 が入る

C = A \* B   C には 5000 が入る

C = A / B   C には 2 が入る

また、

A = "こんにちは"   B = "世界"

C = A + B   C には"こんにちは世界"が入る

ちなみに、以下の処理はエラーになる。

A = "こんにちは"   B = 50

C = A + B   C はエラーになる

この理由はデータ型の項で説明します。

配列とは、複数の変数をまとめたものです。同じ型 (→データ型と宣言) の変数しかまとめることができません。

A = {"日本", "ドイツ", "アメリカ"}

B = A(0)   B には"日本"が入る



C = A(2) C には"アメリカ"が入る

上のように配列と共に番号を指定することで配列の中身を取り出すことができます。なお、配列においては 0 から番号を数えます。ですので、A(0)で日本が取り出せます。

## データ型と宣言

変数にはいくつかの種類があり、変数の種類と変数に入れるデータの種別を合わせないと変数に入れることができません。この種類のことを「型」といいます。

vb.net にはいろいろな型がありますが、基本的に以下のものを知っていればこの同人誌では問題ありません。

Integer：整数型（-2147483648 ～ 2147483647 の範囲の数字を扱うことができますが、少数点以下は扱えません）

Decimal：十進型（少数を扱うことができます。少数の桁数によって扱える範囲が変わります）

String：文字列型（文字を扱います）。" "で文字列を括ります。例："日本"

Date：日付型（日付、年月日時分秒まで扱います）

### 宣言

宣言とは変数のデータの種別と名前を明示することです。vb.net では以下のように書きます。

Dim 変数名 As データ型

Dim Name As String

Dim 配列名 As データ型

Dim NameList() As String

変数を使用する際は最初にこのように宣言を行う必要があります。宣言することでソフトは変数を扱うことができるようになります。

型が異なるもの同士を計算することはできず、エラーになってしまいます。そのような場合は、型変換を行うことで計算ができる場合があります。四則演算などでエラーが起こる場合はその変数の型を確認する癖をつけましょう。

Dim A As String = "348"

Dim B As Integer = Integer.Parse(A) B には 348 が入る

Dim C As Integer = 348

Dim D As String = C.ToString() D には"348"が入る

## 関数とプロシージャ

関数とは入力と処理、出力を持つ完結したプログラムのことです。数学の関数  $y = f(x)$  と同じようにある入力を渡すと決まった処理を行い、出力を返します。

例えば、自動販売機にお金を入れ（入力）、飲み物を選択すれば、飲み物が出てきます（出力）。

抽象的なプログラムで表すと以下のようになります。

```
Tashizan(x, y)
```

```
    z = x + Y
```

```
    Return z
```

```
Goukei = Tashizan(3, 6)
```

実行すると Goukei には 9 が入ります。ここでは、3 と 6 という入力に対し、二つの数字を足し合わせるという処理を行い、合計の 9 を出力しています。そして、9 を変数 Goukei が受け取っています。

なお、入力されたものを「引数」と言い、出力されたものを「戻り値」と言います。このように入力（引数）、処理の他に出力（戻り値）がある関数のことを「関数（ファンクション）」と呼びます。

また、出力、もとい戻り値がない関数も存在します。そういった関数を「プロシージャ」と言います。このような引数も戻り値もない関数を使うことで、コードの重複を減らすことができます。ようするに、複数回登場する複数の処理を一つの処理にまとめるのです。

例えば、毎日のタスクを書き出す時、食事関連のタスクは 3 回書くことになります。その際に、（食事を用意する→食べる→片付け→歯磨き）を 3 回書くより、それらの処理をまとめた（食事）という処理を作ってこれを 3 回書いた方が楽です（食事という処理の内容はどこかに書く必要はありますが 1 回で済みます）。

重複を減らすと何がいいのでしょうか。

第一に、バグが混入する可能性が下がります。重複したコードをまとめることでコードの長さが短くなるためそもそもバグが起こる可能性が減ります。また、短いコードは理解しやすいためバグが減ります。

第二に、コードの変更を行う際に複数箇所を変更する必要がなくなります。同じ処理を何度も書いている場合、その処理で変更があった場合にはそれらすべてを書き直すことになります。変更する数が多ければ変更のミスが発生する可能性は上がりますし、そもそも変更し忘れるものもあるかもしれません。

## クラスと New 演算子

クラスとは設計図のようなものです。オブジェクト指向というプログラミングにおいて出てくる概念の一つに含まれます。

オブジェクト指向については、今回よくわかってなくても問題ありません。

クラスには属性（プロパティ）と操作（メソッド）がくっついています。

属性はそのクラスがどういうものなのかを表す情報のことです。例えば、エクスプローラーで、文章ファイル（.txt など）を右クリックしてプロパティを表示すると、そのファイルのサイズや作成日時が分かります。これらの情報は変数で保持されています。

操作はそのクラスがどんな処理をできるかということです。動きは関数と同じなので関数のようなものなのだという理解で問題ありません。

クラスは、属性＝情報（変数）と操作＝処理（関数）を一つのテーマでまとめることによって扱いやすくしたものです。

プログラミングにおいては、クラスをそのまま使うことはできません。クラスという設計図からインスタンスという実物を作成することで使うことができます。そして、作成した個別のインスタンスを使ってプロパティを変更したり、メソッドを実行したりします。

クラスとインスタンスの関係を理解しやすくするために一つ例を出します。

例えば、RPG ゲームにおいて、戦士や魔法使いには HP や MP といった「情報」と通常攻撃やスキル攻撃などの「処理」があるとします。それぞれのキャラに個別にそれらを設定してもいいでしょうが、関数の項で説明したように重複は減らしたほうがいいです。そこで、ヒューマンというクラスを作成します。ヒューマンクラスには HP や MP というプロパティと通常攻撃とスキル攻撃というメソッドが含まれています。このクラスから戦士 A というインスタンスや魔法使い B というインスタンスを作成します。インスタンスはクラスと同様のプロパティとメソッドを持っていますが、クラスとは同期していません。そして、他のインスタンスとも同期していません。そのため、戦士 A が攻撃を受けて HP が減った場合、戦士 A のインスタンスのプロパティの HP だけを減少させることができます。

インスタンスを作成する際に必要なのが New 演算子です。クラスから「新しい」インスタンスを作成すると覚えましょう。

以下は Human クラスから Mahoutukai インスタンスを作成しています。

```
Dim Mahoutukai As New Human
```

や

```
Dim Mahoutukai As Human 'Human 型の変数を宣言し
```

```
Mahoutukai = New Human 'インスタンスを作成
```

のように書きます。

また、プロパティやメソッドを実行するためには以下のようにします。これを、「メソッドの呼び出し」といいます。

```
Mahoutukai.HP = 100 'HP プロパティの値を 100 にしています。
```

```
Mahoutukai.skill() 'スキルメソッドを実行しています
```

なお、クラスやインスタンスのことをオブジェクトと呼びます。

## スコープ

ソースコード内で変数や関数にアクセスできる範囲のことを指します。

変数は宣言した場所によって使用できるソースコードの範囲が変わってきます。変数のスコープには以下の 3 種類があります。

- ・ブロックスコープ
- ・プロシージャスコープ
- ・クラスレベル（モジュール）のスコープ

が、ブロックスコープとプロシージャスコープはほとんど同じなので「ローカル変数」という名称でまとめて呼ぶことにします。

ソースコードの例文を使って変数のスコープの概念を説明します。

```
Public Class Form1
```

```
    Private x As Integer = 2 'クラスレベルのスコープ
```

```
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
```

```
        Dim y As Integer = 2 'プロシージャスコープ
```

```
        If x = y Then
```

```
            Dim z As Integer = 1 'ブロックスコープ
```

```
            x = y + z
```

```
        End If
```

```
        x = y + z 'z が有効範囲外なのでエラーが発生する
```

```
    End Sub
```

```
    x = y + z 'y と z が有効範囲外なのでエラーが発生する
```

```
End Class
```

ブロックの中で宣言された変数はそのブロックの中でしか利用できないということです。

ローカルの変数になく、クラスレベルのスコープを持つ変数にあるものとして、アクセス指定子があります。アクセス指定子とは、他のクラスからその変数にアクセスできるのかを定義するものです。クラスレベルのスコープを持つ変数にはアクセス指定子をつけるようにしましょう。

よく使うアクセス指定子としては、Public、Friend、Private などが挙げられます。それぞれ、以下の場所からアクセスすることができます。

- ・ Public：別のプロジェクトや同じプロジェクトなどどこからでもアクセスできます。
- ・ Friend：同じプロジェクトからのみアクセスできます。
- ・ Private：同じクラスからのみアクセスできます。つまり、他のクラスからはアクセスすることが出来ません。

バグを減らす工夫として、極力狭いスコープやアクセス指定子を使うことが挙げられます。

## 命名規則

バグを減らすためにソースコードの可読性や視認性を上げる方法の一つとして命名規則があり、特に複数人で開発する際に使われます。これは変数や関数につける名前をどのようなルールに沿って名付けるかというものです。例えば変数においては、String 型の変数だということを伝えるために strName という変数名にする場合があります。この命名規則は定まったものではなく、プロジェクトによってルールが変わります。一人で作業する場合は自分が分かりやすいルールを決めて運用しましょう。

本同人誌の命名規則としては、変数においては「型名+大文字から始まる変数名 (strDataSpec、IOption)」を、関数においては「大文字から始まる関数名 (CloseJV)」を採用しているのですが、ところどころ違う箇所があります（よろしくない）。

なお、ソースコードの書き方のルールをコーディングルールといいます。こちらも可読性や視認性に関わってきます。

## イベント

イベントとは処理を行うきっかけとなる出来事のことです。例えば、ボタンを押したら A というプログラムが動いたり、マウスをクリックしたら B というプログラムが動くといった際に「ボタンを押す」、「マウスをクリックする」などの何らかの外部の刺激のことをイベントと言います。

## Visual Studio、vb.net について

Visual Studio とは、これ一つでソフトウェアの開発が可能なソフト（統合開発環境 (IDE)）のことです。コードの編集から、デバッグ、ビルド（ソフト生成）までこのソフト一つで行えます。Windows、macOS、Android、iOS など、それぞれの OS で使用できるソフトの開発が可能なのも特徴です。

Visual Studio では基本的に vb.net か C#で開発を行います。今回は vb.net を利用します。

vb.net とは、Visual Basic .NET というプログラミング言語の略称のことです。とはいえ、最新のバージョンは「Visual Basic 2019」と Visual Basic と名乗っています（一時期は Visual Basic .NET と名乗っていました）。ですが、Visual Basic と言っても中身は Visual Basic .NET です。ちなみに昔のバージョンである Visual Basic は既にサポートが終了していますので、Visual Basic .NET を利用するようにしましょう。

## 1-2：ソフト開発について

### JRA-VAN とは

JRA-VAN とは以下のようなサービスです。

「JRA-VAN」は、JRA の 100%関連会社である JRA システムサービス株式会社が提供する JRA 公式データを使った競馬情報サービスのブランド名称です。過去 30 年分のレースデータから、オッズ、提供しています。JRA-VAN を使えば、パソコン向け競馬ソフトやスマートフォンで、競馬をより深く・便利に楽しむことができます。

引用：<https://jra-van.jp/sup/first.html> 2023.07.15

JRA-VAN には 4 つのサービスがありますが、今回はその中の JRA-VAN データラボ (JRA-VAN Data Lab.) を利用することになります。利用料金は、月額 2,090 円(税込)です。

毎月 2,090 円の出費が必要かということ、実はそんなことはありません。毎週、リアルタイムで情報を入手して分析し、レースに賭けることをしなければ一月だけの契約で問題ありません。ということかということ、2,090 円を一回払うだけでその日までの過去データを入手することができるため、過去のデータを入手してこの手法が正しいのか確認したい！ などの場合は、何ヵ月も支払いをする必要はないのです。詳しく説明すると、本同人誌で紹介しているソースコードは JRA-VAN からデータをダウンロード、解凍して DB に登録、目的のデータを検索・抽出して CSV 形式で出力するものです。データのダウンロード・解凍には JRA-VAN の機能が必要ですが、DB を操作するには JRA-VAN の機能は必要ありません。ですので、一回データを DB に登録して保存しておけば、手法の検証は DB を弄るだけで達成できるため、何ヵ月も JRA-VAN にお金を払う必要はありません。最新のデータに更新したいという場合にはまた契約して最新のデータを入手すればよいのです。

クレジットカードで支払いを行う場合、契約時に自動的に来月も料金を支払う設定になっているので、しっかりと設定を確認して不本意な契約の更新がないようにしましょう。なお、いつ契約しても有効期間は 1 ヶ月ある（月初めから数えるわけではない）ので気楽に契約しましょう。

### 蓄積系ソフトか非蓄積系ソフトか

JRA-VAN のデータを利用する競馬ソフトは大きく二つに分けることができます。「蓄積系ソフト」と「非蓄積系ソフト」です。二つの違いはデータの保持方法です。

蓄積系ソフトは JRA-VAN の過去データをデータベースとして利用するものです。「3 年前のレースの分析をしたい」や、「この傾向が過去にも当てはまるのかを検証したい」などといった場合はこちらになります。

一方、非蓄積系ソフトは今週開催されるレースに関する情報のみを扱うことになります。「今度のレースでこの条件に当てはまる馬を確認したい」というように、今週に開催されるレースの情報だけにしか興味がない場合は、データ容量の節約になるため、非蓄積系ソフトを利用・作成するといいいでしょう。

なお、本同人誌では蓄積系ソフトの説明を行い、非蓄積系ソフトの説明は行いません。

## 蓄積系データか速報系（リアルタイム系）データか

ソフトはデータを蓄積するかないかで二つに分けることができました。一方で JRA-VAN で入手できるデータも二つの種類に分けることができます。蓄積系データか、速報系（リアルタイム系）データかです。詳しくは「JV-Data 仕様書」のデータ提供タイミング・提供単位を確認してください。

ここでざっくり説明すると、蓄積系データは JVOpen メソッド（→蓄積系データの場合）で取得し、昔のデータもセットアップを行うことで入手できます。データ提供タイミング・提供単位では提供期間が1年間と書かれていますが、これは通常データ（→次の項）として提供される期間のことであり、1年以上前のデータはセットアップデータとして入手できるという意味のようです。（JV-Data の使い方あれこれ No.2711, No.2712 を参考）

一方で速報系データは、JVRTOpen メソッドで取得します。昔のデータは取得することができず、字句通りの意味でデータ提供期間は基本的に1週間となっています。時系列オッズの単複枠と馬連のみ1年間のデータ提供期間となっています。本同人誌では速報系データを扱いません。

## 蓄積系ソフトのデータ取得

蓄積ソフトにおいて過去データの取得方法について説明します。

蓄積系ソフトは最初に、過去から現在までの情報を漏れなく取り込みます。これを「セットアップ」と呼びます。その後、日々発生する更新データ（差分データ）を漏れなく取得することにより、DB の完全性を保ちます。この差分データの取得を「通常データ取得」と呼びます。なお、セットアップは対象ファイルが多くなる場合が多く、処理に時間が掛かります。

セットアップや通常データ取得で取り込んだ最後のファイルを記録することで、漏れのない通常データ取得が可能になります。前回取得したデータの最終時刻を「FROM タイム」と呼び、競馬ソフトにこの FROM タイムを次の取り込みまで保存しておく必要があります。Visual Studio には作成したソフトに情報を記録させる仕組みがあるので、その機能を使っていくことになります。

その他の情報や詳しい情報は、JRA-VAN Data Lab.開発ガイド（Ver.4.2.2）の記述を確認してください。



## 1-3：今回の開発で使う機能

### コントロール

Visual Studio においてソフトの画面に貼り付けて使う部品のことを「コントロール」と呼びます。ボタン（押すとイベントが発生する）、ラベル（テキストを表示する）、テキストボックス（テキストを入力するためのもの）、チェックボックス、コンボボックスなどが存在します。コントロールを使用するためには、左側面にあるツールボックスから目的のコントロールを選択し、フォーム上にドラッグ&ドロップするだけです。

もし、ツールボックスが表示されていない場合は、[表示]メニュー>[ツールボックス]をクリックすると表示されます。ツールボックスを常に表示したい場合はツールボックス右上にある画鋲アイコンをクリックして縦方向にしましょう。

よく使用されるコントロールはコモンコントロールのタブに登録されています。このタブのコントロールだけで今回のソフト開発は事足ります。

コントロールは端を選択しながら動かすことで大きさや形を変更することができます。

### プロパティ

コントロールのプロパティを変更することでコントロールの見かけを変えたり、プログラムの動作を変更することができます。プロパティの変更はプロパティウィンドウで簡単に変更できます。

プロパティウィンドウは画面の右下にあります。表示されていない場合は[表示]メニュー>[プロパティウィンドウ]で表示できます。ウィンドウを最初の状態に戻したい場合は、[ウィンドウ]メニュー>[ウィンドウレイアウトのリセット]をクリックしてください。

プロパティを変更する際には、変更したいコントロールをクリックして選択状態にする必要があります。

特に、Design の(Name)と表示の Text の変更はよく行います。(Name)はそのコントロールの名前であり、Text はボタンにおいてソフト上で表示されるボタン上の文章のことです。

### ボタン

ボタンを押すことでイベントが発生します。ボタンが押されたらこの処理を行うといった実装を行うことになります。

コモンコントロールからボタンを選択し、フォームにドラッグ&ドロップすることでボタンが作成されます。ボタンの周りにある白い四角などをクリックしたまま動かすことでボタンのサイズを変更できます。また、ボタンの何処かをクリックした（掴んだ）まま動かすことで、ボタンの位置を変更できます。

配置したボタンを削除する場合は、フォーム上のボタンを右クリックして「削除」を選択すれば完了です。

ボタンをクリックした時に起こる動作をプログラミングする方法について説明します。フォーム上のボタンをダブルクリックすることで、そのボタンに紐づけられたイベントがコード上に記入されます。記入された Private Sub～End Sub の間に処理を書くことでイベントの処理を書くことができます。

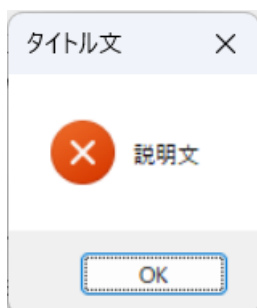
## メッセージボックス

なんらかの出来事（大抵はエラーです）が発生した際に、ユーザーにアナウンスするためにメッセージボックスは使用されます。

構文としては以下の通りです。

MessageBox.Show("文章","タイトル",メッセージボックスのボタンの種類,メッセージボックスのアイコン)

例：MessageBox.Show("説明文", "タイトル文", MessageBoxButtons.OK, MessageBoxIcon.Error)



写真：メッセージボックスの一例

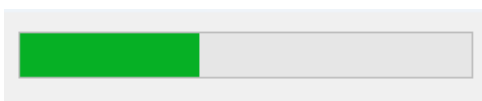
メッセージボックスのボタンの種類とメッセージボックスのアイコンについては、以下のサイトを見るのが分かりやすいです。

参考：<https://dobon.net/vb/dotnet/form/msgbox.html>

とはいえ、本同人誌では、メッセージボックスのボタンの種類は、MessageBoxButtons.OK（「OK」ボタンのみ）と MessageBoxButtons.OKCancel（「OK」と「キャンセル」ボタン）しか使わず、メッセージボックスのアイコンは MessageBoxIcon.Error（赤丸に白い X（停止マーク））と、MessageBoxIcon.None（アイコンなし）、MessageBoxIcon.Exclamation（黄色い三角に感嘆符記号）しか使いません。

## プログレスバー

プログレスバーとは、進捗状況を棒状の表示領域が変化していくことで表すもののことです。このことです。



長い処理を待っている際、あとどれだけ待てばいいのかが分からないのは意外に大きなストレスです。どれだけ処理をやったのかを知らせることで、そのストレスを軽減することができるのでできるだけ実装するようにしましょう。

ソースコードにおいての使い方としては、プログレスバーの最大値 (Maximum) を設定した後、プログレスバーの値 (Value) を徐々に増やしていくことで緑のバーが増えていきます。最大値には処理すべき項目の数を入力し、処理が一つ終わったら値を+1 することで役割を果たすことができます。

## デバッグ

デバッグとは不具合 (バグ) の原因を探して直すことを言います。今回のようなソースコードをコピペし、適宜改変するといった場合に多いであろうバグはソースコードの打ち間違いですが、そのほかにも様々な原因によってバグが引き起こされます。Visual Studio にはバグの原因が何によって引き起こされているのかを見つけるためのツールを提供しています。

まず、ソースコードにおいて赤い波線がコードの下に引かれていた場合、その個所はソフトを実行した場合になんらかのエラーが発生することを知らせています。

Visual Studio の中央上付近にある緑色に塗りつぶされた▶を押すことによってデバッグモードでソフトを実行します。ソースコードをキリのいいところまで書いたら、そのソースコードで思惑通りにソフトが動くのかを確認する癖をつけると、どこにソースコードの間違ひがあるのかを探しやすくなります。

今回のようなコピペと改変を使う場合、一番多いであろうエラーはソースコードの打ち間違いです。これは、あからさまな間違いなら赤い波線で教えてくれます。

デバッグはどこにエラーがあるのかを探す作業が第一です。そのために、ブレークポイントの設定や、出力ログを使ってどこでエラーが起きているのかを探すといいでしょう。

また、大前提ですが、エラーログは読みましょう。何を書いてあるのかわからなくてもとにかくエラーログは確認しましょう。エラーログの最初の行をコピペして検索すれば、大体の場合、そのエラーを解決する方法が書かれた公式のテキストやブログ、記事などがヒットします。エラーログの意味が分からなくてもそういった外部の情報で解決できることが多いです。あきらめずに頑張りましょう。

## コメントアウト

ソースコードを後から見返した時に理解しやすいよう、補足事項を書き込みたい場合があります。そういった目的で、ソースコードの処理には影響せずにコメントを書き込む方法がありま

す。vb.net では「'」をコメントの前につけることでそれ以後の文章は処理になんの影響も及ぼさなくなります。これをコメントアウトといいます。

例

```
Kokugo = 30 '国語の点数
```

```
Suugaku = 40 '数学の点数
```

```
Goukei = Kougo + Suugaku 'テストの合計点
```

なお、注意点として、' の位置には気を付けましょう。

例

```
'Kokugo = 30 '国語の点数 ※この処理は実行されない
```

```
Suugaku = 40 '数学の点数
```

```
Goukei = Kougo + Suugaku 'テストの合計点 ※エラーを吐く
```

変数 Kokugo の前に ' があるため Koukugo に数字が代入されていません。そのため、Goukei が計算される行でエラーが発生します。

コメントアウトされた文章は Visual Studio が勝手に緑色に染めるので、間違えてコメントアウトした場合でもその個所は見つけやすいかと思います。

## 第二章 ソフト開発

### 2-1：全体像の説明

#### Visual Studio のインストール

情報が更新されることが多いので、各自でインストール方法は調べてください。

インストールするのは Visual Studio 2019 の Community エディションです。このエディションは無料で Visual Studio を利用することができます。本同人誌では Visual Studio2019 を利用しています。2022 版で開発する場合は幾つかの変更点がありますが、本同人誌では紹介しません。

.NET デスクトップ開発を選択してインストールしましょう。

Visual Studio をインストールすれば vb.net を使用できるので、別途インストールする必要はありません。

Visual Studio は紫色のアイコンです。似ている visual studio code は青色のアイコンですので、インストールの違いに気を付けましょう。

#### ダウンロードしておくもの

・DataLab. (データラボ) 会員サービス ソフト開発サポート ソフトウェア開発キット (SDK) 提供コーナー

<https://jra-van.jp/dlb/sdv/sdk.html>

から最新版の「JRA-VAN SDK 本体 (Ver.4.8.0.2) (2023/2/22)」(2023.07.11 現在) をダウンロード

・プログラミングパーツ・開発支援ツール提供コーナー

<https://jra-van.jp/dlb/sdv/pgm.html>

より、

・データベース作成クラス

Microsoft Visual Basic 2019 (2021/5/26)

・JV-Data 登録クラス

Microsoft Visual Basic 2019 (2021/12/28)

をダウンロード

## 簡単な流れの説明

ソースコードの簡単な流れを説明します。

まず、JV-Link をソフトに導入します。こうすることで、JV-Link のメソッドを利用することができるようになります。その後、JV-Link の設定画面を開く機能を実装します。また、JV-Link の初期化機能も実装します。

次に JV-Link で取得したデータを登録する DB に関連する機能を実装していきます。それぞれの機能は DB の作成、削除、最適化です。その後、その DB に接続・切断する機能を実装します。

次に、JV-Link を終了するための機能を実装します。そして、JV-Link からデータを取得（ダウンロード）し、DB にデータを登録する機能を実装します。

最後に、DB に登録した大量のデータの中から SQL でデータをどう抽出するのかの説明を行い、抽出したデータを CSV 形式のファイルに出力する機能を実装します。

なお、JV-Link において契約が必要なメソッドは JRA の DB とやり取りする JVOpen・JVRTOpen（そして、JVGets）のみです。ですので、「ダウンロード機能+DB への登録機能」より前に紹介しているソースコードは契約をせずともデバッグが可能です。ですので、プログラミングが沼る可能性も踏まえ、契約は「ダウンロード機能+DB への登録機能」のデバッグを行わなければいけない段階まで控えておくといいと思います。

## 全体のソースコード

Form1.vb

Imports System.IO

Imports System.Text

Public Class Form1

Dim clsJVLink As New JVDTLabLib.JVLinkClass

Private objDBControl As clsDBBuilder

'DB のパス

Dim appPath As String = AppDomain.CurrentDomain.BaseDirectory.TrimEnd("\c") '

動いているソフトのパスを取得する。

Private dbPath As String = appPath.Replace("\Debug", "") 'パスから\Debug を""無に書き  
かえ。Debug だから Debug だけど完成したら Release になることに注意

Private txtPath As String = dbPath + "\Data.accdb"

Private Sub Form1\_Load(sender As Object, e As EventArgs) Handles MyBase.Load

Dim ReturnCodeInit As Long = clsJVLink.JVInit("UNKNOWN")

If ReturnCodeInit = -101 Then

MessageBox.Show("JVInit エラー：引数が設定されていない", "エラー",  
MessageBoxButtons.OK, MessageBoxIcon.Error)

ElseIf ReturnCodeInit = -102 Then

MessageBox.Show("JVInit エラー：引数が 64byte を超えている", "エラー",  
MessageBoxButtons.OK, MessageBoxIcon.Error)

ElseIf ReturnCodeInit = -103 Then

MessageBox.Show("JVInit エラー：引数の 1 桁目がスペースになっている", "エラ  
ー", MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

End Sub

Private Sub btnJVSetUI\_Click(sender As Object, e As EventArgs) Handles  
btnJVSetUI.Click

Dim ReturnCodeSetUIPro As Long = clsJVLink.JVSetUIProperties()

If ReturnCodeSetUIPro <> 0 Then

MessageBox.Show("JVSetUIProperties エラー", "エラー",  
MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

End Sub

Private Sub btnDBCreate\_Click(sender As Object, e As EventArgs) Handles  
btnDBCreate.Click

objDBControl = New clsDBBuilder

If objDBControl.CreateDB(txtPath) = True Then

    MessageBox.Show(txtPath & "に DB を作成しました", "",  
    MessageBoxButtons.OK, MessageBoxIcon.None)

Else

    MessageBox.Show("エラー：DB 作成に失敗しました", "エラー",  
    MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

objDBControl = Nothing

Debug.WriteLine(appPath)

End Sub

Private Sub btnDBDelete\_Click(sender As Object, e As EventArgs) Handles  
btnDBDelete.Click

    Dim result As DialogResult = MessageBox.Show("DB を削除していいですか？", "確  
    認", MessageBoxButtons.OKCancel, MessageBoxIcon.Exclamation)

    If result = DialogResult.OK Then

        objDBControl = New clsDBBuilder()

        If objDBControl.KillDB(txtPath) = True Then

            MessageBox.Show(txtPath & "を削除しました", "", MessageBoxButtons.OK,  
            MessageBoxIcon.None)

        Else

            MessageBox.Show("エラー：削除に失敗しました", "エラー",  
            MessageBoxButtons.OK, MessageBoxIcon.Error)

        End If

        objDBControl = Nothing

    ElseIf result = DialogResult.Cancel Then

        MessageBox.Show("削除をキャンセルしました", "", MessageBoxButtons.OK,  
        MessageBoxIcon.None)

    End If

End Sub



```

Private Sub btnDBOpt_Click(sender As Object, e As EventArgs) Handles
btnDBOpt.Click
    objDBControl = New clsDBBuilder()
    If objDBControl.CompactDB(txtPath) = True Then
        MessageBox.Show(txtPath & "の最適化に成功しました", "",
        MessageBoxButtons.OK, MessageBoxIcon.None)
    Else
        MessageBox.Show("エラー：最適化に失敗しました", "エラー",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
    End If
    objDBControl = Nothing
End Sub

Friend Sub ConnectDBF()
    'DB 接続文字列の取得
    strConnectionString = "Provider = Microsoft.ACE.OLEDB.16.0;" & "Data Source = " &
txtPath
    'データベースとの接続を行う
    If ConnectDB() = True Then
        ImportRA = New clsImportRA
        ImportHR = New clsImportHR
    Else
        'ADODB オブジェクトの開放
        gCon = Nothing
        MessageBox.Show("データベースと接続できませんでした", "エラー",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
    End If
End Sub

Friend Sub CloseDBF()
    'データベース登録クラスのクローズ
    If ImportRA Is Nothing = False Then
        ImportRA.Close()
        ImportRA = Nothing
    End If
    If ImportHR Is Nothing = False Then
        ImportHR.Close()

```

ImportHR = Nothing

End If

'データベースとの切断を行う。

If gCon Is Nothing = False Then

gCon.Close()

gCon = Nothing

End If

End Sub

Friend Sub JVClose()

Dim ReturnCodeClose As Long = clsJVLink.JVClose()

If ReturnCodeClose <> 0 Then

MessageBox.Show("JVClose エラー：" & ReturnCodeClose, "エラー",  
MessageBoxButtons.OK, MessageBoxIcon.Error)

End If

End Sub

Friend Sub BtnClose()

Me.btnJVSetUI.Enabled = False

Me.btnDBCCreate.Enabled = False

Me.btnDBDelete.Enabled = False

Me.btnDBOpt.Enabled = False

Me.btnJVDownload.Enabled = False

Me.btnCSVEx.Enabled = False

End Sub

Friend Sub BtnOpen()

Me.btnJVSetUI.Enabled = True

Me.btnDBCCreate.Enabled = True

Me.btnDBDelete.Enabled = True

Me.btnDBOpt.Enabled = True

Me.btnJVDownload.Enabled = True

Me.btnCSVEx.Enabled = True

End Sub

Friend Sub Kaishi()

```

    prgJVGets.Value = 0
    ConnectDBF()
    BtnClose()
End Sub

```

```

Friend Sub Syuryou()
    CloseDBF()
    BtnOpen()
    MessageBox.Show("処理が終了しました", "", MessageBoxButtons.OK,
MessageBoxIcon.None)
End Sub

```

```

Private Sub btnJVDownload_Click(sender As Object, e As EventArgs) Handles
btnJVDownload.Click

```

```

    Dim strDataSpec As String
    Dim strFromTime As String
    Dim IOption As Long
    Dim ReadCount As Long
    Dim DownloadCount As Long
    Dim strLastFileTimestanmp As String = ""

```

```

'引数設定

```

```

strDataSpec = "RACE"
strFromTime = "20160101000000-20170101000000"
IOption = "3"

```

```

Kaishi()

```

```

'ダウンロード処理

```

```

    Dim ReturnCodeOpen As Long = clsJVLink.JVOpen(strDataSpec, strFromTime,
IOption, ReadCount, DownloadCount, strLastFileTimestanmp)

```

```

'エラー判定

```

```

    If ReturnCodeOpen = -1 Then
        MessageBox.Show("該当データ無し" & vbCrLf & "OR" & vbCrLf & "データベ
ースは最新の状態です", "エラー", MessageBoxButtons.OK, MessageBoxIcon.Error)
        JVClose()
    End If

```

```

        Syuryou()
        Exit Sub
    ElseIf ReturnCodeOpen = 0 Then
        MessageBox.Show("正常", "", MessageBoxButtons.OK, MessageBoxIcon.None)
    Else
        MessageBox.Show("JVOpen エラー：" & ReturnCodeOpen, "エラー",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
        JVClose()
        Syuryou()
        Exit Sub
    End If

    'lastfiletimestamp の保存
    My.Settings.LastTime = strLastFileTimeStanmp
    My.Settings.Save()

    'JVGets 用
    Dim szBuff(0) As Byte
    Dim strBuff As String = ""
    Dim lbuffSize As Long = 110000
    Dim strFileName As String = ""
    Dim strRecID As String

    prgJVGets.Maximum = ReadCount
    MessageBox.Show("JVGets を開始します", "", MessageBoxButtons.OK,
    MessageBoxIcon.None)

    Do
        Dim ReturnCodeGets As Long = clsJVLink.JVGets(szBuff, lbuffSize,
        strFileName)
        'エラー判定
        Select Case ReturnCodeGets
            Case Is > 0
                '正常
                ' 文字コード変換
                strBuff = System.Text.Encoding.GetEncoding(932).GetString(szBuff)
                strRecID = strBuff.Substring(0, 2)

```

```

'処理対象データのみデータベースへ登録
If strRecID = "RA" Then
    ImportRA.Add(strBuff, lbuffSize)
ElseIf strRecID = "HR" Then
    ImportHR.Add(strBuff, lbuffSize)
Else
    clsJVLink.JVSkip()
End If
ReDim szBuff(0)
Case -1
    'ファイルの区切れ
    prgJVGets.Value = prgJVGets.Value + 1
Case 0
    '全レコード読み込み終了
    prgJVGets.Value = prgJVGets.Maximum
    Exit Do
Case -3
    'ダウンロード中
    System.Threading.Thread.Sleep(1000)
Case Is <= -2
    'エラー
    MessageBox.Show("JVGets エラー コード：" & ReturnCodeGets +
vbCrLf + "動作を終了します", "", MessageBoxButtons.OK, MessageBoxIcon.None)
    JVClose()
    Syuryou()
    Exit Sub
End Select
Loop

    MessageBox.Show("JVGets 終了", "", MessageBoxButtons.OK,
MessageBoxIcon.None)
    JVClose()
    Syuryou()
End Sub

Private Sub btnCSVEx_Click(sender As Object, e As EventArgs) Handles
btnCSVEx.Click

```

```
Dim csvFileName As String = dbPath + "\KeibaCSV.csv"
```

```
Kaishi()
```

```
Dim dbRS As ADODB.Recordset
```

```
Dim dbFld As ADODB.Fields
```

```
Dim SQL As String
```

```
SQL = "SELECT Year, MonthDay, JyoCD, RaceNum, Honsyokin1, TenkoCD,  
DataKubun FROM RACE WHERE JyoCD='05' AND DataKubun ='7' ORDER BY Year desc,  
MonthDay desc, RaceNum desc"
```

```
dbRS = New ADODB.Recordset()
```

```
dbRS.CursorLocation = ADODB.CursorLocationEnum.adUseClient
```

```
dbRS.Open(SQL, gCon, ADODB.CursorTypeEnum.adOpenForwardOnly,  
ADODB.LockTypeEnum.adLockPessimistic)
```

```
prgJVGets.Maximum = dbRS.RecordCount
```

```
'CSV ファイル
```

```
System.IO.File.Delete(csvFileName) 'CSV ファイル削除
```

```
Dim csvSR As New StreamWriter(csvFileName, True, Encoding.UTF8)
```

```
Dim csvnakami As String = "開催年', '開催月日', '競馬場コード', 'レース番号', '本賞金',  
'天候コード'"
```

```
csvSR.WriteLine(csvnakami)
```

```
While Not dbRS.EOF
```

```
    dbFld = dbRS.Fields
```

```
    csvnakami = "" + dbFld("Year").Value() + "," + dbFld("MonthDay").Value() +  
    "," + dbFld("JyoCD").Value() + "," + dbFld("RaceNum").Value() + "," +  
    dbFld("Honsyokin1").Value() + "," + dbFld("TenkoCD").Value() + ""
```

```
    csvSR.WriteLine(csvnakami)
```

```
    dbRS.MoveNext()
```

```
    prgJVGets.Value = prgJVGets.Value + 1
```

```
End While
```

```
prgJVGets.Value = prgJVGets.Maximum
```

```
dbRS.Close()
```

```
dbRS = Nothing
```

```
csvSR.Close()
```

```
Syuryou()
```

```
End Sub
```

End Class

basUtility.vb

Public ImportRA As clsImportRA

Public ImportHR As clsImportHR

## プロジェクトの作成

Visual Studio においてプロジェクトを作成する方法を説明します。変化している可能性もあるため、以下の説明でわからなければ「Visual Studio 2019 プロジェクト作成」で検索してください。

Visual Studio を起動し、右側にある「開始する」の下にある「新しいプロジェクトの作成 (N)」をクリックします。

次のページでプロジェクトテンプレートを選択します。数が多くて探すのが大変なので、テンプレートの検索 (S) の下にある三つのプルダウンをそれぞれ「Visual Basic」、「Windows」、「デスクトップ」に設定することで、「Windows フォーム アプリケーション (.NET Framework)」テンプレートが表示されるはずです。そして、「Windows フォーム アプリケーション (.NET Framework)」をクリックします。

次のページではプロジェクト名と、場所、ソリューション名を設定します。これら三つは後から変更するのが面倒なのでそれを踏まえて設定しましょう。なお、ソリューションとはプロジェクトが入るフォルダの名前だという認識で問題ありません。今回はプロジェクト名とソリューション名が同じでも問題ありません。なお、日本語は使用しないようにしましょう。今回はプロジェクト名とソリューション名は両方とも「KeibaSoft」と名付けます。フレームワークは.NET Framework 4.8 を選択しました。

これでプロジェクトの作成は完了です。

## 適宜確認するもの

- ・「JRA-VAN SDK 本体 (Ver.4.8.0.2) (2023/2/22)」>「ドキュメント」>「JV-Link インターフェース仕様書\_4.7.0.1(Win).pdf」：JV-Link のメソッドの解説を確認できます。

- ・「JRA-VAN SDK 本体 (Ver.4.8.0.2) (2023/2/22)」>「ドキュメント」>「JV-Data 仕様書\_4.7.0.1.xlsx」：JV-Link で取得できるデータの詳細。データ種別 ID やレコード種別 ID、コード表、各レコードの中身の詳細などを確認できます。pdf よりも xlsx の方が見やすい気がします。

- ・「データベース作成 VisualBasic2019」>「データベース仕様書.xls」：CreateDB で作成される DB の構造が確認できる。

DataLab. (データラボ) 会員サービス ソフト開発サポート 競馬ソフト開発体験教室  
<https://jra-van.jp/dlb/sdv/trial.html>：JRA-VAN が提供している JV-Link のチュートリアルです。

## 質問掲示板について

JRA-VAN のソフト開発者向けサイト (<https://jra-van.jp/dlb/sdv/>) には三つの質問掲示板が存在します。それぞれ「JV-Data の使い方あれこれ」、「JV-Link 質問箱」、「プログラミング質問広場」です。「適宜確認するもの」の項で紹介した資料では分からない点があれば質問掲示板を確認するとその答えが載っている場合があります。ワード検索で過去記事を確認してみましょう。注意点としては、ワード検索で検索できるのは過去ログに含まれていない質問だけのようです。過去ログを調べる際は「過去ログ」を開き、ページ (P) ごとにワード検索を行うようにしましょう。それでも分からない場合は、質問したい内容が対応している質問掲示板において質問を試みましょう。公式や親切な方が回答してくれます。



## 2-2：諸々の機能

### Access2016 の導入

#### 今回の目的

Access とは、Microsoft 社が開発し提供しているデータベース管理ソフトのことです。正式名称は Microsoft Office Access です。

データを登録する DB を管理するために Access が必要になります。Word や Excel などを利用している場合、マイクロソフトとの契約内容によっては Access が使える場合がありますので、確認してみてください。Access が使えない場合でも新たに契約をする必要はありません。ランタイム版の Access をインストールすれば問題ありません。ランタイム版は DB の開発や操作に制限が掛かっているものですが、本同人誌では制限されている機能を使うことはないため、ランタイム版でも問題ないはずです。「Microsoft Office Access 2016 ランタイム」などのキーワードで検索してインストールを行ってください。なお、インストールするのはどんな PC であれ、32bit 版にしてください。JV-Link が 32bit で動いているため、64bit の Access を利用するとエラーが起きます。また、32bit 版の Access ランタイムをインストールする際に、既に 64bit 版の office をインストールしている場合はエラーが起きますので、事前に 64bit 版の office をアンインストールして 32bit 版の office に入れ直しましょう。

また、頭の片隅に入れておいて欲しいのが、Access は 2GB 以上の DB を扱うことができない点です。2GB 以上書き込もうとするとエラーが発生します。ですので、DB に書き込むデータ量が 2GB を超えないように登録するデータを調整する必要があります。2GB 以上の DB を扱う方法もあるにはあるのですが、その方法を取るより DB を分割した方が簡単です。

### JV-Link コントロールの追加

#### 今回の目的

JRA-VAN が提供しているクラスや関数を使用するために JV-Link コントロールをプロジェクトに追加します。JV-Link コントロールがなければ今後書いていくコードは動きません。

#### 下準備

「JRA-VAN Data Lab. SDK Ver〇〇」>「JV-Link」>「JV-Link.exe」をダブルクリックしてインストールしてください。

参考：DataLab.（データラボ） 会員サービス ソフト開発サポート>ステップ2 開発環境を整えよう！>Microsoft Visual Basic 2019> <https://jra-van.jp/dlb/sdv/workshop/20210526/get-kit.pdf>

## 追加方法

追加方法は以下の通りです。なお、以下の説明は JV-Link 質問箱にておこなわれた回答を整理・訂正したものです。リンクを張りたいところなのですが、メモしておいた URL はリンクが切れており、過去ログを探しても記事が探し出せないという状況です。No.6772 だということは分かっているのですが……。 (追記：どうやら記事が消えたらしいです) (参考：JV-Link 質問箱 7020)

「JRA-VAN Data Lab. 開発ガイド」の通りに JV-Link コントロールを追加しようとする「自己登録に失敗しました。」などのエラーが出るため、以下の方法で追加しましょう。

1. Visual Studio 上のメニューバーから「プロジェクト(P)」>「参照の追加(R)」
2. 参照マネージャの COM タブの一覧から「JRA-VAN Data Lab. 1.1.8 タイプ ライブラリ」にチェックをつけて OK ボタンをクリックする。
3. ソリューションエクスプローラーのプロジェクト名「KeibaSoft」下の「参照」を開き、「JVDTLibLib」を選択し、「相互運用型の埋め込み」プロパティを「False」にする。
4. JV-Link を使用する箇所に以下のように記述すると、開発ガイドのサンプルソース内の「AxJVLink1」の代わりに「clsJVLink」を使用できる。

ソフトの画面をダブルクリックするとソースコード画面が開きます。以下のコードは Form1\_Load イベントの中には書かないよう注意してください。Class Form1 の下に書いてください。

```
Public Class Form1
```

```
    (省略)    DB 作成機能 1 のコードが書かれている
```

```
    Dim clsJVLink As New JVDTLibLib.JVLinkClass
```

```
    (省略)    様々な処理
```

```
End Class
```

## 変更ポイント

今回はありません。

## 確認

デバッグを行い、エラーがでなければ問題ありません。

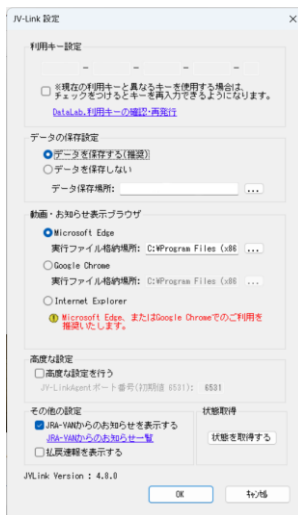
## JV-Link の設定変更機能 (JVSetUIProperties)

今回の目的

JV-Link の設定を行うためのダイアログを呼び出す機能を実装します。(写真)

このダイアログは、JRA-VAN を利用するために必要な利用キーを入力するために必要な機能となります。

(写真)



下準備

ボタンを配置します。ボタンの Name は「btnJVSetUI」、Text は「JV-Link 設定」にします。フォーム上に置いたボタンをダブルクリックすることでボタンクリックイベントが作成されます。

ソースコード：JV-Link の設定変更機能

```
1: Private Sub btnJVSetUI_Click(sender As Object, e As  
    EventArgs) Handles btnJVSetUI.Click  
2:     Dim ReturnCodeSetUIPro As Long =  
        clsJVLink.JVSetUIProperties()  
3:     If ReturnCodeSetUIPro <> 0 Then  
4:         MessageBox.Show("JVSetUIPropertiesエラー", "エラー",  
            MessageBoxButtons.OK, MessageBoxIcon.Error)  
5:     End If  
6:     End Su
```

## 解説

1：「JV-Link 設定」と書かれた btnJVSetUI ボタンをクリックしたらこの一連の処理が開始されます。

2：JVSetUIProperties メソッドを実行し、戻り値を ReturnCodeSetUIPro 変数に代入します。

3～5：if 文で戻り値（ReturnCodeSetUIPro）が 0（正常）以外の場合にメッセージボックスを表示してエラーを知らせます。

JVSetUIProperties メソッドの戻り値は 0 か-100 のどちらかです。「JV-Link インターフェース仕様書」>「3. コード表」の JVSetUIProperties を確認してください。戻り値が 0 の場合は正常であり、問題はありません。しかし、戻り値が-100 の場合はエラーが生じているのでそれを利用者に伝えないといけません。

そこで、4 行目の MessageBox クラスの Show メソッドを使用し、エラーを知らせています。

今回、エラーは一種類しかないため、エラーが起きたということだけを伝えていますが、2 種類以上のエラーがある場合はエラーコード、つまり戻り値を共に表示しましょう。

## 改変ポイント

今回はありません。

## 確認

開始を押して、ボタンを押しましょう。写真のようなダイアログが表示されれば問題ありません。

## JV-Link の初期化機能（JVInit）

### 今回の目的

JV-Link を初期化するための機能を実装します。競馬データ（JV-Data）を取得するためには、JVInit メソッドによる JV-Link の初期化が必ず必要なためです。

JVInit は JV-Link の初期化のために必ず最初に呼び出さないといけないため、Load イベントが発生する時、つまりソフトの画面が表示されるときに呼び出します。

JVInit メソッドには 4 種類の戻り値（正常+3つのエラーコード）が設定されています。ここでは、いちいちコード表を確認しなくて済むように、ここではメッセージボックスで戻り値と共にエラーの意味を表示することにしました。

### 下準備

JV-Link コントロールを追加した時にソフトの画面が表示された時に実行される Load イベントを追加しました。

なお、Load イベントを追加する方法は 2 通りあります。

- 1：Form1.vb[デザイン]画面において、ボタンが置かれていない Form1 画面のどこかをダブルクリックする。
- 2：タブの一つ下にある三つのドロップダウンリストをそれぞれ、ソフト名、(Form1 イベント)、Load にする。

ソースコード：JV-Link の初期化機能

```
1:      Private Sub Form1_Load(sender As Object, e As EventArgs)
      Handles MyBase.Load
2:          Dim ReturnCodeInit As Long = clsJVLink.JVInit("UNKNOWN")
3:          If ReturnCodeInit = -101 Then
4:              MessageBox.Show("JVInitエラー：引数が設定されていない", "エラー
      ", MessageBoxButtons.OK, MessageBoxIcon.Error)
5:          ElseIf ReturnCodeInit = -102 Then
6:              MessageBox.Show("JVInitエラー：引数が64byteを超えている", "
      エラー", MessageBoxButtons.OK, MessageBoxIcon.Error)
7:          ElseIf ReturnCodeInit = -103 Then
8:              MessageBox.Show("JVInitエラー：引数の1桁目がスペースになっている
      ", "エラー", MessageBoxButtons.OK, MessageBoxIcon.Error)
9:          End If
10:      End Sub
```

#### 解説

1：ソフトが起動し画面（Form1）が表示された時に一連の処理が実行されます。つまり一番初めに一回だけ実行されます。

2：JVInit メソッドを実行し、戻り値を ReturnCodeInit に代入します。

JVInit("UNKNOWN")の”UNKNOWN”はソフトウェア ID を記入します。ソフトウェア ID は JRA-VAN に登録済みのものである必要がありますが、”UNKNOWN”を記入しておけば問題なく動作します。

3～9：if 文で戻り値（ReturnCodeInit）が 0（正常）以外の場合にメッセージボックスを表示してエラーを知らせます。戻り値のみを表示してもいいのですが、エラー内容が簡単に修正できるものですので、いちいち「JV-Link インターフェース仕様書」を参照しないで済むように、戻り値とともに戻り値（エラーコード）の意味を表示するようにしています。エラーコードだけではどの処理工程でエラーが発生したのかわからないため、今後のことも考え、「JVInit エラー：」の文章も一緒にメッセージボックスに表示させます。

改変ポイント

今回はありません。

確認

メッセージボックスが表示されなければ成功です。

メッセージボックスが表示された場合はエラー内容を確認して修正しましょう。

## DB 作成機能（CreateDB）

今回の目的

JRA-VAN で取得したデータは自分のパソコンにある DB に登録（入力）されます。その後、その DB を叩いて情報を活用することになります。今回はデータを保管するための DB（Data.accdb と名付けます）を作成する機能を実装します。JRA-VAN 公式がそのための機能を用意してくれていますので、そちらを利用して実装します。

DB が作成される（エクスプローラー上で DB が配置される）場所は、  
/KeibaSoft/KeibaSoft/bin/Data.accdb となります。これは、JV-Data 登録クラスを使用する際にデータベースファイルが「bin」フォルダの中にある必要があるからです。

下準備

DataLab.（データラボ） 会員サービス ソフト開発サポート プログラミングパーツ・開発支援 ツール提供コーナー（<http://jra-van.jp/dlb/sdv/pgm.html>）の「データベース作成クラス」の項目から Microsoft Visual Basic 2019 版をダウンロードします。

zip ファイルを解凍すると色々なファイルがありますが、その中の「clsDBBuilder.vb」ファイルのみを使います。

ファイルをプロジェクトに追加するには、ソリューションエクスプローラーの「KeibaSoft」の部分を右クリックして「追加(D)」→「既存の項目(G)」を選択、ダイアログが表示されたら clsDBBuilder.vb ファイルを選択して「追加」をクリックします。ソリューションエクスプローラーに「clsDBBuilder.vb」が追加されていれば OK です。

次に、Visual Studio 上のメニューバーから「プロジェクト(P)」>「参照の追加(R)」、参照マネージャの COM タブの一覧から「Microsoft ActiveX Data Objects 6.1 Library」にチェックをつけて OK ボタンをクリックする。

ボタンを配置します。ボタンの Name は「btnDBCreate」、Text は「DB 作成」にします。

ソースコード：DB 作成機能 1

```

1: Public Class Form1
2:     Dim clsJVLink As New JVDTLabLib.JVLinkClass
3:     Private objDBControl As clsDBBuilder
4:
5:     'DBのパス
6:     Dim appPath As String =
        AppDomain.CurrentDomain.BaseDirectory.TrimEnd("\c") '動いているソ
        フトのパスを取得する。
7:     Private dbPath As String = appPath.Replace("\Debug", "") 'パ
        スから\Debugを""無に書きかえ。DebugだからDebugだけど完成したらReleaseになる
        ことに注意
8:     Private txtPath As String = dbPath + "\Data.accdb"
9:     ~~~~~

```

#### 解説1

- 1：以下のコードはボタニイベントに書くものではありません。最初から表示されている一番上の場  
所に書いていきます。
- 2：objDBControl 変数を宣言します。この変数は DB の作成・削除・最適化の際に使用します。
- 6：現在実行中のプログラム（KeibaSoft ソフト）のパスを取得しています。ソフトがCドライブに  
ない場合は適宜変更してください。
- 7：取得したパスから「\Debug」の文字を消します。モードの種類によって動いているプログラムの  
存在するファイルの位置が異なるため、Debug モードでソフトを起動している場合は「\Debug」  
の文字を消す必要がありますが、リリースモードでソフトを起動している場合は「\Release」に変更  
する必要があります。
- 8：「\Debug」を消したパスに「\Data.accdb」を付け加えます。
- 9：省略。この間に様々なコードが記述されます。

## ソースコード：DB 作成機能 2

```
1: Private Sub btnDBCreate_Click(sender As Object, e As
    EventArgs) Handles btnDBCreate.Click
2:     objDBControl = New clsDBBuilder
3:     If objDBControl.CreateDB(txtPath) = True Then
4:         MessageBox.Show(txtPath & "にDBを作成しました", "",
            MessageBoxButtons.OK, MessageBoxIcon.None)
5:     Else
6:         MessageBox.Show("エラー：DB作成に失敗しました", "エラー",
            MessageBoxButtons.OK, MessageBoxIcon.Error)
7:     End If
8:
9:     objDBControl = Nothing
10:    Debug.WriteLine(appPath)
11: End Sub
```

### 解説 2

- 1：「DB 作成」と書かれた btnDBCreate ボタンがクリックされた時に一連の処理が実行されます。
- 2：インスタンスを作成しています。New 演算子が必要になります。
- 3～7：CreateDB メソッドを実行すると同時に、if 文によって、エラーの有無を確認しています。このコードは以下のものと同じです。

ソースコード：3～7 と同じ処理

```
1: Dim ReturnBool As bool = objDBControl.CreateDB(txtPath)
2: If ReturnBool = True Then
3:     MessageBox.Show(txtPath & "にDBを作成しました",
        "", MessageBoxButtons.OK, MessageBoxIcon.None)
4: Else
5:     MessageBox.Show("エラー：DB作成に失敗しました", "エラー",
        MessageBoxButtons.OK, MessageBoxIcon.Error)
6: End If
```

- 9：役割を終えたため、インスタンスを削除してメモリを開放します。
- 10：DB の場所を出力ウィンドウにも表示します。メッセージボックスは消さないと他の動作ができません。しかしながら、メッセージボックスを消してしまうとメッセージボックスで表示した DB の場所が消えてしまいます。そこで、出力ウィンドウにも表示することでその問題を解決しています。



改変ポイント

今回はありません。

確認

ボタンをクリックしたら、txtPath (/KeibaSoft/KeibaSoft/bin/Data.accdb) の場所に DB が作成されていることを確認しましょう。

## DB 削除機能 (KillDB)

今回の目的

CreateDB メソッドで作成した DB を削除する機能を実装します。データベース作成クラス (clsDBBuilder.cls ファイル) にある KillDB メソッドを使います。

下準備

ボタンを配置します。ボタンの Name は「btnDBDelete」、Text は「DB 削除」にします。

ソースコード：DB 削除機能

```
1: Private Sub btnDBDelete_Click(sender As Object, e As
    EventArgs) Handles btnDBDelete.Click
2:     Dim result As DialogResult = MessageBox.Show("DBを削除して
        いいですか?", "確認", MessageBoxButtons.OKCancel,
        MessageBoxIcon.Exclamation)
3:     If result = DialogResult.OK Then
4:         objDBControl = New clsDBBuilder()
5:         If objDBControl.KillDB(txtPath) = True Then
6:             MessageBox.Show(txtPath & "を削除しました", "",
                MessageBoxButtons.OK, MessageBoxIcon.None)
7:         Else
8:             MessageBox.Show("エラー：削除に失敗しました", "エラー",
                MessageBoxButtons.OK, MessageBoxIcon.Error)
9:         End If
10:        objDBControl = Nothing
11:    ElseIf result = DialogResult.Cancel Then
12:        MessageBox.Show("削除をキャンセルしました", "",
            MessageBoxButtons.OK, MessageBoxIcon.None)
13:    End If
14: End Sub
```

#### 解説

- 1：「DB 削除」と書かれた btnDBDelete ボタンがクリックされた時に一連の処理が実行されます。
- 2：メッセージボックスを表示して DB を削除してもいいかを尋ねます。この場合、「DB を削除していいですか?」という文章とともに、「OK」と「キャンセル」ボタンのあるメッセージボックスが表示されます。その後、押されたボタンの種類が変数 result に代入されます。
- 3：if 文です。変数 result の中身が OK ボタンと同じならば 4~10 行の処理を実行します。
- 4：インスタンスを作成しています。New 演算子が必要になります。
- 5~9：KillDB メソッドを実行すると同時に、if 文によって、エラーの有無を確認しています。この一連の処理は DB 作成機能 2 の 3~7 行と同じです。
- 10：役割を終えたため、インスタンスを削除してメモリを開放します。
- 11：3 行目の if 文の ElseIf です。つまり、2 行目に表示されたメッセージボックスでキャンセルボタンが押されれば 12 行目の処理を実行します。
- 12：メッセージボックスで削除をキャンセルしたことを伝えます。削除しないので動作を終了します。

改変ポイント

今回はありません。

確認

「DB 削除」と書かれた btnDBDelete ボタンをクリックすると、「DB 作成」(btnDBCreate) ボタンを押して作成した DB が削除されたことを確認します。

## DB 最適化機能 (CompactDB)

今回の目的

CreateDB メソッドで作成した DB を最適化する機能を実装します。データベース作成クラス (clsDBBuilder.cls ファイル) にある CompactDB メソッドを使います。

下準備

ボタンを配置します。ボタンの Name は「btnDBOpt」、Text は「DB 最適化」にします。

ソースコード：DB 最適化機能

```
1: Private Sub btnDBOpt_Click(sender As Object, e As  
    EventArgs) Handles btnDBOpt.Click  
2:     objDBControl = New clsDBBuilder()  
3:     If objDBControl.CompactDB(txtPath) = True Then  
4:         MessageBox.Show(txtPath & "の最適化に成功しました", "",  
            MessageBoxButtons.OK, MessageBoxIcon.None)  
5:     Else  
6:         MessageBox.Show("エラー：最適化に失敗しました", "エラー",  
            MessageBoxButtons.OK, MessageBoxIcon.Error)  
7:     End If  
8:     objDBControl = Nothing  
9: End Sub
```

解説

1：「DB 最適化」と書かれた btnDBOpt ボタンがクリックされた時に一連の処理が実行されます。  
2：インスタンスを作成しています。New 演算子が必要になります。  
3～7：CompactDB メソッドを実行すると同時に、if 文によって、エラーの有無を確認しています。この一連の処理は DB 作成機能 2 の 3～7 行と同じです。

8：役割を終えたため、インスタンスを削除してメモリを開放します。

#### 改変ポイント

今回はありません。

#### 確認

「DB 最適化」と書かれた btnDBOpt ボタンをクリックして、エラーが表示されなければ問題ありません。もちろん、この確認は作成された DB が存在していなければいけません。

## DB との接続機能 (ConnectDBF)

#### 今回の目的

DB 作成機能 (CreateDB) で作成したデータベースと接続する機能を実装します。データベースと接続しなければ、データベースへの書き込みや中身を読み取ることができません。

#### 下準備

DataLab. (データラボ) 会員サービス ソフト開発サポート プログラミングパーツ・開発支援 ツール提供コーナー (<http://jra-van.jp/dlb/sdv/pgm.html>) の「JV-Data 登録クラス」の項目から Microsoft Visual Basic 2019 版をダウンロードします。

zip ファイルを解凍すると色々なファイルがありますが、その中の「source」フォルダ内の「basUtility.vb」、「basVar.bas」、「clsDBImport.vb」、「JVData\_Structure.vb」、そして、「clsImport〇〇.vb」(AV～YS までの 36 個) のファイルをプロジェクトに追加します。追加する方法は DB 作成機能 (CreateDB) の下準備の項目を確認してください。

そして、DataLab. (データラボ) 会員サービス ソフト開発サポート ソフトウェア開発キット (SDK) 提供コーナー (<http://jra-van.jp/dlb/sdv/sdk.html>) の JRA-VAN Data Lab. SDK 本体をダウンロードします。

zip ファイルを解凍すると色々なファイルがありますが、その中の「ドキュメント」フォルダを開き、「JV-Data 仕様書.xlsx」を開いてください。そして、下のシート一覧から「データ種別一覧」のシートを開いてください。すると、データ種別、レコード種別、収録内容が書かれた表があるはずです。

また、「JV-Data 登録クラス」フォルダの「document」フォルダ内にある「データベース仕様書.xls」も開いておきましょう。

## ソースコード：DB との接続機能 1

```
1:      Friend Sub ConnectDBF()  
2:          'DB接続文字列の取得  
3:          strConnectionString = "Provider =  
          Microsoft.ACE.OLEDB.16.0;" & "Data Source = " & txtPath  
4:          'データベースとの接続を行う  
5:          If ConnectDB() = True Then  
6:              ImportRA = New clsImportRA  
7:              ImportHR = New clsImportHR  
8:          Else  
9:              'ADODBオブジェクトの開放  
10:             gCon = Nothing  
11:             MessageBox.Show("データベースと接続できませんでした", "エラー",  
                MessageBoxButtons.OK, MessageBoxIcon.Error)  
12:         End If  
13:     End Sub
```

### 解説 1

1：今回はボタンイベントではないので、すべて自分で書く必要があります。以下のコードは ConnectDBF()関数が呼び出された時に実行されます。

3：DB 接続文字列の代入を行います。DB に Access2016 を使用しているため、ACE.OLEDB.16.0 と書いています。

5：basUtility.vb ファイルにある ConnectDB()関数を実行します。ConnectDBF()関数ではありません。ConnectDB()関数が問題なく実行できれば 6~7 行目を実行します。

6～7：データベースに入力したい情報が含まれているものだけを宣言します。

自分の入力したい情報が clsImport○○.vb のどれに含まれているのかを確認するために、下準備で開いた「JV-Data 仕様書.xlsx」と「データベース仕様書.xls」が必要になります。clsImport○○.vb の○○に含まれる英文字は「JV-Data 仕様書.xlsx」のレコード種別 ID と同じ英文字 2 文字になります。例：レース詳細は RA、競走馬マスタは UM です。

今回のコードでは、それぞれのレースの情報である RA とレースごとの払戻の情報である HR を宣言しています。

8：データベースとの接続が失敗した時に実行されます。

10：コネクション変数を空にします。gCon にはデータベースが代入されているという理解で問題ないと思います。

11：メッセージボックスでデータベースと接続出来なかったことを知らせます。

## ソースコード：DB との接続機能 2

```
1: Module basUtility
2: ~~~~~
3:     Public ImportRA As clsImportRA
4:     Public ImportHR As clsImportHR
5:
6:     Public strConnectionString As String
7: ~~~~~
```

### 解説 2

- 1：このソースコードは Form1.vb には書きません。basUtility.vb の一番上付近に追加します。
- 3：変数を宣言します。構文は Import○○と clsImport○○です。RA を使いたいのでどちらも RA です。
- 4：変数を宣言します。HR を使いたいのでどちらも HR です。
- 6：追加しないコードです。位置関係を表すためのものです。

### 改変ポイント

ソースコード 1 の 6～7 行目とソースコード 2 を改変することで、自分の求めている情報の取得・保存ができます。自分の求めている情報がどこに含まれているのかを確認するためには「JRA-VAN Data Lab. SDK 本体」の「JV-Data 仕様書.xlsx」と「JV-Data 登録クラス」の「データベース仕様書.xls」を参考にしてください。

### 確認

これ単体での確認はできませんので、後述する「ダウンロード機能+DB への登録機能」の項目において問題なく動くかを確認します。

とはいえ、デバッグを起動して無事にソフトが立ち上がるかの確認はしておきましょう。

## DB との切断機能 (CloseDBF)

### 今回の目的

DB 作成機能 (CreateDB) で作成したデータベースと ConnectDBF で接続しましたが、今度はその接続を解除する機能を実装します。データベースは適切に切断しないと面倒くさいことになるため、接続したら適宜切断をしましょう。タイミングとしてはデータベースに接続する必要がなくなったらきちんと切断するようにしましょう。また、CloseDBF は ConnectDBF と一対一で使うと覚えておきましょう。

## ソースコード：DB との切断機能

```
1:      Friend Sub CloseDBF()  
2:          'データベース登録クラスのクローズ  
3:          If ImportRA Is Nothing = False Then  
4:              ImportRA.Close()  
5:              ImportRA = Nothing  
6:          End If  
7:          If ImportHR Is Nothing = False Then  
8:              ImportHR.Close()  
9:              ImportHR = Nothing  
10:         End If  
11:  
12:         'データベースとの切断を行う。  
13:         If gCon Is Nothing = False Then  
14:             gCon.Close()  
15:             gCon = Nothing  
16:         End If  
17:     End Sub
```

### 解説

1：CloseDBF 関数が呼ばれた時に一連のコードは実行されます。

3～6：データベース登録クラスのクローズ（メモリ解放）を行います。

3：ImportRA が Nothing（何にも関連づけられていないの）ではないのであれば4～5行目が実行されます。つまり、ImportRA が関連付けられている（使用されている）のであれば4～5行目が実行されます。

4～5：ImportRA を閉じています。

7～10：同じように ImportHR を閉じています。

13：同じように gCon を閉じています。gCon はデータベースが代入されたものだと考えておけば問題ないと思います。

### 改変ポイント

ソースコードの3～10行を改変することができます。ConnectDBF メソッド内でデータベースと接続を行った clsImport○○.vb を対応させましょう。

## 確認

これ単体での確認はできませんので、後述する「ダウンロード機能+DB への登録機能」の項目において問題なく動くかを確認します。

とはいえ、デバッグを起動して無事にソフトが立ち上がるかの確認はしておきましょう。

## JV-Link 終了機能 (JVClose)

### 今回の目的

後述する JVOpen/JVRTOpen メソッドと共に使います。このメソッドを最後に実行することで JVOpen/JVRTOpen メソッドを正しく終了させることができます。JVOpen/JVRTOpen メソッド内に書いてもいいのですが、終了機能は複数回にわたって必要になるため、関数化を行いコードの簡略化を狙います。

### ソースコード：JV-Link 終了機能

```
1:      Friend Sub JVClose()  
2:          Dim ReturnCodeClose As Long = clsJVLink.JVClose()  
3:          If ReturnCodeClose <> 0 Then  
4:              MessageBox.Show("JVCloseエラー：" & ReturnCodeClose, "エ  
                ラー", MessageBoxButtons.OK, MessageBoxIcon.Error)  
5:          End If  
6:      End Sub
```

### 解説

1：JVClose 関数が呼ばれた時に一連のコードは実行されます。  
2：JVClose メソッドを実行し、戻り値を ReturnCodeClose に代入しています。  
3～5：if 文で戻り値 (ReturnCodeClose) が 0 (正常) 以外の場合にメッセージボックスを表示してエラーを知らせます。

### 改変ポイント

今回はありません。

## 確認

これ単体での確認はできませんので、後述する「ダウンロード機能+DB への登録機能」の項目において問題なく動くかを確認します。

とはいえ、デバッグを起動して無事にソフトが立ち上がるかの確認はしておきましょう。