最後のレッスンでは、コードを大幅に簡素化してリファクタリングしました。

そして今、それは非常にシンプルになり、読みやすくなりました。

しかし、一つだけあります。現在、特定のマージン、特定の外観、または特定の丸みを帯びた境界線を持つカードを作成できるように、reusableCardを使用していることを知っています。

ただし、ユーザーがそのカードをタップしたことを検出できるように、ジェスチャ検出器に子として追加し、それが発生した場合にselectedGenderを設定します。

現在、他の再利用可能なカードでは、タッチを検出できるこの機能が必要になる可能性が非常に高くなります。

必要がない場合でも、同じコードを繰り返し使用している2つの領域がありますか？

reusableCardにジェスチャー検出器を追加します。

では、なぜ弾丸を噛んで再利用可能カードをアップグレードして、タッチも検出できるようにしないのでしょうか？

そして、それがこのレッスンでやろうとしていることです。そしてその過程で、

関数を他のオブジェクトと同じように渡すDartでできる、本当にクールなものを紹介します。

そして、私は実際に自分のファイルの名前にタイプミスがあることに気づきました。

ここでは、入力ではなく入力を記述したことがわかりますが、これはコードに影響を与えないため、修正が非常に簡単です。

そのため、ファイル名にタイプミスやスペルミスがあった場合、それを右クリックしてリファクタリングをクリックし、名前を変更すると、これが使用されるすべての場所が変更されます。それで、参照を検索し、コメントと文字列を検索する入力を変更し、すべてを正しいスペルにリファクタリングします。

Dartでは、関数はファーストクラスの市民です。

これは、それらに型があり、文字列やintなどの他の型と同じように渡すことができることを意味します。また、変数または定数の値として設定することもできます。

それでは、これを見てみましょう。

ここで、私のDartPadには、2つの機能があります。

1つは、指定された2つの数値として単純に追加される関数です。

そして、2番目のものは乗算と呼ばれ、与えられた2つの数値を乗算します。

計算を実行したい場合は、たとえば3番と5番を追加してから入力すると、結果が得られます。

したがって、追加の結果に等しいint結果と呼び、コンソールに結果を出力します。

そして、あなたは私がそれをそのまま実行できることを見ることができます。

しかし、ユーザーがプラスボタンまたはマイナスボタンをタップして必要な計算を実行できる計算機を構築したい場合はどうすればよいですか？

代わりに毎回異なる関数を呼び出す代わりに、intを返す新しい関数を作成し、それが計算機と呼ばれ、int n1とint n2の2つの数値を取るだけでなく、関数を取ることもできたらどうでしょう引数として？

そのため、タイプとして追加できます。

したがって、文字列の場合はintの文字列になり、intになります。しかし、関数の場合は、大文字のFを使用してFunctionという名前を書くだけです。そして、名前を付けて計算と呼びましょう。

そして、関数の本体内で、これらの2つの数値n1とn2で計算を実行します。そして、この計算の結果を計算機の出力として返します。

今私ができることは、addを呼び出してから乗算してから何もする必要がないことです。この場合、結果を書くことができるのは、2つの数値で関数を実行する電卓です、5と8を呼び出しましょうここでの3番目の入力は関数です。

入力として実際の機能を期待しています。

そのため、単に名前を追加します。

たとえば、追加します。この場合、5と8で計算機が何らかの機能を実行し、それらを加算することは明らかです。

したがって、実行をクリックすると、5と8を加算しようとすることがわかります。

ただし、ここでパラメータを単純に乗算に切り替えた場合、正しいスペルである限り、5と8を乗算します。

したがって、関数は他の型と同じように渡されます。それがintであるか、文字列である場合です。

そして、これは私たちに多くの柔軟性を与え、これから見るように本当に強力です。関数でできることのもう1つは、変数の値として関数を割り当てることです。

したがって、たとえば、int計算機と言う代わりに、Function型の新しい変数を作成し、それを計算機と呼んで、ここで関数と等しくなるように設定します。そして最後に、セミコロンで終わらせるだけです。

そのため、今では以前とまったく同じように機能します。

[実行]をクリックして、加算または乗算に変更しても、実際には問題ではなく、そのまま機能します。私の関数は、計算機と呼ばれる変数内に保存され、必要に応じてこれを最終的な変数にすることさえできます。

つまり、電卓の割り当てを変更できなくなったため、たとえば電卓が別の機能と同等になったとは言えません。電卓は最終変数であり、一度しか設定できないことを伝えるエラーが表示されます。

現時点では、クラスの完全に外部で、メイン関数の外部でさえも関数を作成していることに注目してください。

そして、これはDartでは完全に合法です。

このようなトップレベルの関数を使用できますが、それらはクラスやオブジェクトに関連付けられていません。

しかし、クラス内でどのように機能するかを見てみましょう。

ここまでのコードをすべて削除し、Carという新しいクラスを作成します。そして、車には機能であるプロパティがあり、ドライブと呼ばれます。

これで、carが構築されると、このプロパティが初期化されます。

そのため、車のコンストラクターを作成し、this.driveを作成して、車を構築するときに、それを車のオブジェクトに関連付けるためのドライブメソッドを提供する必要があると言います。

2つの運転モードがあるとしましょう。 slowDriveと呼ばれるものを作成してみましょう。この場合、単に「ゆっくり運転」と印刷します。

そして、もう1つ高速ドライブと呼ばれるものがあるとします。この場合、「超高速運転」と表示されます。

大丈夫。

だから今私はトップレベルの関数である2つの関数を持っています。それらはどのクラスにも関連付けられていません。しかし、たとえば、メイン関数で車を初期化すると、myCarはCarクラスから構築された新しい車に等しいと言えます。

そして、今度は、ドライブプロパティの値を指定する必要があります。

それで、ドライブのプロパティは機能と同じに設定されますよね？

だから、機能を期待しています。

2のように数字を入力することはできません。または、型関数である必要があるため、文字列を入力することはできません。

必要があるので、機能を与えましょう。 slowDriveまたはfastDriveと言うことができます。

さて、私の車がslowDrive車になり始めたとしましょう。

関数の後に実際に括弧を追加していないことに注意してください。

その関数の名前を単に渡しているだけです。

したがって、この段階でmyCar.driveはslowDrive関数に関連付けられています。

したがって、print myCar.driveを実行するだけでこれを表示できます。

繰り返しますが、ドライブの後に括弧を入れていないことに注意してください。これは機能を有効にするためです。

そのため、今すぐ印刷すると、クロージャー「slowDive」が印刷されていることに気付くでしょう。

それがこの特定の機能を指しているのです。

myCarでは、ドライブプロパティはslowDriveと等しいと言っています。

しかし、実際にドライブメソッドを呼び出したい場合は、myCar.driveと言います。

そして、必要に応じて括弧と入力を追加します。

そして、私の車に関連付けられている駆動方法を実際にトリガーするのはこの行です。

そして、私たちはコンソールにゆっくりと印刷されています。

今、私は自分の車をアップグレードしたかったとしましょう。

myCar.driveと言いますが、ドライブプロパティはfastDriveと等しくなります。

また、メソッドの後に括弧はありません。

しかし、私は単にmyCarのそのドライブプロパティの値を変更しています。

だから今、私は車をガレージに連れて行った。

スープが出ています。今ではターボになり、過給され、燃え上がります。

そのため、myCar.driveを作成して結果を出力します。

だから私はその行をコメントアウトするつもりです。

そして今、myCarがアップグレードされたこと、そして今、それが超高速で走行していることがわかります。関数を渡すときは、その名前を使用しています。

括弧を追加していません。

そして、メソッドを呼び出して、必要に応じて括弧と入力を追加したときの機能を実際にトリガーしたいとき。他のオブジェクトや他のタイプと同様に、柔軟な関数がどのように渡され、どのように渡されるかを見てきたので、学習したことを私たちのコードに適用するのはあなたの挑戦の時です。

チャレンジの目標は、最初のアイコンカードと2番目のアイコンカードでジェスチャ検出機能を実行し、それをreusableCard.dartファイルにリファクタリングすることです。そのため、colorプロパティまたはcardChildを渡したように、実際に関数をreusableCardに渡すことができ、これらのプロパティの1つの値として関数も渡されます。

したがって、ジェスチャ検出器がreusableCard内およびこのビルドメソッド内に存在するという事実を除いて、最後まで成功した場合、アプリに関する変更はありません。

ビデオを一時停止し、この課題を解決できるかどうかを確認します。大丈夫。

まず最初に。このページからジェスチャ検出器を削除するだけです。したがって、OPTIONまたはALTを押しながらEnterキーを押し、両方の場所からウィジェットを削除します。

したがって、最初のカードと2番目のカードの両方に。

これで純粋なreusableCardができました。ユーザーがこのカードまたはこのカードをクリックしてreusableCardになったときに、状態を設定し、selectedGenderを変更する機能を引き継ぎます。

そのため、ここではbuildメソッドで、コンテナを返すだけでなく、実際にコンテナをジェスチャ検出器でラップします。

そしてもちろん、ここで再びonTapにアクセスできるようになります。そして、関数を私のreusableCardウィジェットに渡します。

したがって、前の2つのプロパティcolorとcardChildで行った方法とまったく同じ方法で、型として機能する別の最終プロパティを追加します。

そして、私はそれをonPressと呼びます。また、新しい再利用可能なカードを作成すると、これも初期化されます。

したがって、ここにthis.onPressを追加して、reusableCard内で使用する関数を渡すことができるようにします。

そして、onTapの値として設定されます。

つまり、これは、reusableCardを作成するときに、ジェスチャ検出器がreusableCardのタップを検出したときに呼び出される関数を指定できることを意味します。

これで、入力ページに戻り、reusableCardにいくつかのプロパティを追加できます。そのため、今定義したばかりのon Pressを追加できます。これは、ジェスチャ検出器で行っていたものと同じになります。

匿名関数を追加し、匿名関数の内部に設定状態と内部状態を設定し、最初のreusableCardと2番目の再利用可能なカードでselectedGenderをgender.maleに変更し、onPressとこの場合、設定状態はselectedGenderをfemaleに変更します。これで、コードで保存をクリックしても、アプリに関する変更はありません。前回とまったく同じように機能することがわかりますが、reusableCardの機能をリファクタリングしました。このカスタムウィジェットは、コンテナに基づいて、ボックス装飾に基づいて、ジェスチャ検出器に基づいて構築したものです。など。また、onPressを使用できるようになりました。したがって、タップに応答することができます。

そのため、onPressプロパティの値として関数を渡すだけでタッチを検出できるように、reusableCardを本質的にアップグレードしました。この結果、コードの埋め込みが少なくなり、すべての機能とデザインが1つのウィジェットにまとめられた、よりクリーンなコードになります。

コードをリファクタリングするのは常に良い習慣です。

この方法では、最後にスパゲッティコードで終わり、それを整理するためのすべてのモチベーションを失うことはありません。

あなたは、あなたの部屋が特定の状態になり、それを回復することがもはや不可能になったとき、あなたはあきらめることを知っています。

それは私たちのコードで望んでいることではありません。

私たちはそれがきちんと整頓されたパフォーマンスであることを望んでいます。

次のレッスンでは、ユーザーインターフェイスにある他のカードの作業を続けます。

そして、スケールでユーザーの身長を選択できるようにFlutterスライダーを実装します。

それで、それ以上については、次のレッスンで説明します。