さて、前回のレッスンで新しいキーワードに出くわしました。これは静的と呼ばれます。

これで、クラスや拡張、リターンなど、すでにかなりの数のキーワードを見てきました。

そして、このURLを見ると、実際にはDartが特別であると考える多くのキーワードがあることがわかります。

また、enumやelse、inまたはshowまたはhideなど、すでに多くの機能について説明しました。

しかし、このレッスンでこれから深く掘り下げていくのは静的です。また、静的により、クラス全体の変数およびメソッドを実装できます。

これはどういう意味ですか？

以前、クラスキーワードとそれに続くプロパティの定義を使用してクラスを作成できることを確認しました。

また、クラス内にメソッドを作成することもできます。

後で、車の設計図からの車など、このクラスからオブジェクトを構築できます。

そして、この構文でクラスの名前とそれに続く括弧のセットを使用して、クラスから新しいオブジェクトを作成することを忘れないでください。

そして、この括弧のセットがクラスのコンストラクターを引き起こします。これはデフォルトでは無料で取得できる単なる空のコンストラクターです。

しかし、もちろん独自のコンストラクタを作成したい場合は可能です。

this.numberOfDoorsでcarのようなものを言うと、このcarクラスから新しいcarを構築するときに、青写真からオブジェクトを作成します。

次に、ドアの数プロパティの値を指定する必要があります。

私の車は3ドア車だとしましょう。

この場合、このクラスからオブジェクトを構築する方法を変更します。

しかし、いずれにせよ、新しいオブジェクトを構築するために、実際にはまったく新しいオブジェクトを作成し、プロパティで初期化して機能を与えていることを示すために、常に括弧のセットを置きます。

しかし、別のクラスがある場合はどうでしょうか？

Squareと呼ばれるクラスがあり、このクラスにはデフォルトで4に設定されているnumberOfSidesとcolorプロパティの2つのプロパティしかないとします。

さて、正方形の青写真からmySquareオブジェクトを作成し、正方形の青写真からyourSquareオブジェクトを作成すると、mySquareとyourSquareが異なる色のプロパティを持つ可能性があります。

ですから、正方形の色を赤に、正方形を青に変えることができるはずです。

しかし、正方形の辺の数のようなプロパティはどうでしょうか？

まあ、それはすべての正方形で共通になるでしょう？

正方形にするためには、定義上、4つの辺が必要です。

正方形のオブジェクトに固有のものではありません。

この正方形に3つの辺を持たせることはできませんが、この正方形には4つの辺を持たせることはできません。

したがって、メイン関数で正方形の辺の数を知りたい場合は、すぐに正方形の外にある新しいオブジェクトを初期化して、辺の数というプロパティを利用できるようにする必要があります。

今ではそれはひどく無駄に思えますか？

クラスから新しいオブジェクトを作成するには、多くのリソースが必要です。側面の数だけでなく、色のプロパティも指定する必要があります。ほとんどのクラスと同様に、多くのメソッドと多くの処理が行われる可能性があります。

そのため、numberOfSidesなどのプロパティを取得できるようにするためだけに、新しい正方形のオブジェクトを作成するのに多大な労力を費やしているようです。正方形の辺の数を知りたいのはなぜでしょうか？

かなり明らかですね。

4になります。さて、ユースケースを少し明確にするために、別のクラスを作成しましょう。

それで、Icosagonと呼ばれるクラスがあったとしましょう。ちょうどそのスペルを確認してください。

イコサゴンにはいくつの側面がありますか？

ウィキペディアによると、20あります。

今、あなたはそれを思い出すことができないかもしれません、あなたは頭のてっぺんを知っています。そのため、実際には、Icosagonの面の数を使用または印刷できるようにする必要性が正当にあるかもしれません。そのため、初期化が必要になる場合があります。次に、いくつかの面をタップしてから、それを印刷して、20になります。

もちろん今は覚えています。

しかし、再び、同じ状況。 Icosagonオブジェクト全体を初期化する必要があります。このオブジェクトは二度と使用されません。後でこのオブジェクトを利用できるように、このオブジェクトを変数に指定することすらしません。

また、プロパティの1つの値を取得するためだけにオブジェクトを作成する必要がある前に、このようなコードを見てきました。

それでは、コードとコードを実行するリソースを使用して、どのように効率を高めることができますか？

この場合、辺の数を静的プロパティに変更できます。

そのためには、静的という単語を前に追加するだけです。

そしてこれはもちろん、この変数をこのクラスのインスタンスに関連付けられているため、作成されたIcosagonオブジェクトに関連付けられているため、インスタンス変数と呼ばれる沼地の標準からこの変数を変更する修飾子です。静的キーワードを前に追加することで、クラス全体の変数に変換できました。そして、クラス変数を使用するために、実際にオブジェクトを作成する必要はありません。

「このクラスのプロパティは何ですか？」と言うことができます。

考えてみると、新しい正方形を作成するたびに、辺の数は常に4になります。

新しいIcosagonを作成するたびに、サイドの数は常に20になります。

これはオブジェクトごとに変わるものではありません。

そのクラスから作成されたすべてのオブジェクトに対して汎用的になります。

したがって、numberOfSidesは完全な静的変数を作成します。

次に、別のクラスを作成しましょう。

そこで、Circleというクラスを作成します。

そして、高校の数学から何かを覚えていて、円で何かをするために、一定の円周率を使用することは非常に重要ですか？

そのため、サークルまたはそのカテゴリに含まれる何かを扱う必要があるときはいつでも、常に便利なpiを持ちたいと思うでしょう。それでは、Circleクラスでpiというdouble型の定数を作成し、3.141592に設定しましょう。

これは、piで覚えられる小数点以下の桁数だと思います。

これを行うとすぐに、静的フィールドのみがクラス内でconstとして宣言できることを伝えるエラーが表示されます。

以前は、constを使用していたとき、通常は定数ファイル内にある種のフリーフローティング定数のように、常にクラスの外で作成していました。

しかし、実際に定数値をクラスに関連付けたい場合はどうでしょうか？

それを行うには、静的なconstに変換する必要があります。

何故ですか？なぜ静的である必要があるのですか？

理由は、既にデフォルト値を持つconstを持つことにより、このcircleクラスから作成されるすべてのオブジェクトがPiというこのプロパティに対して同じ値を持ち、変更されないためです。

それについて考えると、静的変数がどのように正しく機能するかのようにひどく聞こえますか？

そのため、静的ではない場合、円のプロパティである定数を作成することは意味がありません。

そのため、クラスに定数値を関連付ける場合は、常に静的定数にする必要があります。

これまで静的修飾子をプロパティの概念と変数に適用することだけを見てきましたが、メソッドに適用することもできます。

したがって、workOutCircumferenceというメソッドがあり、ワークアウトする円周の円周の半径の形式で入力を提供する必要があるとしましょう。そして、メソッドの本体の内部で、円周を計算します

2つのpi rの古い式を使用することにより、2 \* piの値に半径を掛けたものになります。そして、それはすべての円の円周に等しくなります。

最後に、メソッドは円周をコンソールに出力します。

したがって、メインに戻って以前の印刷機能をコメントアウトすると、先に進み、新しいサークルを作成します。

次に、そのクラスから来るworkOutCircumferenceメソッドを呼び出して、半径を指定します。

それでは、半径が15.6に等しいとしましょう。

したがって、runを押すと、半径15.6の円の円周として、コンソールに98ポイントの何かが印刷されていることがわかります。

この場合も、このメソッドを使用できるようにするためだけに円を作成しています。

それはそのクラスに関連付けられています。

そのサークルクラスを後で必要とするつもりはありません。なぜなら、もしそうしたければ、そのサークルオブジェクトを保持する変数を作成していたからです。

そのため、後でそのサークルオブジェクトを別のシナリオで再び使用したり、そのプロパティの1つにアクセスしたりできるようになりました。

オブジェクトを変数内に保存し、さまざまなタスクで繰り返し使用します。しかし、多くの場合、クラスからのメソッドまたはプロパティが必要なだけです。

そして、この場合、オブジェクトのメソッドまたはプロパティをすぐに使用できるようにするためにオブジェクトをインラインで構築するときは常に、このようなコードを記述しているときは、静的キーワードを何でも追加することで恩恵を受けるでしょう。

この場合、静的キーワードをメソッドに追加することもできます。

そのため、このメソッドは静的なvoidになり、このメソッドはクラスから構築されたオブジェクトではなく、クラスに関連付けられたメソッドになりました。

そのため、そのメソッドを使用できるようにするためだけに新しいCircleオブジェクトを作成する代わりに、

そのようにクラスメソッドを使用するだけで、何も構築する必要はありません。

サークルクラスに関連付けられているメソッドを使用したい場合は、この方法の方がはるかに効率的です。

さて、ウェルカム画面の例に戻ります。main.dartのウェルカム画面IDを利用できるようにしたい場合、これが静的変数でない限り、非常に単純なことをするためにウェルカム画面オブジェクト全体をインスタンス化する必要があり、ウェルカム画面の状態を作成する必要があります。もちろん、足場、パディング、作成されているすべてのものを含むウィジェットを作成する必要があります。 。これを静的変数に変更することで、すべての重いリフティングをバイパスし、代わりにクラスに関連付けられたそのプロパティにホップするだけで、はるかに効率的になります。さらに良いことに、これを静的なconstに変更して、メイン画面や登録画面などの別のクラスのどこかに誤って変更しないようにして、常にこの値のままにすることができます。

そのため、その文字列は、1つの場所とそれを参照する他のすべての場所にある文字列としてのみ、その静的プロパティを使用します。