さて、以前のレッスンでは、Dartコンストラクターを見始めました。これらは、クラスからオブジェクトを作成できるようにするために、Dartクラス内に記述するコードの一部です。

したがって、クラスがオブジェクトの設計図であり、最終的なオブジェクトが持つプロパティ、実行可能なメソッドを計画する場合、コンストラクターは、実際にこの設計図を実際のオブジェクトに変換するコードの一部です。

ですから、もしそれが家の計画であれば、建設者は実際にそれを実際の家に変える建築者になります。

これで、ヒューマンクラスを作成したときに、以前のコードでクラスにプロパティを設定できることがわかりました。

そして、これらはクラス内に存在する単なるインスタンス変数です。

そして、クラス内にある関数の本質的に別の単語であるメソッドがあります。

クラス内で言うと、基本的にはこのクラスから作成される最終的なオブジェクトを意味するため、この場合は人間のオブジェクトであり、これらのプロパティがあり、これらのメソッドを実行できます。

したがって、これらのメソッドとプロパティは、単に自由に動き回るのではなく、実際にオブジェクトに関連付けられています。

さて、私たちがすでに見た3番目のものはコンストラクタです。また、これには特別な構文がありますが、本質的には、この場合の人間のオブジェクトの高さなど、プロパティの値を初期化するものをクラスに伝えます。

したがって、実際にこのクラスの設計図からオブジェクトを作成する場合は、そのコンストラクターを使用してそれを実行します。

そのため、このようなコードを作成します。

このクラスhumanのタイプを持つオブジェクトを作成しています。オブジェクトはJennyと呼ばれ、作成されます。

そのため、この式の右辺では、この特定のコンストラクタを使用して、「human」という単語で開始し、開始高さの入力を指定します。これは、そのheightプロパティに割り当てられます。コンストラクターを作成するときは、最終的にオブジェクトを作成するときに、オブジェクトの作成時点で値を使用して初期化するプロパティを指定できるように、単に事前計画を立てるだけです。

コンストラクターの動作のこの部分は、すでに見ました。そして、Humanクラスを使用してJennyオブジェクトまたはJamesオブジェクトを作成しました。この特定のオブジェクトのheightプロパティの初期値を提供しました。その後、必要に応じてその高さプロパティを利用したり、その高さプロパティを変更したりできます。しかし、このheightプロパティはこのオブジェクトに関連付けられているため、ドット表記を書くことでこのオブジェクトを取得できます。しかし、この高さは異なるオブジェクトに関連付けられているため、ここで見ることができるように、この高さとは異なります。

そして、彼らがこれらの開始値を持っている理由は、クラスで開始値を与えたからではありません。そして、それらはすべて値20で始まります。しかし、代わりに、ここでコンストラクタを使用してそのオブジェクトを構築したときにのみ値を指定しました。そして、これは多くの場合、赤ちゃんがデフォルトの身長で生まれる傾向がないという理由だけでなく、たとえばゲームでは、ユーザー名の値が正しくない可能性があるため、非常に理にかなっていますか？

ユーザーが実際に入力した場合にのみ、そのユーザー名の値にアクセスできます。

頻繁に、値なしで開始するクラスを作成します。したがって、高さは現在nullに等しくなります。

そして、それが初期化されたとき、または構築されたときのみ、実際にそのプロパティに割り当てる値を与えます。これでデフォルトで、コンストラクタを作成しなかったとしても、コメントアウトしたとしましょう。デフォルトのコンストラクタもあります。実際、このようなことを言うことができますか？

私は人間のクラスから新しいジェニーオブジェクトを作成していると言えます。ジェニーはジェームズと同じように高さプロパティを持っていますが、その高さプロパティはこの値が何であれ、現在はnullに等しくなります。

そのため、ここに出力されます。オブジェクトを使用してその値を変更する必要があります。

したがって、同じことを実現するには、jenny.heightが15に等しくなるように言わなければなりません。

また、初期値が必要なプロパティが多数ある場合、これらの2つの手順を使用してこれを行うには多くの時間がかかると想像できます。

これが、コンストラクターが便利な理由です。

これが、デフォルトでDartが無料のコンストラクタを提供する理由でもあります。

したがって、私の人間のクラスでは、現時点ではコンストラクターがないことがわかります。ただし、デフォルトでは、Dartは次のようなコンストラクタを提供します。基本的には、値も入力も受け取らない空のコンストラクターです。

また、人間のクラスから人間のオブジェクトを作成する以外には何もしません。常に入力するため、入力する必要はありません。現時点では、このコンストラクターがあるので、コメントを外してコードに戻します。また、startingHeightというこのパラメーターを使用して、このクラスから新しいオブジェクトを構築し、プロパティまたは高さをその入力の値（startingHeight）に設定するときに渡すことができるようにします。

そのため、startingHeightを記述して値を指定することができます。

これにより、コードが以前の状態に戻ります。

したがって、これらのコンストラクタに基づいて、ジェニーの開始高さは15で、ジェームズの開始高さは20です。

しかし、startingHeightと呼びたくない場合はどうでしょうか？

単に高さと呼ぶ方が理にかなっているとしたらどうでしょう？

入力は高さと呼ばれ、プロパティは高さと呼ばれます。

したがって、高さを同じ高さに設定します。

そして、人間のジェニーを初期化するとき、身長15の新しい人間を作成していると言います。

それで、コードのこの部分は、より自然に正しく読み取れますか？

ジェニーと呼ばれるヒューマンタイプの新しいオブジェクトを作成しています。このオブジェクトは、この開始時の高さビジネスではなく、高さ15の人間クラスから作成されます。

しかし、ここで、これは混乱を招きます。これは、どの高さを設定しているのか、どの高さを設定しているのですか？

これは、この値を取得して、この値に割り当てようとしているためです。

しかし、それはこの高さですか、それともこの高さですか？

本当に紛らわしいです。

実際、それを実行すると、コードも混乱します。なぜなら、ジェニーの身長とジェームズの身長の両方がヌルになるのは、コードがこの身長が実際にこの身長について話していると考えているからです。

これも同じです。

つまり、この入力、ここではこの変数は15になります。

基本的に、15は15に等しくなければならないということです。

そして、ここに高さを印刷するだけでそれを確認できます。

ジェニーとジェームズが構築されると、15と20が表示されます。

では、Jennyオブジェクトを作成すると15が印刷され、Jamesオブジェクトを作成すると20が印刷されることがわかります。

しかし、この特定の高さには実際には値が割り当てられていません。

したがって、これは私たちが望むものではありません。

また、コードが非常に読みにくくなります。

それで、これを回避する方法は何ですか？

Dartでは、キーワードコードを使用できます。したがって、this.heightはheightと等しいと言えます。 JavaScriptやJavaなどの他のプログラミング言語を使用している場合は、これも見たことがあるかもしれません。

しかし、基本的にthisという言葉を書き始めるとき、それはこのクラスから作成される最終的なオブジェクトを指します。

したがって、この場合はJennyになり、この場合はJamesになります。そして、この最終的なオブジェクトの高さをこのプロパティに設定し、入力として通過する高さに設定すると言っています。

したがって、ここでこのprintステートメントを削除し、高さが高さに等しくなった場合、高さはそれ自体に設定されます。

したがって、15は15に等しくなり、これは何にも影響しません。

そして、それを実行すると、ジェニーの身長に対してnullを、ジェームズの身長に対してnullを取得することを知っています。

しかし、これをthis.heightに変更するとすぐに、heightはheightに等しくなります。これは、このヒューマンクラスのheightプロパティが、コンストラクタから渡される高さに設定されることを意味します。次に、実行した場合、突然動作し、コードが参照しているものを認識できるようになります。

this.heightについて話しているときと、単に高さについて話しているとき。それで、すべての人間のオブジェクトにも開始重量プロパティも与えたい場合はどうでしょうか？

それでは、重みと呼ばれる別のプロパティを追加しましょう。

また、開始値がないため、何にも等しく設定していません。

そして、人間のコンストラクターの中に、重みと呼ばれるdoubleを追加しています。

そして、this.weightはweightと等しいと書きます。

したがって、ジェニーの開始重量は3.5 Kgになります。そして、ジェームズにも開始重量を与えます。

たぶん、彼は4.2キロになるでしょう。そして、これは機能しますが、これが機能するためにコンストラクターでどれだけの作業をしなければならないかを見ました。

入力パラメーターとして追加する必要がありました。

次に、このプロパティがここで渡されるものと等しくなるように指定するために、this.weightはweightに等しいと書く必要がありました。

また、初期化するプロパティが10個または20個ある場合、これには非常に時間がかかります。

そのため、これは非常に一般的な動作であるため、Dartには実際には少しの構文糖衣があります。

このプロセスを簡単にするために作成したものです。

したがって、このすべてを実行する代わりに、中かっこを含む中かっこ内のすべてを単純に削除できます。そしてこれを書く代わりに、これは身長と呼ばれる入力であり、これは入力コードの重みであり、単純にthis.heightとthis.weightを書くことができます。

そして、セミコロンでそれを締めくくります。

これらの中括弧について話しているときも同じことが当てはまります。

したがって、これらのパラメーターの周りにそれらがある場合、人間を構築するときに名前が付けられます。

しかし、これらの中括弧を削除する場合、これらの名前付きパラメーターを省略して、ここで定義された順序で初期化することができます。

入力パラメータが多すぎると混乱するので、このように見えるのが好きです。

ただし、これにより、クラスのプロパティの一部を初期化するために記述する必要があるコードの量が劇的に削減されます。

したがって、このコード行は、このすべてのコードが行うこととまったく同じことを行いますが、かなり削減します。そして、それはずっと簡単に見えます。

さあ、あなたの挑戦の時です。質問クラスで覚えている場合、以前のDartPadで見たのと同じ形式がまだあります。

したがって、今学んだことを使用して、このコードを再フォーマットして、Dartが提供する構文シュガーを使用して、質問クラスのコンストラクタを大幅に短く、はるかに単純にします。

そのため、ビデオを一時停止して、このチャレンジを完了してみてください。大丈夫。

したがって、これは非常に簡単なはずです。

質問クラスへの入力はここに入り、コンストラクタ内でプロパティと等しくなるように設定しています。

したがって、このコードをすべて使用する代わりに、コンストラクターの本体全体を削除してセミコロンを追加できます。

そして、入力を持ち、入力のデータ型を持つ代わりに、括弧の中に、私たちが書く必要があるのはこれだけです。次に、最初に初期化するプロパティの名前、questionTextです。そして、this.question Answerを初期化する2番目のものになります。そして、これは、質問テキストと質問の回答を渡す新しい質問オブジェクトを初期化するときに、QuizBrainのコードがまったく同じように機能することを意味します。

しかし、今では、このプロセスを少し冗長にするDartの何かを使用して、このコンストラクタを大幅に単純化しました。 Dartコンストラクターの詳細を知りたい場合は、もちろんDart言語を使用して目的地に向かい、このレッスンで学んだすべてのことと、それを使ってできることについて話します。

もちろん、コースリソースにはこれへのリンクがあります。