さて、前回のレッスンで、クイズの頭脳内のプロパティの値を取得できることに加えて、この他のクラスを紹介しました。現在、他のクラスのプロパティを変更することもできます。また、場合によっては、アプリが予期しない動作をする可能性があります。

そして、OOPの2番目の柱であるカプセル化に進みます。

また、ワーカーを別のオフィスキューブにカプセル化して、自分で作業できるように、クラスとオブジェクトをカプセル化することもできます。

だから最近、Inc。Magazineで面白いクイズに出会いました。

あなたはマイクロマネージャーですか？そして、それはあなたが基本的に他のすべての人の時間をマイクロ管理しようとしているか、実際に他の人の仕事をしようとしているかどうかを確認するクイズです。クラスを抽象化し、異なるジョブを異なるロールと異なるクラスに分離しようとするとき、各クラスが個別のロールと個別のジョブを持ち、それらをシンプルに保つことが望まれます。

私たちは、彼らが自分の仕事に関心を持ち、自分の仕事をうまくやることを望んでいます。

したがって、ウェイタークラスがあり、シェフクラスがある場合、ウェイターがシェフの仕事に干渉することは望ましくありません。

ウェイターがちょうど来て、「あなたはそれを正しく調理していません。

このようにする必要があります。」

それはウェイターの仕事ではありません。

そして、これは本当に悲しいシェフにつながり、彼はウェイターによって細かく管理されていることに本当に腹を立てます。

しかし、それはまた、ウェイターの仕事がより複雑になることを意味します。そして、ウェイターがクラスの場合、彼はより多くのコードを持ち、シェフのコードとより混ざり合います。

そして、これは実際には複雑さではなく、より複雑になり、すべてのクラスを個別のモジュールに抽象化するという目的に反します。

私たちは何をしますか？

さて、ウェイターとシェフの間に壁を作ることができました。

そして、これが、キッチンが通常レストランから分離している理由です。

シェフが自分の仕事をすることができ、ウェイターが自分の仕事をすることができます。

そのため、シェフは食べ物の調理にのみ関心があり、ウェイターは注文の受け取りと人々への食べ物の配達にのみ関心があります。

そして、シェフとウェイターの間に小さな窓があるだけで、彼らは必ずやり取りする必要があるものがあります。

そのため、たとえばシェフがウェイターに食べ物を渡すとき。

それでは、どのようにコードに対してこれを行うことができますか？

さて、前回のレッスンで、おそらくmain.dartの正しい答えの価値を変更できないはずだと思いましたか？

質問の回答を変更することは、main.dartのビジネスではありません。クイズの脳と非常に関係があります。クイズの脳は、何が真実で何が偽かを制御する必要があります。

そして理想的には、質問に対する答えは決して変わらないはずです。

この場合、この質問バンクをカプセル化するために何ができますか？

さて、このプロパティをプライベートにすることができます。そして、プロパティの名前の前にアンダースコアを追加するだけで、Dartでそれを行います。

したがって、main.dartに戻り、質問バンクの価値を調整しようとすると、それはもう機能しません。

実際、その質問バンクにアクセスすることさえできなくなりました。

質問バンクは現在プライベートであるため、「クイズ脳から質問バンクをくれ」とは言えません。つまり、クイズ脳だけがこのプロパティのコンテンツにアクセスでき、他の誰もアクセスできないことを意味します。

では、どのようにしてアプリを引き続き機能させるのでしょうか？

さて、これらの質問と回答を提供するタスクを、クイズに関連するすべてのことを担当する人、クイズ脳に委任しましょう。

それでは、新しい関数を作成しましょう。 getQuestionText関数と呼ばれるものを作成しましょう。また、クラス内ではメソッドと呼ばれていますが、基本的には同じことです。また、質問の取得テキストには、現在の質問番号である整数を渡す必要があります。

では、実際に把握したい質問番号は何ですか？

そして、その質問番号の実際の質問テキストを返すか出力します。

したがって、それを表現する方法は、プライベートである質問バンクに起こりますが、同じクラス内にいるため、アクセスできます。そして、質問番号のインデックスで質問バンクを利用します。

では、どの質問が必要ですか？それがゼロに等しかった場合、この質問オブジェクトを取得し、その質問テキスト、つまり.questionTextを利用します。そして今、main.dartに戻って、私たちのビジネスのいずれにも干渉しないこのプライベートプロパティを取得しようとする代わりに、クイズの頭脳に質問テキストを取得してもらい、この特定の質問番号の質問テキスト。

したがって、それから始めることはゼロになります。

そして、クイズの脳に入り、クイズの脳の質問テキストを取得し、最初の質問を取得して、questionTextを取得します。これで、実際に質問バンクをカプセル化しただけでなく、他の関係者から干渉されないようになっただけでなく、実際にクイズアプリをもう少し抽象化したことがわかります。それで今、挑戦として、私たちが抱えている他の2つの問題に対処できるようにしてほしい。

これについては心配しないでください。

しかし、ユーザーがtrueを押したとき、およびユーザーがfalseを押したときに正しい答えを取得する必要があります。ビデオを一時停止し、そのチャレンジを完了します。

大丈夫。

まず最初に、クイズの頭脳に入り、クイズの頭脳に正しい答えを提供してもらいましょう。

ブール値を返すメソッドを作成します。getCorrectAnswerと呼ばれるか、それを呼び出したものは何でも構いません。そして、正しい答えを取得しようとしている質問番号を再度渡します。

そこで、質問バンクに、.questionAnswerを提供した質問番号で返します。そして今、戻ってmain.dartを実行し、コードを修正できます。

したがって、質問バンク全体をmain.dartに保持しようとする代わりに、クイズの頭脳に等しい正解と答えて、現在の質問番号の正解を取得します。そして、正しい答えがtrueに等しいか、Falseに等しい場合はここにあるかを比較することができます。

ここで同じ方法でこれを修正しましょう。

クイズ脳から正解を得るために、正解は等しくなるはずです。これでアプリが動作し、正しい答えと正しい質問を取得することができますが、もはやその質問バンクに干渉することはできなくなりました。

クイズの脳の質問バンクとは言えません。答えの1つをtrueに設定してみましょう。これは、質問バンクが独自のクラス内にカプセル化され、独自のクラスのみになったためです。

クイズ脳は、この質問バンクで何をすべきかを決定できます。

たとえば、main.dartで誤って答えを完全に台無しにして変更したり、質問を変更したり、質問バンクを削除したりすることはできません。

そのため、カプセル化により少し安全になりました。現時点でアプリを実行すると、まったく同じように機能することがわかります。ただし、13の質問の最後に到達すると、まだクラッシュします。

質問は13個しかありませんが、質問バンクから13番目のアイテムを引き出しようとすると、もちろんリストがゼロからカウントされるため、これは14番目の質問に相当します。

では、これについて何ができるでしょうか？

クイズの頭の中では、質問バンクにまだ質問が残っているかどうかを確認できるはずです。

もしそうなら、私たちはそれを提供しますが、そうでなければ、実際には存在しない質問を手に入れようとしないことでアプリを保護する必要があります。

では、どうすればこれを行うことができますか？

まあ、実際には仕事ではないmain.dartのどの質問を追跡するのではなく、これを削除してクイズの頭脳に入れましょう。そのため、クイズの頭脳クラスにプロパティとして配置し、実際にはプライベートプロパティにします。

したがって、クイズ脳だけがこれにアクセスできます。だから私は誤ってmain.dartや質問で言うことはできません

.dartが突然質問番号を12または13に設定すると、アプリがクラッシュします。そして、getQuestionTextを呼び出してmain.dartから質問番号を提供する代わりに、クイズ脳内でquestionNumberを使用します。そのため、ここでそのプライベート質問番号に変更する必要があります。そして、次の質問に進むために、nextQuestionと呼ばれるメソッドを作成します。実際に次の質問に進むのが安全かどうかを確認するだけです。それはつまり、questionNumberがquestionBank.lengthよりも小さいかどうかを確認する必要があるということです。

したがって、.lengthがquestionBankのサイズを提供することがわかります。

質問バンクには13個のアイテムがあるため、これは13になります。

そして、質問数がそれよりも少ない場合、理論的には、次の質問に進んでも安全であるべきだということです。

したがって、questionNumber ++と言うことができます。そして、これは次の質問に行く必要があるたびにquestionNumberをインクリメントするはずです。

したがって、main.dartでは、questionNumberを更新する代わりに、代わりに次の質問に進むようにクイズの脳を呼び出します。また、質問番号を処理したり、main.dart内で追跡する必要がなくなりました。

両方の場所でそれを変更しましょう。

quizBrain.nextQuestionになります。そして、このprintステートメントを削除します。また、質問の回答を取得するのに入力が不要になり、質問のテキストを取得するのに入力が不要になることも覚えています。

そこで、実行をクリックするか、ホットリスタートをクリックして、最初から実行しましょう。そして、クイズの最初から始めなければなりません。

クイズを最後までやり終えると、アプリがまだクラッシュしていることがわかります。まだリストの13番目の要素を取得しようとしているが、それは12にしかならないということです。

だからここの理由は何ですか？

さて、ここにprintステートメントを追加して、質問番号が何であるかを確認しましょう。

また、questionBank.lengthも出力します。そして、アプリを最初からもう一度実行してみましょう。これで、最初にtrueをクリックして次の質問に進むと、質問番号は1になります。

したがって、次の質問に進み、合計13の質問があります。そのため、アプリのクラッシュに近づくまでtrueを押し続けましょう。

したがって、現在、質問番号は12です。そして、まだ存在するquestionBankから12番目の項目を引き出しようとしていることを意味します。

ここにある最後の質問、ウェストバージニアに関する質問です。

次にtrueボタンをクリックすると、次の質問メソッドがトリガーされます。そして、この時点での質問番号（12）がquestionBank.length（13）よりも小さいかどうかを確認します。

これは本当です。

そのため、次の質問に進み、質問テキストを更新します。そして、リストから13番目のアイテムを抜き出そうとしますが、それは存在しません。

だから今、これがアプリがクラッシュするポイントです。

では、この問題をどのように解決できますか？

まあ、questionBank.lengthでquestionBankのサイズよりも1つ小さいかどうかを確認するだけです。

今では実際に13に到達することはありません。

12に到達し、次にクリックすると、他に何も起こりません。

クイズが終わり、アプリがクラッシュしなくなりました。

パーフェクト。そのため、クイズの脳内でアプリの安全性を確認しています。これは、クイズに関連するすべてのことを処理する論理的な場所です。

そして、main.dartは単に画面上に物を表示するか、ボタンをクリックして次の質問に進むことだけに関心があり、クイズ機能に対処する必要はありません。