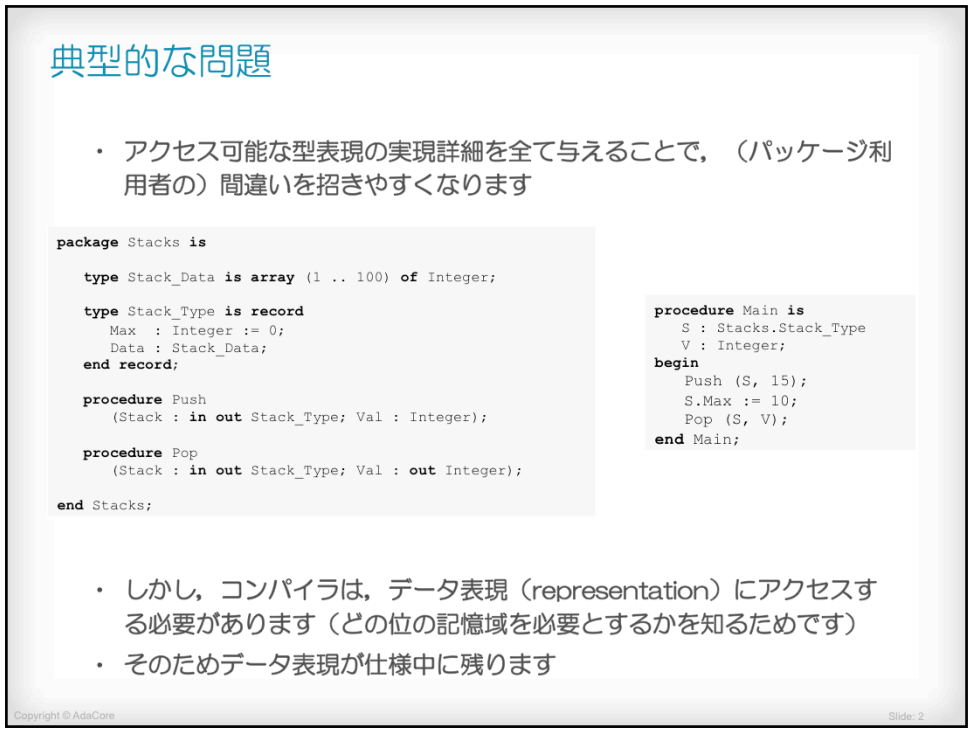
こんにちは．私の名前は Martyn Pike です．大規模プログラムについての Ada University の講義に，ようこそ．

この講義の主題は，Ada プログラム設計のパターンとしてのカプセル化に関するものです．最初に一連のスライドによる学習を終えた後，幾つかのクイズを通して，学んだことを確認します．



プログラミングにおける典型的な問題の一つは，既存のパッケージやモジュールに関して，利用者に対して，どの程度の情報を与えるかということです．パッケージの開発者は，首尾一貫した API を，利用者に提供する必要があります．また，（提供後）変化しやすい情報の量を最小化する必要があります．型をどのように表現するかを考えるときに同様の注意が必要となります．

アクセス可能な型の実現の詳細の全てを，利用者に与えると間違いが生じやすくなります．このスライドの例では，パッケージの利用者に対して，データ表現を示している，2 つの完全な型表現があります．

前述のパッケージ利用者は，例にある Main の手続きのように，型の要素に対してアクセスすることができます．

これは，保守の問題を引き起こします．（パッケージ開発者が）意図しないやりかたで（パッケージ利用者が）パッケージを利用するかもしれません．

しかし，コンパイラは，どの位の記憶域を使用する必要があるかを知るために，データ表現にアクセスする必要があります．

従って，データ表現は，仕様中に存在する必要があります．しかし，（利用者にとっては）非公開である必要があります．

先のスライドで述べたことを実現するために，パッケージ仕様は，非公開領域を持つことが可能です．コンパイラ・ボディ部・子パッケージでは可視となり，パッケージの利用者には不可視のスコープとなります．

C++ におけるクラス定義中の private キーワードをよくご存じかもしれません．非公開領域に関しては，Ada と C++ で少し違いがあります． Ada では，非公開は型全体を対象とし，属性毎に記述することはできません．

二番目として，管理される非公開性は，パッケージレベルにおいて管理され，クラスレベルではありません．

次に，非公開領域について例を見ます． C++ と比較したいと思いますが …

さて，質問を拡張したいと思います．パッケージ仕様部の非公開領域に誰がアクセスするのか，です．

答えは，パッケージのボディ部と子パッケージです．

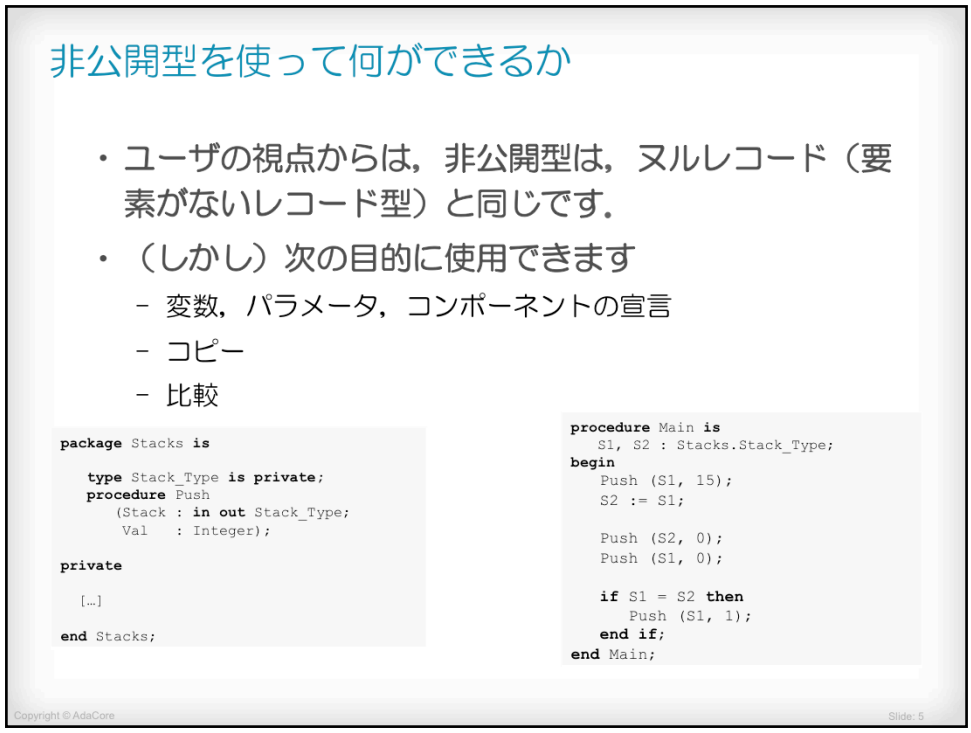
この例にあるコードでは， Stack_Type 非公開型と，引数として Stack_Type 型と整数を受け取る手続き Push が，パッケージ Stack 中にあります．パッケージ仕様の非公開領域には， Stack_Type をどう実現しているかが書かれています．

Stack パッケージのボディ部は， Stack_Type の実現にアクセスすることができます．また，レコード型の要素に直接アクセスすることさえできます．

次に， Stack の Utils 子パッケージのパッケージ仕様を見ます．この子パッケージでは，親の Stack パッケージ中で宣言されている非公開 Stack_Type 型によって型付けされたパラメータを受け取る手続き Empty を記述しています．

子パッケージの実現は， Stack_Type の実現にアクセスすることができます．実際に直接レコードの要素にアクセスすることもできます．

しかし，直接レコードの要素にアクセスしようとする Main 手続き中のあらゆる試みは，エラーとなります．



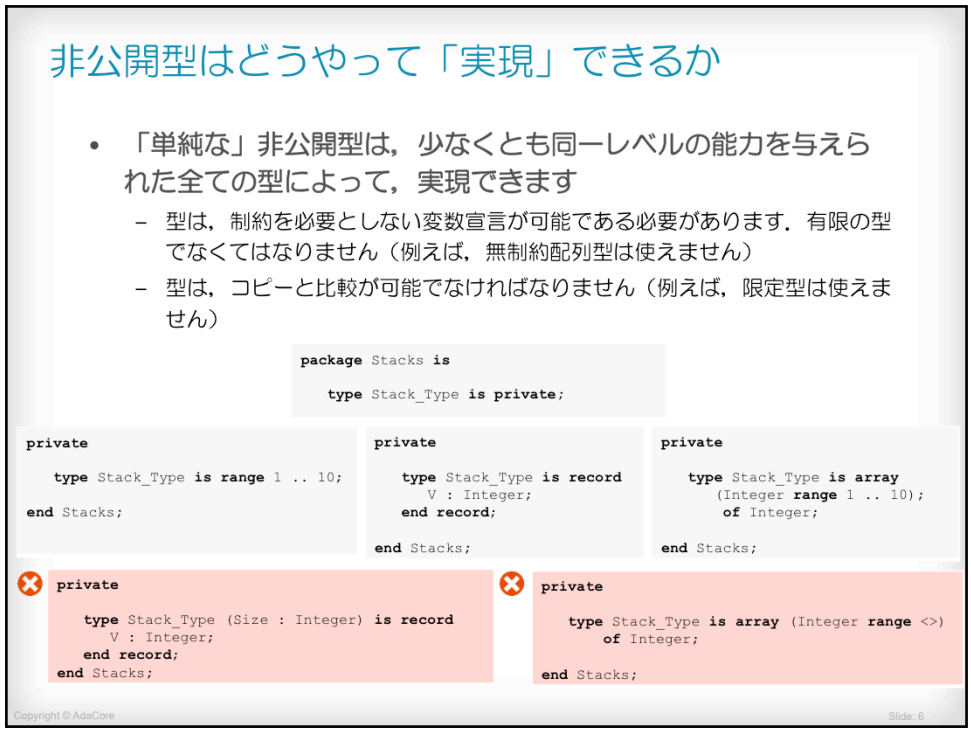
パッケージ利用者は，非公開型の実現を，利用することができません．しかし，ヌルレコード型と同様にその型を利用することができます．

パッケージ利用者は，自由にオブジェクトを型付けたり，パラメータ型や，コンポーネント宣言で用いることができます．

また，コピーすることができ，比較で用いることもできます．

次の例では，Stack_Type と呼ぶ非公開型を宣言しているパッケージ Stack を示しています．ここでは，非公開型を，手続き Push のパラメータとして使用しています．

Main 手続きは，パッケージ Stack の利用者であり，非公開 Stack_Type を用いて，スタックオブジェクトを宣言することができます．また，Push を呼び出すために用いたり，代入や比較を行うことができます．



非公開型には2つのタイプがあります．単純な非公開型と，不定の非公開型です．ここでは，単純な非公開型のみを考えます．

単純な非公開型は，少なくとも同一レベルの能力を与えられている全ての型によって実現できます．

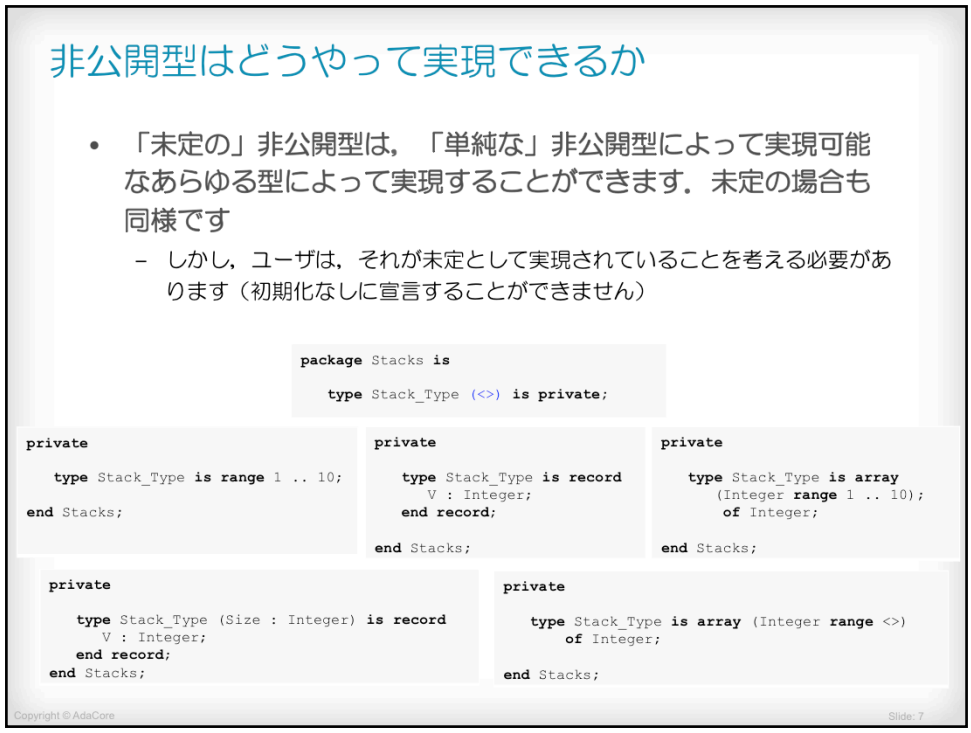
確定型であり，コピーと比較ができる必要があります．この理由から，無制約配列型や限定型は除かれます．

Stack_Type の単純な非公開型宣言を示します．ここでは，明確な範囲を持つ数値型の例，レコード型の例，そして，制約付き配列型を用いて実現した 3 つの正しい例を示しています．

不正な 2 つの例を，理解の助けとなるよう追加しています．区別子付きレコード型，或いは無制約配列型を使うことはできません．

訳注；

限定型（limited type）：　同一型でも代入ができない，事前に等値演算子を持たない型のこと



無制約の非公開型は，単純な非公開型で可能である実現の上位集合として実現することができます．

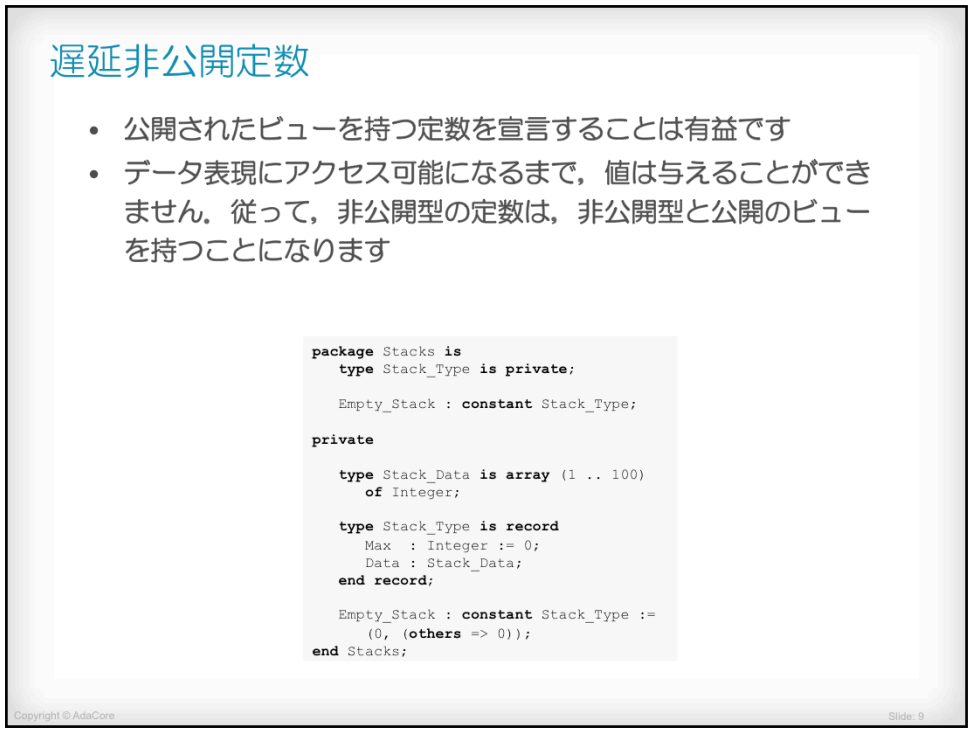
このことは，数値範囲，レコード型，制約付き配列型，区別子付きレコード型，無制約配列型によって，未定非公開型を実現できることを意味しています．

パッケージの利用者は，オブジェクトを記憶域プールないしは，スタック領域から割り当てようとするとき，初期化を行わなければなりません．

ここに示した例で，先の単純な型の実現では規則違反だった実現が，無制約型では正当な実現となっていることが分かります．

区別子を用いて，「単純な」非公開型と「未定」非公開型を折衷することが可能です．この例では，Stack_Typeは， Size と呼ぶ区別子によって型づけられた整数を用いています．

区別子付きレコードの実現を用いることによって，パッケージの非公開領域は， Stack_Type の宣言を完成できることを示しています．



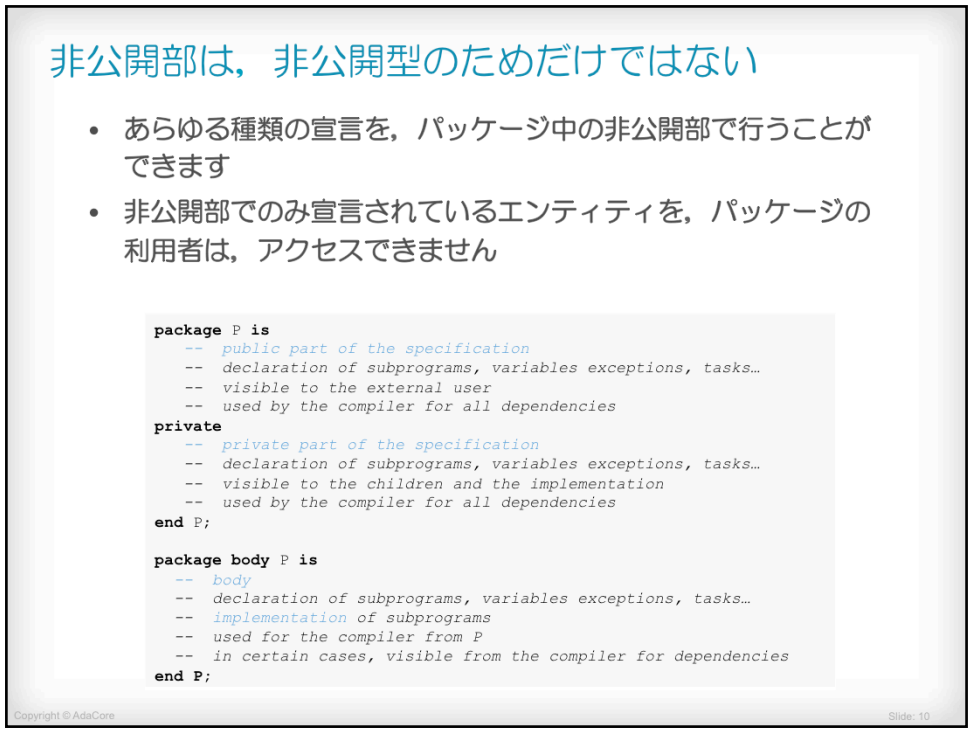
全ての非公開型で遅延非公開定数を宣言できることは，非公開型の強力な特徴の一つです．

遅延として宣言された定数は，公開ビューと非公開ビューを持ちます．公開ビューでは，代入や比較といった暗黙的に定まっている型基本操作（primitive operation）が可能です．これは，全ての非公開型にデフォルト値を与えるために特に有効です．

このスライドの例は，Empty_Stack を意味づける遅延定数を用いた古典的な例です．パッケージ利用者は，比較や初期化が可能です．

遅延定数の実現は，パッケージ利用者から完全に隠されています．ここでは，その実現は，非公開型実現の内部で，集成代入（aggregate assignment）を用いて，実現しています．

Ada 言語でコードを書くとき，使う機会の多いパターンになります．



パッケージ仕様部における非公開部は，非公開型の実現を提供する以外にも用いることができます．

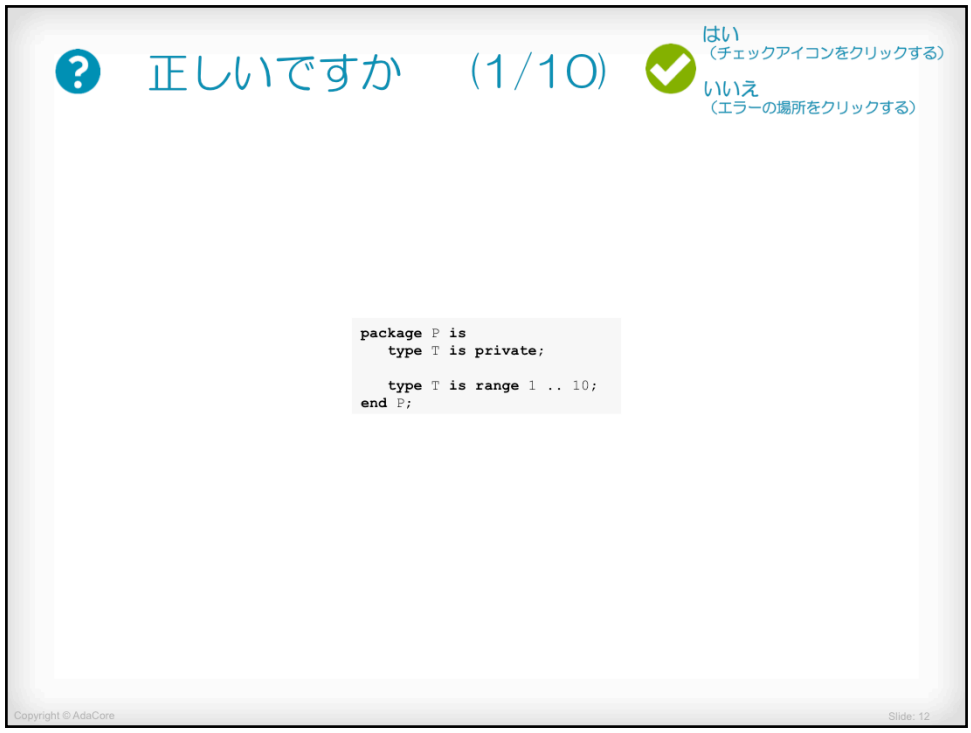
パッケージの利用者は，パッケージ仕様における非公開部のいかなる場所にもアクセスできないことを思い起こして下さい．

まとめます．パッケージ仕様部は，公開部と非公開部に分かれます．公開部は，パッケージ利用者にとって可視であり，非公開部は，パッケージのボディ部と子パッケージにとってのみ可視になります．

ついに，この講義の最後のセクションになりました．

既に，Ada 言語のカプセル化について十分な知識を得たと思います．理解を試すためのちょっとしたクイズをやりましょう．

幸運を！

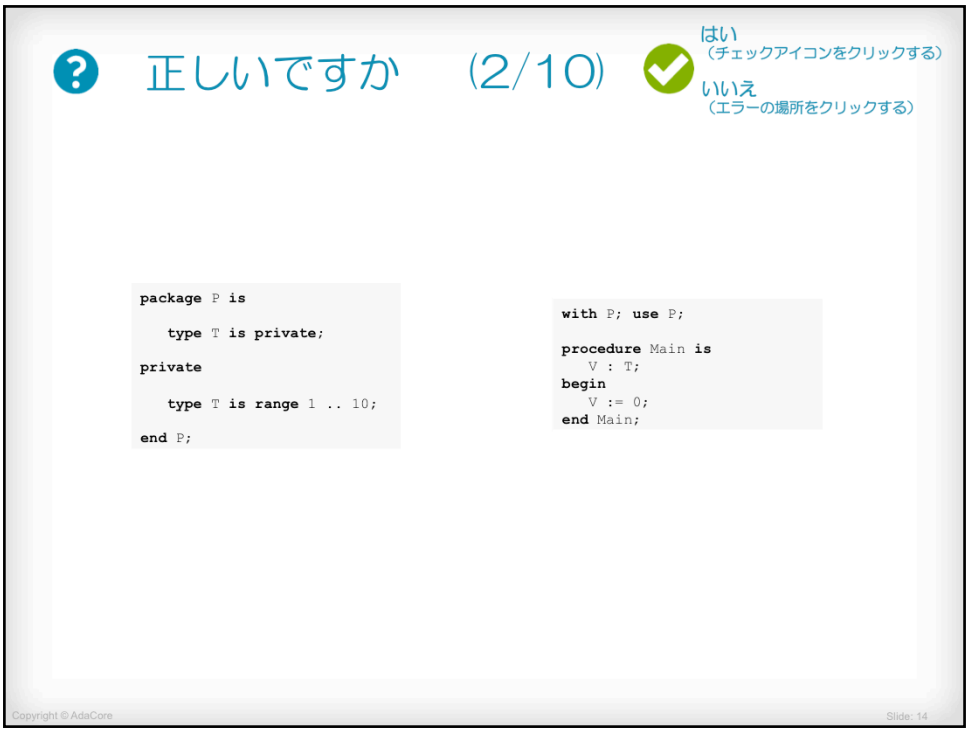


ゆっくりと始めます．単純な例を使います．パッケージ P は，公開部中に，T という名前の非公開型の宣言と，T の実現は 1 から 10 の範囲であると，宣言しています．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．

このコードは正しくありません．非公開型 T の実現は，パッケージ仕様部の非公開領域に現れなければなりません．

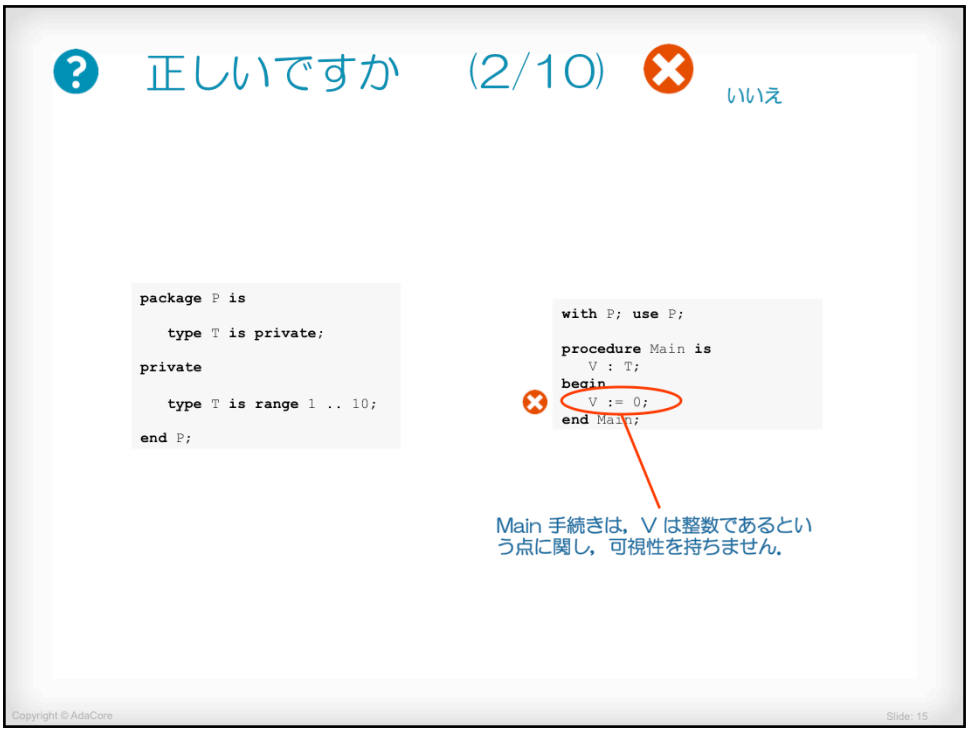
この場合， private キーワードがないため非公開領域がありません．



次の問題では，P のパッケージ仕様部を修正しています．今回，T は，非公開領域で正しく実現されています．

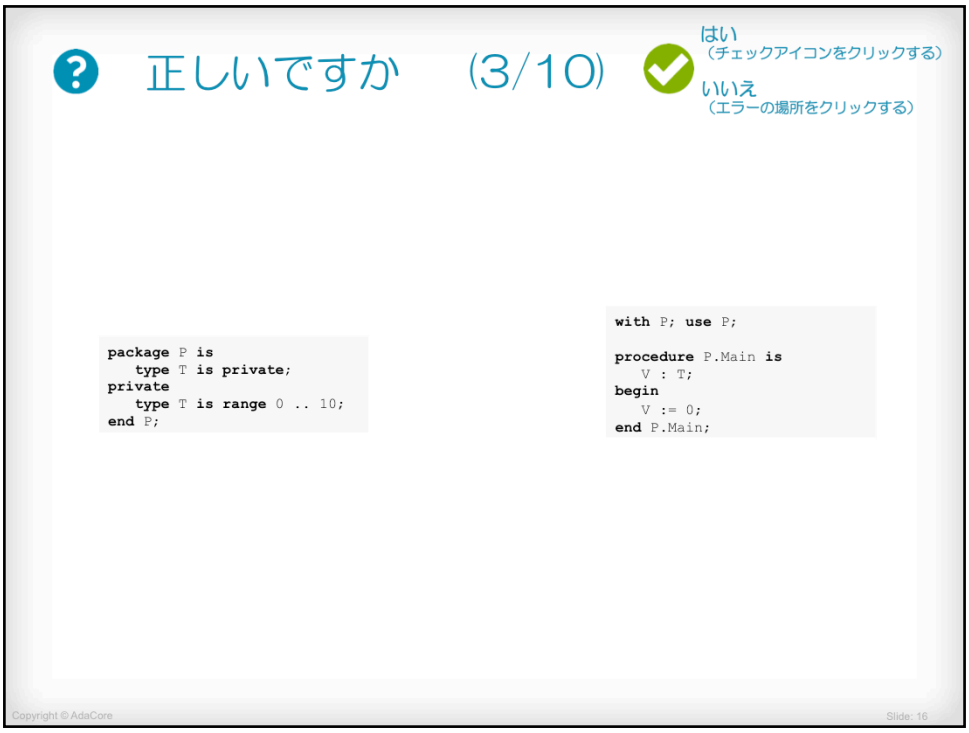
手続き Main は，非公開型 T を用いて，V と云う名前のスタック領域オブジェクトを割り当てています．手続きボディ部で，V に 0 を割り当てています．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



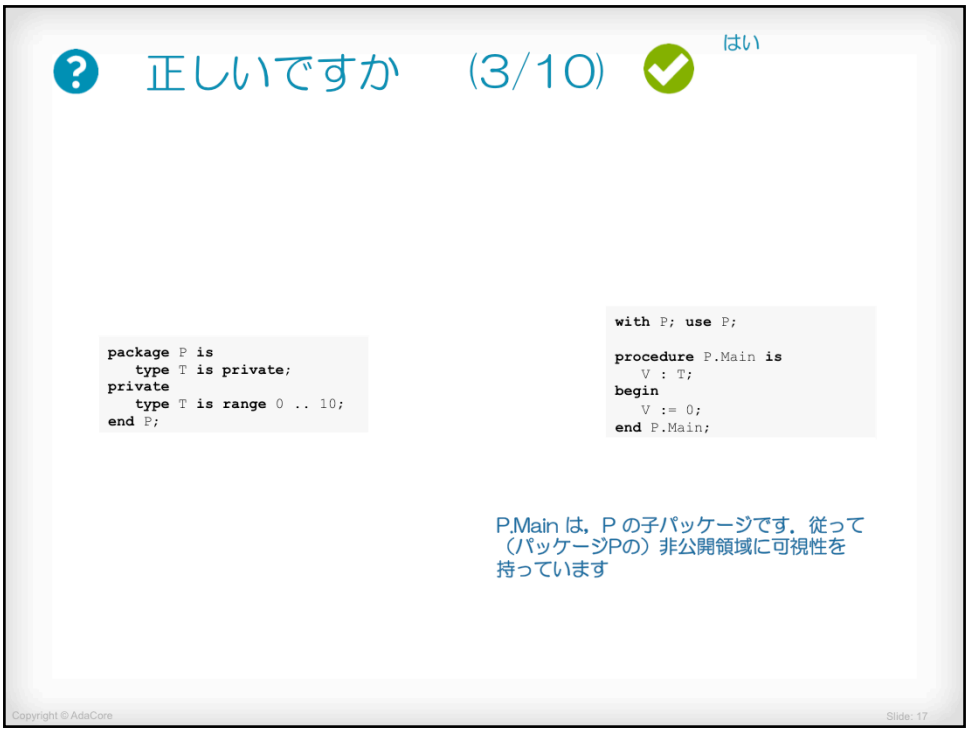
このコードにも間違いがあります．手続き Main は，非公開型 T の実現が整数であるという仮定をしています．

それは間違いです．P の利用者は，P の非公開型の実現に関して，可視性を持ちません．



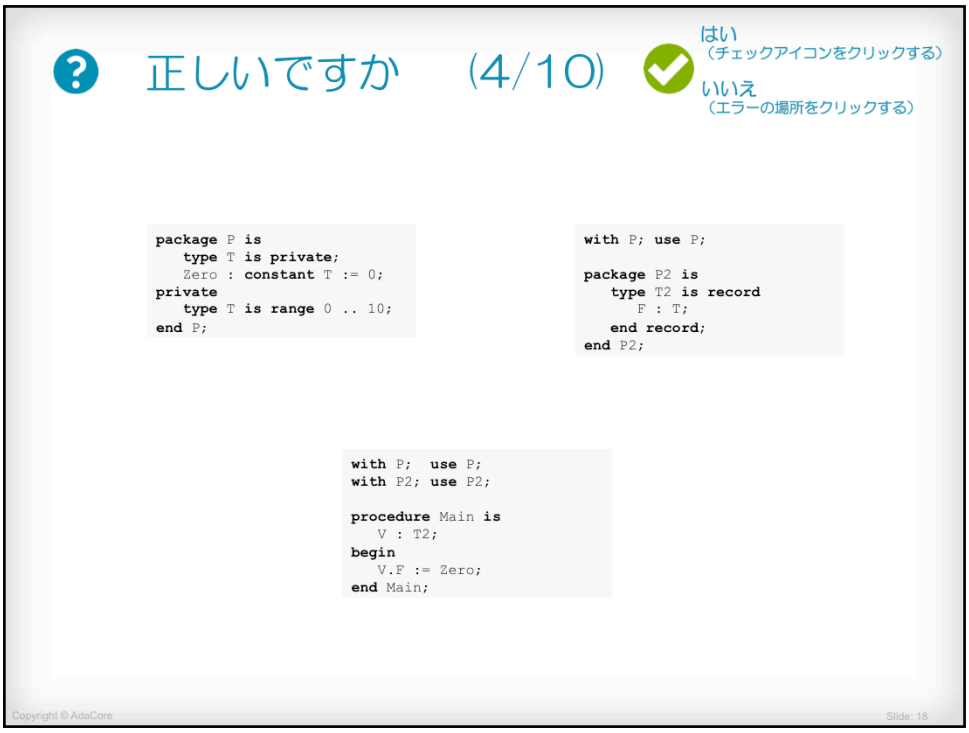
問題 3 では，先の問題を少し変更しています．ここでは，手続き Main は，パッケージ P の子パッケージとして宣言されています．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



これは正しいコードです．手続き Main は，パッケージ P の子パッケージであり，P の非公開領域に対して完全な可視性を持っています．

非公開領域を持っているパッケージのボディ部と子パッケージは，その非公開型の実現に関して可視性を持っています．

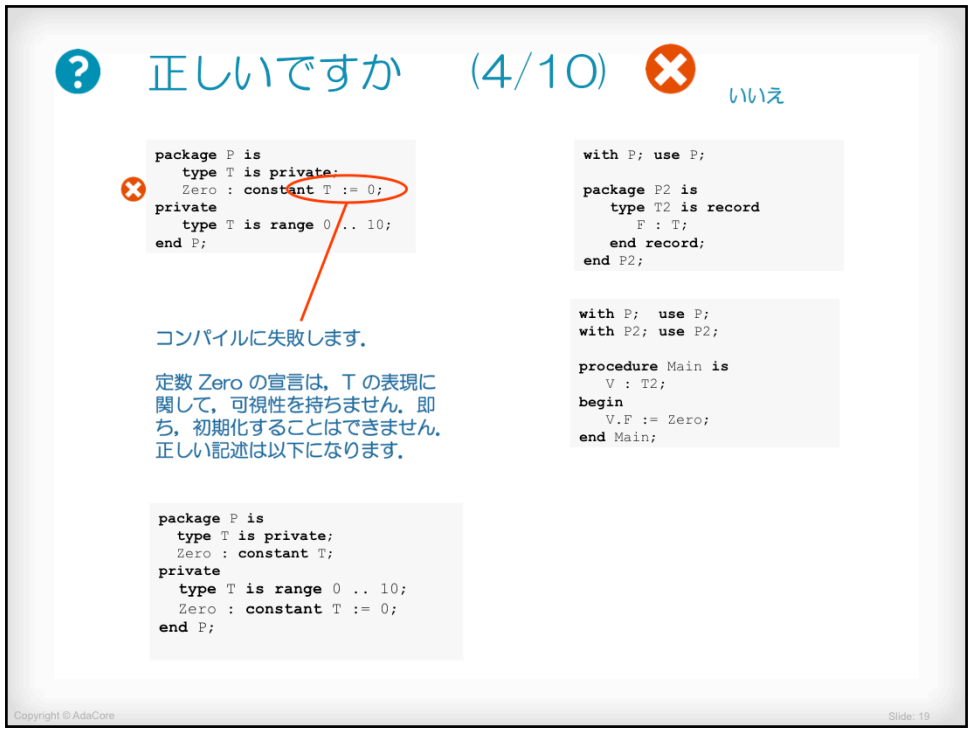


次の問題は，パッケージ P に関してです．P は，非公開型 T と，型 T の Zero という名前の非公開定数を持ち，ゼロに初期化しています．

非公開領域には，T の実現があり，パッケージ P2 の仕様部では，T2 という型のレコード型宣言内部で，この T を使っています．

手続き Main は，パッケージ P とパッケージ P2 の両方にアクセスします．また，パッケージ P2 の T2 を，スタック領域オブジェクト V として実体化します．更に，代入文において，パッケージ P の非公開定数 Zero を利用します．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



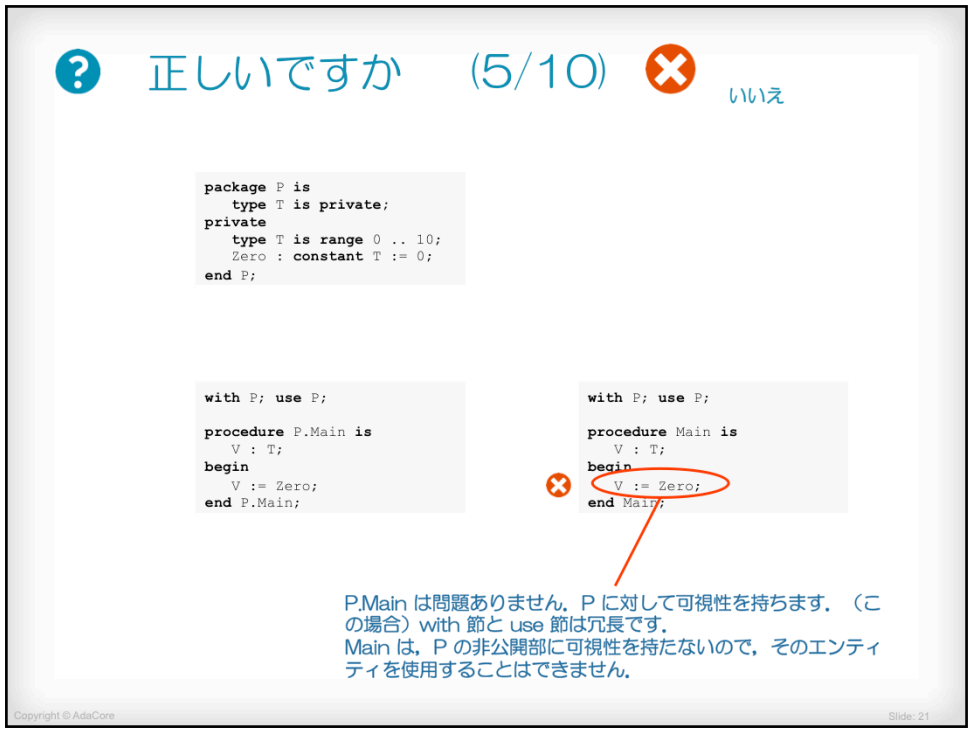
このコードは，あいにくコンパイルに失敗します．パッケージ P 内部の非公開定数 Zero は，パッケージ仕様部の非公開領域内部で実現前に設定されています．

代わりに何を行うべきでしょうか．定数　Zero　は，遅延定数です．公開部での宣言と非公開部での実現とすべきです．

更に，例題を拡張します．手続き P.Main と呼ぶパッケージ P の子パッケージ，および， Main という名前の独立した手続きがあります．

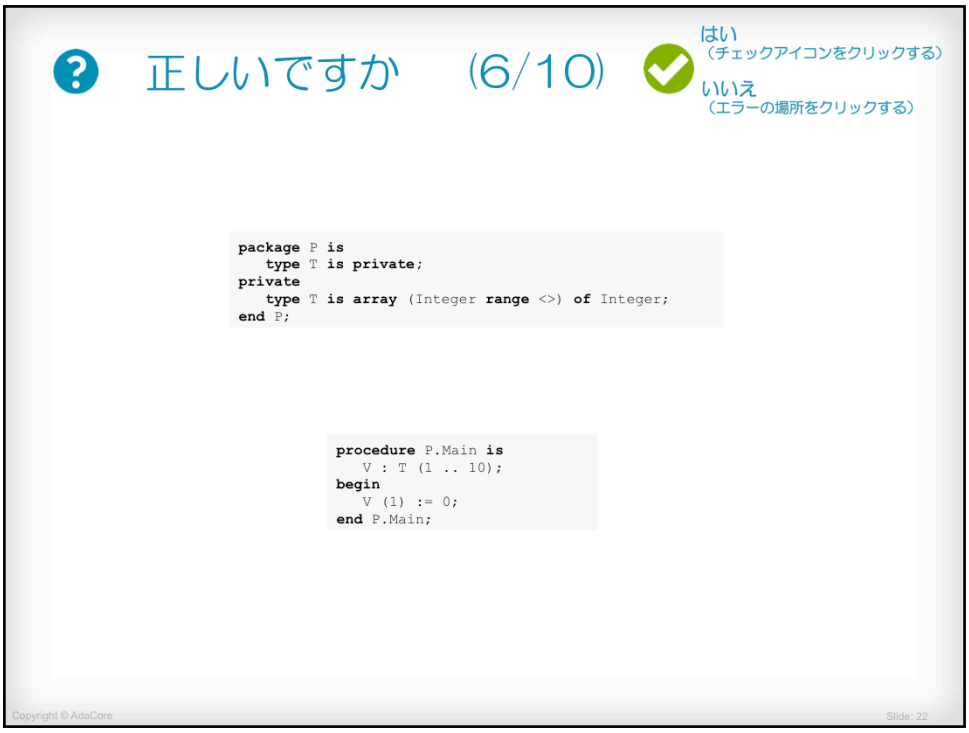
両者とも独立した実現です．両方ともパッケージ P 中の非公開定数 Zero を，非公開型 P.T の局所的に実体化したオブジェクトに代入しています．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



このコードは誤りです．

手続き Main は，パッケージ P の非公開領域に可視性を持ちません．従って，非公開エンティティを使用することはできません．この場合は，定数 Zero が非公開エンティティに相当します．

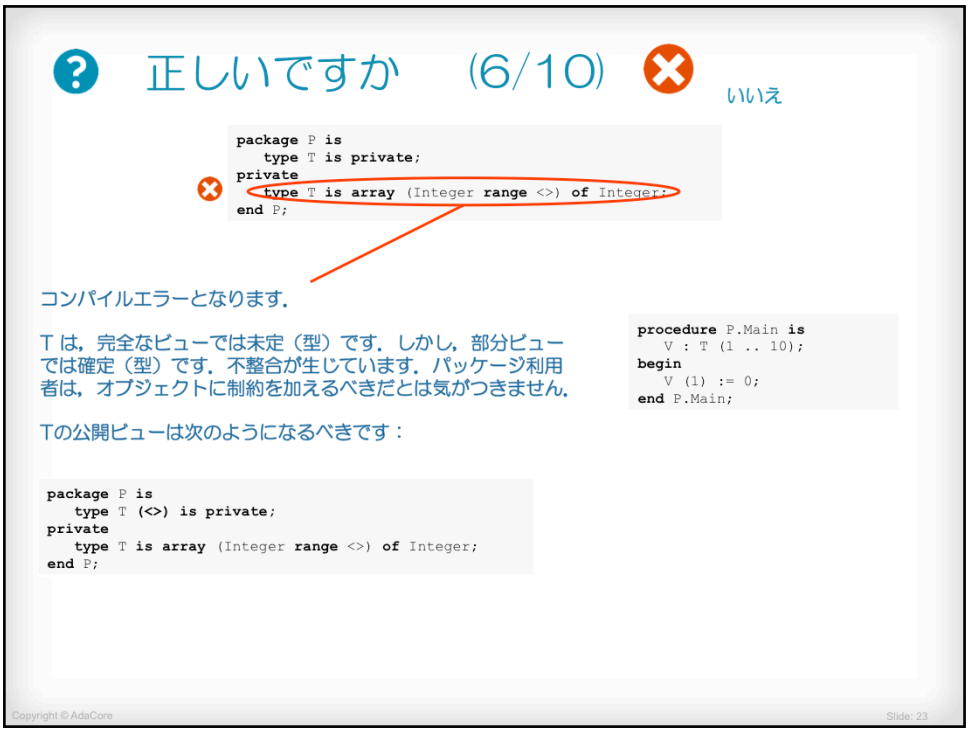


問題 6 では，最初にパッケージ P を示しています．このパッケージは，非公開型 T を持ち，非公開領域において，整数の無制約配列を用いて実現しています．

手続き Main は，P の子パッケージであり，スタック領域オブジェクト V を宣言しています．この V は，型が T であり，1 から 10 の添字制約を持っています．

この手続き本体で，V の最初の要素にゼロを代入しています．

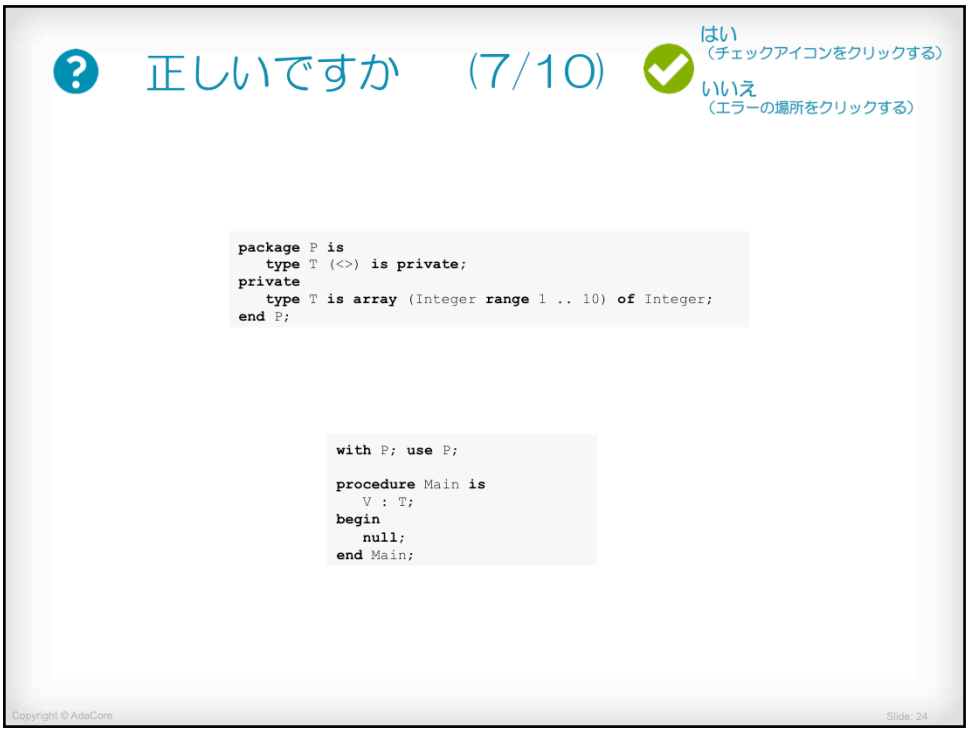
このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



この例は，コンパイルエラーとなります．T は，単純非公開型として宣言されていますが，未定型として実現しています．

パッケージの利用者は，オブジェクトに制約を与えないといけないとは分からないので，不整合が生じます．

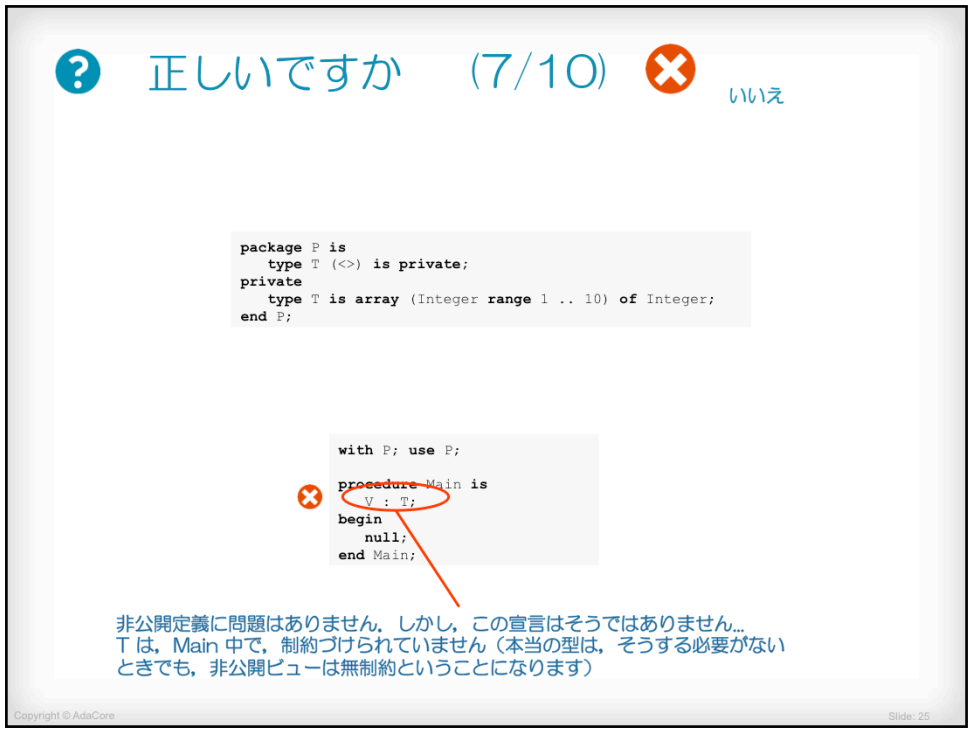
正しいパッケージ仕様を，スライドの最後に示します．



この問題では，パッケージ P と非公開未定型を宣言しています．また，実現においては，整数配列の添字に対して，1 から 10 の範囲を持つ整数の副型を定義しています．

手続き Main は，非公開型 T のスタック領域オブジェクトを実体化し，ヌルボディ部を持ちます．

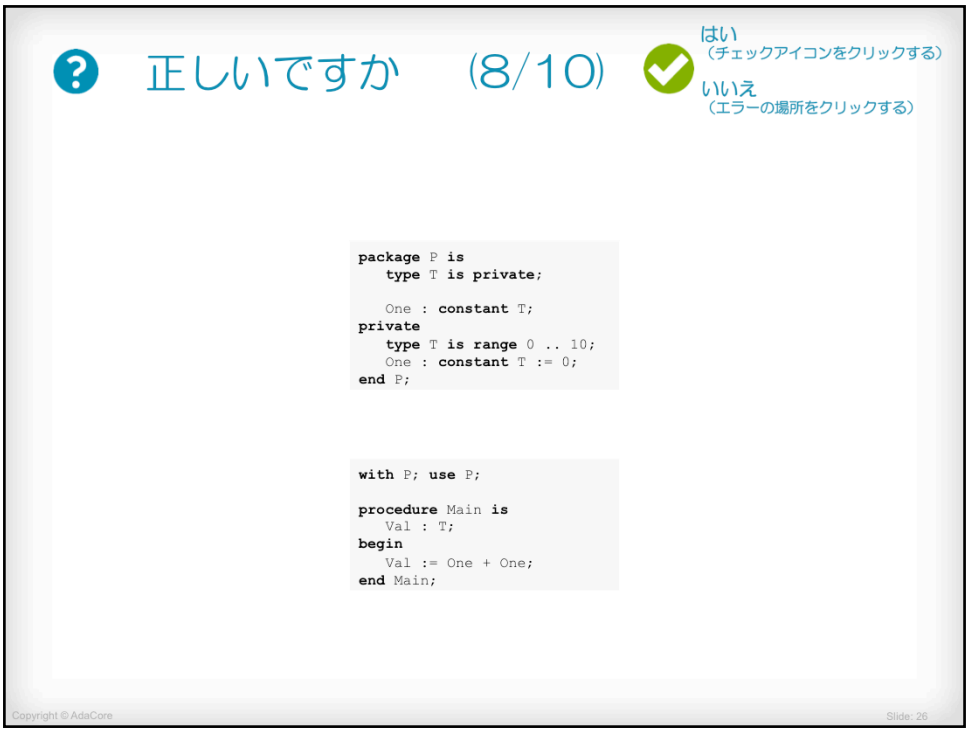
このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



非公開型宣言とその実現を持つ P のパッケージ仕様に問題はありません．

問題は，オブジェクト V を宣言するときの，手続き Main における非公開型の扱いにあります．

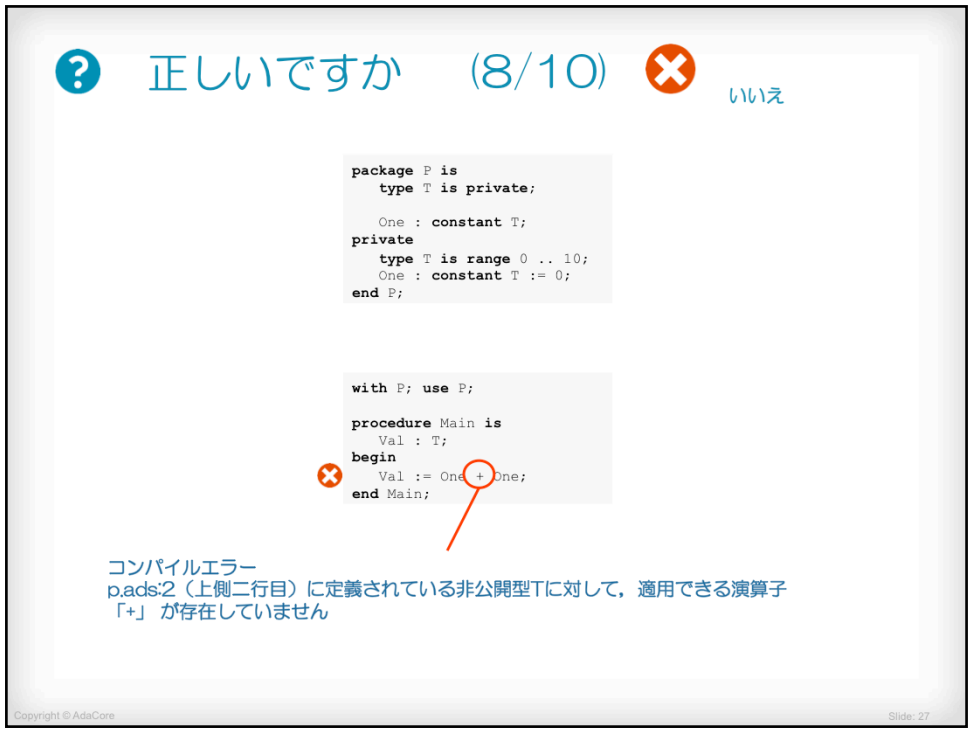
未定非公開型を利用するときには，全ての利用者（ここでは手続き Main です）は，利用時に，制約をつけなければなりません．



問題 8 では，最初にパッケージ P の仕様部があり，単純な非公開型と非公開定数を宣言しています．T の実現は，0 から 10 までの数字で，T 型の定数 Oneは，0 に初期化されています．

手続き Main では，非公開型 T に対してスタック領域オブジェクト Val を実体化しています．ボディ部で，One と One の加算結果を Val に代入しています．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



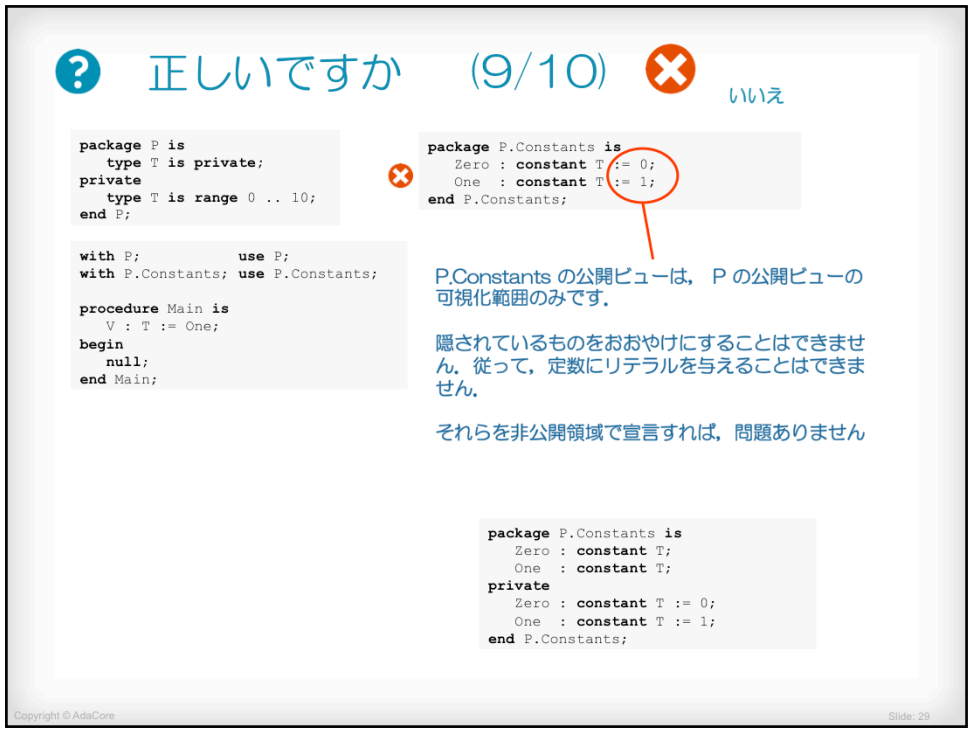
このコードは間違いです．あいにく，コンパイルエラーとなります．"+" 演算子は，P 内部で定義されている非公開型 T で宣言されていないからです．

終わりから二番目の問題は，P のパッケージ仕様を最初に扱います．ここでは，単純な非公開型 T と，非公開部内で T は 0 から 10 までの数字であるという実現を持ちます．

Constants という名前の P の子パッケージは，公開部において，親パッケージ P の非公開型 T の定数を持ちます．

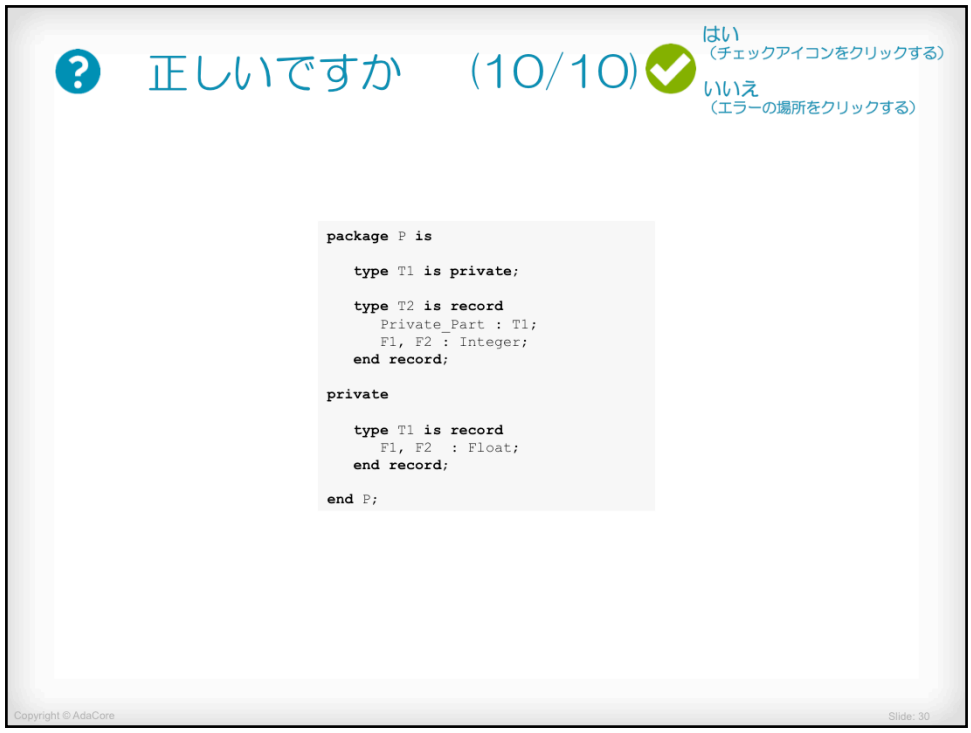
手続き Main は，非公開型 T のスタック領域オブジェクト V を実体化しています．更に，子パッケージ Constants から定数の一つを使って，この V を初期化しています．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



子パッケージ P.Constants 内で，コンパイルエラーが生じます．親パッケージの非公開領域にある情報を公開部で使用しているからです．

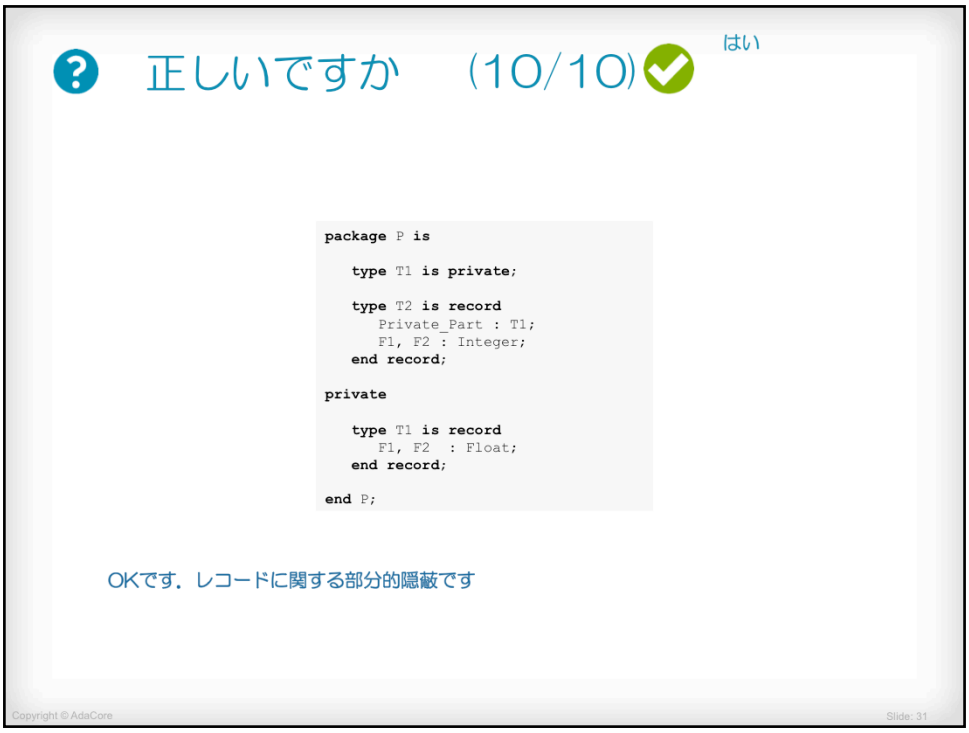
解決策の一部をスライドの下部に示しています．2 つの遅延定数を用い，非公開領域で，実現・初期化しています．



最後のクイズです．パッケージ P があります．ここでは，単純な非公開型 T1 があり，公開レコード型定義 T2 の要素として，この T1 を用いています．

非公開領域には，単純な非公開型 T1 の実現があります．

このコードが正しいと思うときは，「はい」のアイコンをクリックしてください．間違いと思えば，その場所をクリックしてください．



良い結果で終わります．このコードは，完全に正しく，部分を非公開にした公開レコード型として，再利用可能なパターンを提供しています．

Ada 大規模プログラムのカプセル化のコースに参加いただきありがとうございました.

Ada プログラム言語を学ぶ上で，この講義が貴重なーステップとなったこと，Ada University における他のコースも引き続き受けられることを期待しています.

ありがとうございました.