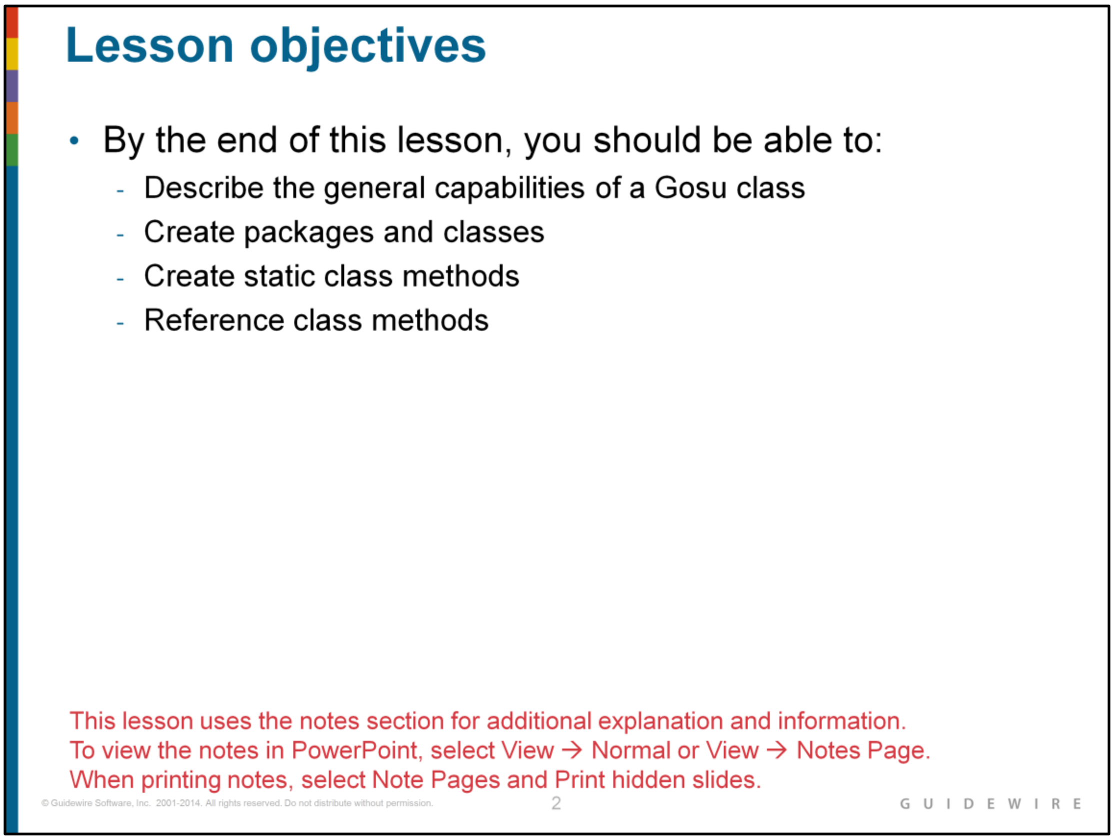
Gosu Classes

June 10, 2014

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.

Guidewire training materials contain Guidewire proprietary information that is subject to confidentiality and non-disclosure agreements. You agree to use the information in this manual solely for the purpose of training to implement Guidewire software solutions. You also agree not to disclose the information in this manual to third parties or copy this manual without prior written consent from Guidewire. Guidewire training may be given only by Guidewire employees or certified Guidewire partners under the appropriate agreement with Guidewire.



Lesson objectives

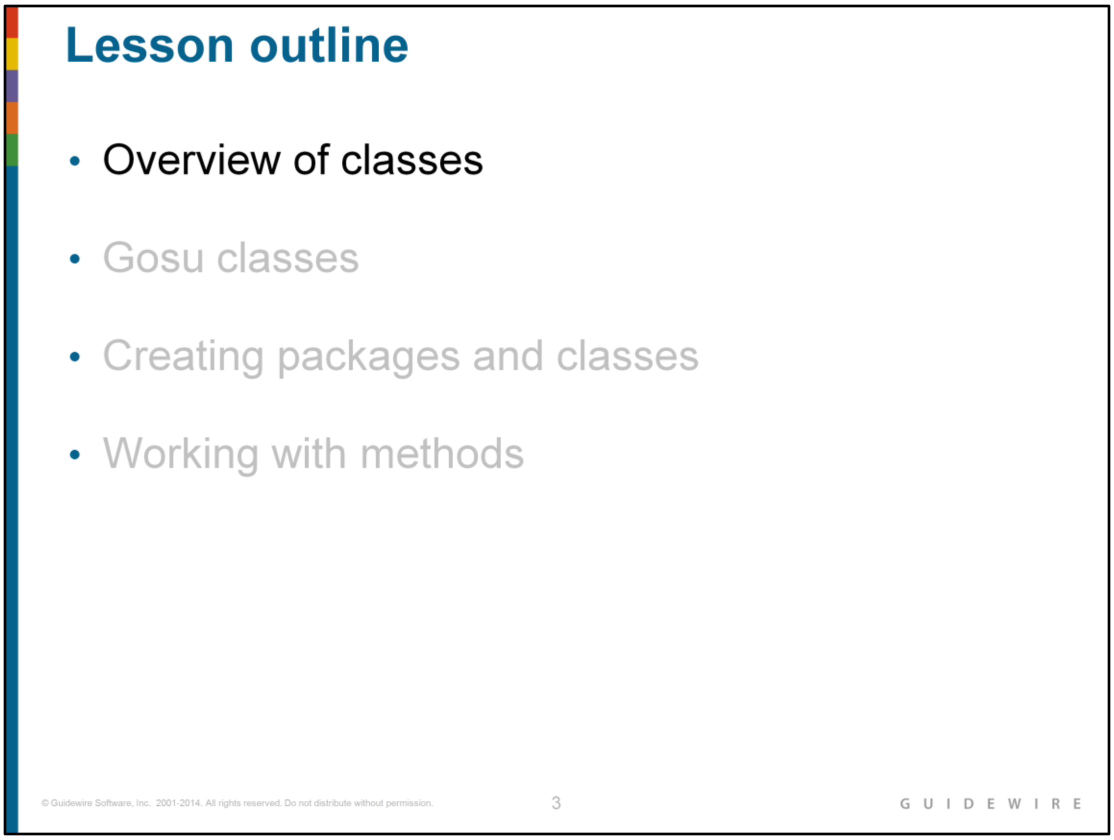
- By the end of this lesson, you should be able to:
  - Describe the general capabilities of a Gosu class
  - Create packages and classes
  - Create static class methods
  - Reference class methods

This lesson uses the notes section for additional explanation and information.
To view the notes in PowerPoint, select View → Normal or View → Notes Page.
When printing notes, select Note Pages and Print hidden slides.

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



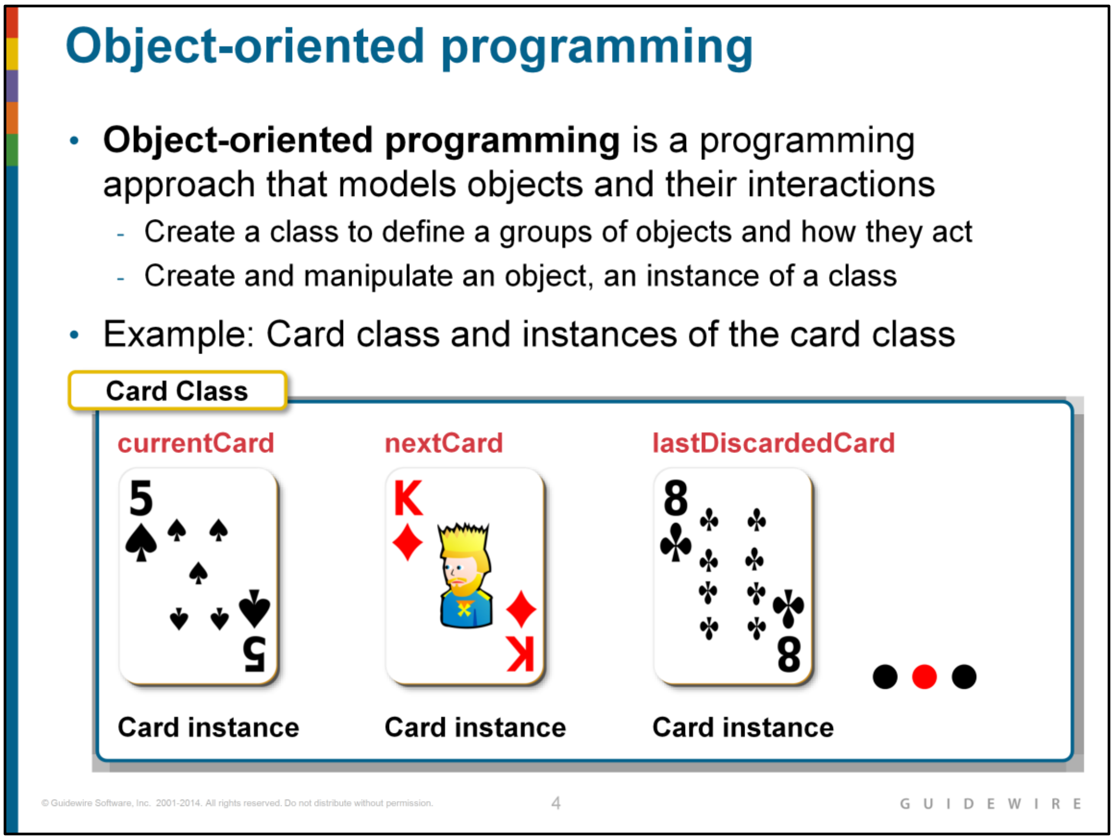
Lesson outline

- Overview of classes
- Gosu classes
- Creating packages and classes
- Working with methods

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Object-oriented programming

- **Object-oriented programming** is a programming approach that models objects and their interactions
  - Create a class to define a groups of objects and how they act
  - Create and manipulate an object, an instance of a class
- Example: Card class and instances of the card class

Card Class
currentCard
nextCard
lastDiscardedCard
Card instance
Card instance
Card instance

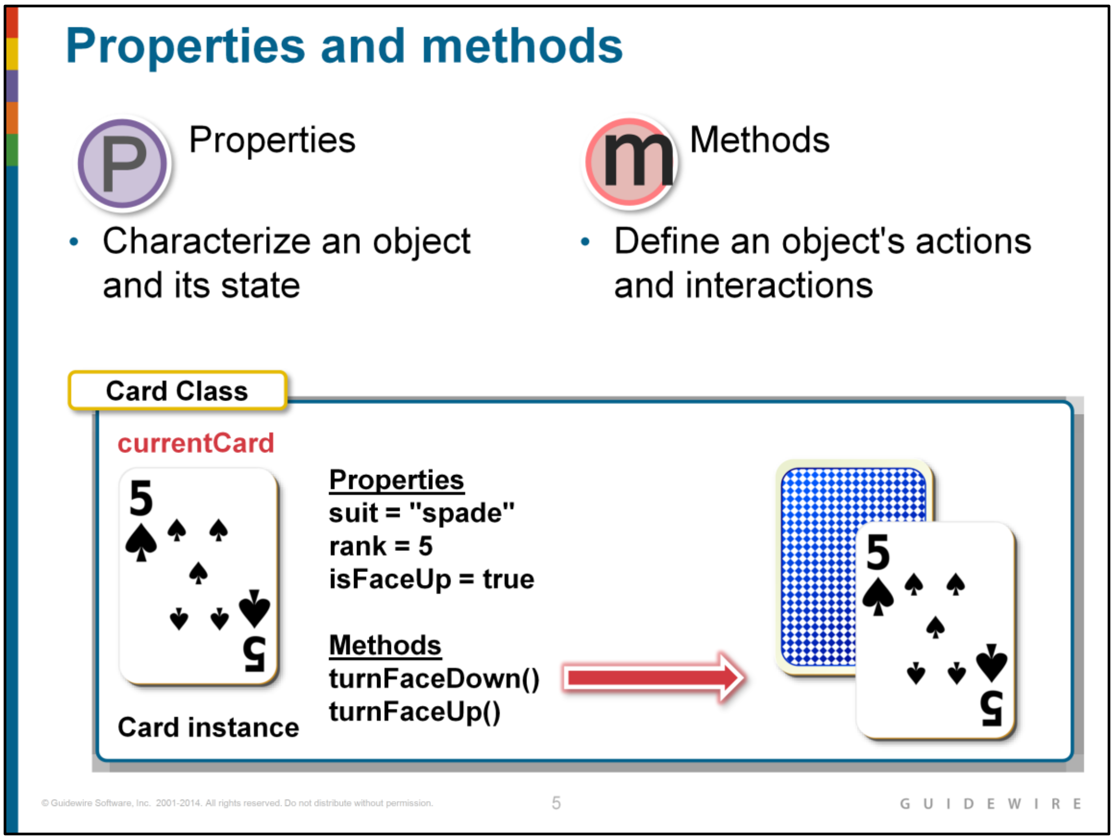
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 4 GUIDEWIRE

Imagine a playing card software program and a class known as Card. In the slide example, there are three specific instances of the Card class: currentCard, nextCard, and lastDiscardedCard.

Object-oriented programming is a programming approach in which developers create classes, which are instructions for how to build a set of objects and the properties and behaviors those objects will have, and create and manipulate individual objects, known as instances of a class or type. Every class is created by a class definition, which specifies the class name, the properties that every instance of the class has, and the methods that every instance of the class has.

Object-oriented programming consists of four principles: Encapsulation, Abstraction, Polymorphism and Inheritence. Encapsulation is the hiding of a data implementation by restricting access to accessors and mutators. An accessor is a method that is used to ask an object about itself. It can be a property or a public method. Mutators are public methods that modify the state of an object, while hiding the implementation of exactly how the data gets modified. Abstraction denotes the essential characteristics of an object that distinguish it from all other kinds of object and thus provide crisply defined conceptual boundaries, relative to the perspective of the viewer. Inheritance allows you to inherit functionality from another class, called a superclass or base class. Inheritance is the ability to extend components without any knowledge of the way in which a class was implemented. Polymorphism means one name, many forms. Polymorphism manifests itself by having multiple methods all with the same name, but slighty different functionality.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



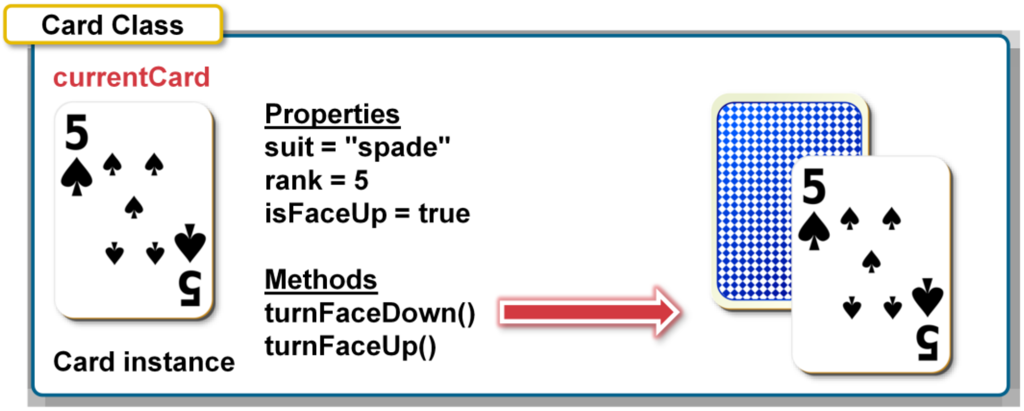
Properties and methods

Properties

- Characterize an object and its state

Methods

- Define an object's actions and interactions



Properties
suit = "spade"
rank = 5
isFaceUp = true

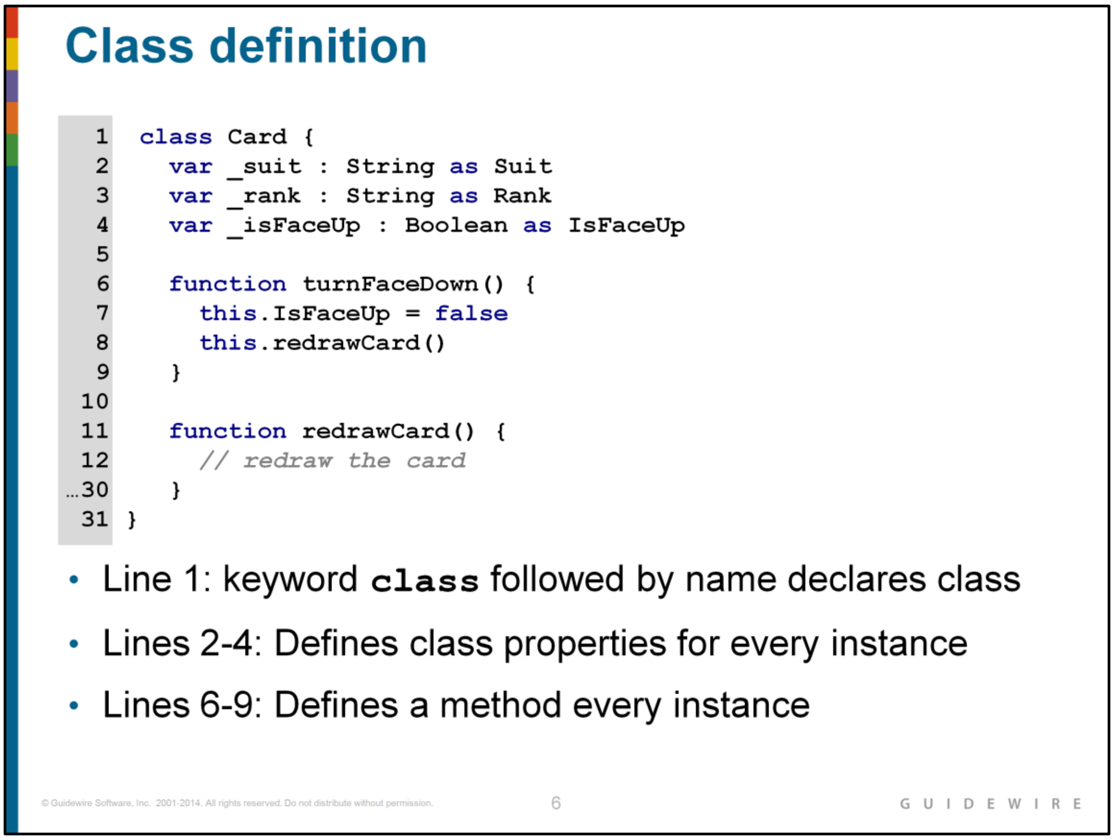
Methods
turnFaceDown()
turnFaceUp()

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

All instances of the same class have the same properties and methods. The only time you would encounter an instance without properties (or methods) is if the instance's class definition did not specify any properties (or methods) for instances of that class.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Class definition

```
 1  class Card {
 2    var _suit : String as Suit
 3    var _rank : String as Rank
 4    var _isFaceUp : Boolean as IsFaceUp
 5
 6    function turnFaceDown() {
 7      this.IsFaceUp = false
 8      this.redrawCard()
 9    }
10
11    function redrawCard() {
12      // redraw the card
…30    }
31  }
```

- Line 1: keyword **`class`** followed by name declares class
- Lines 2-4: Defines class properties for every instance
- Lines 6-9: Defines a method every instance

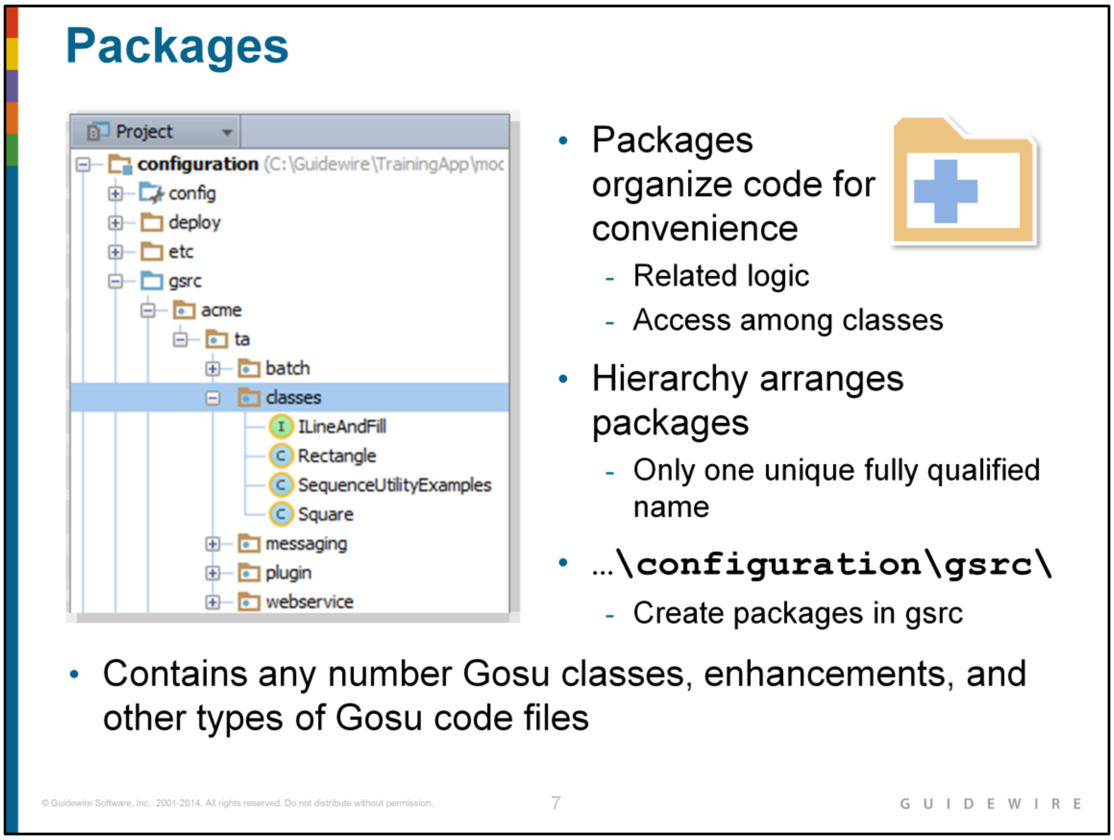
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 6 GUIDEWIRE

In a Card class used for an actual program, the suit and rank properties would probably have constrained data types. For example, suit would be limited specifically to hearts, diamonds, clubs and spades. In the slide example, suit is a property of the type String, but could be other data types.

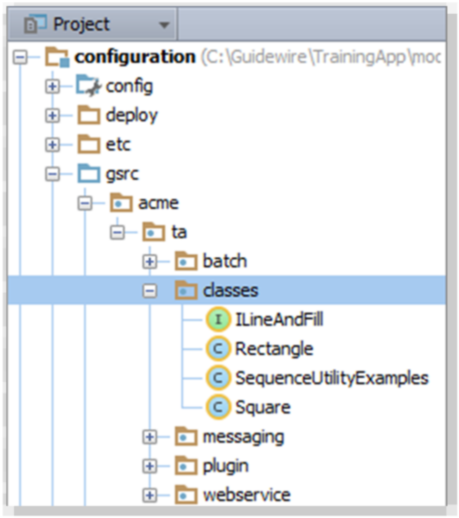
There is a variety of object-oriented programming languages, each with variations in syntax. In the slide example, the syntax is an example using Gosu, Guidewire's programming language.

Gosu supports object-oriented programming using classes, interfaces and polymorphism. Also, Gosu is fully compatible with Java types, so Gosu types can extend Java types, or implement Java interfaces.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Packages



- Packages organize code for convenience
  - Related logic
  - Access among classes
- Hierarchy arranges packages
  - Only one unique fully qualified name
- …`\configuration\gsrc\`
  - Create packages in gsrc
- Contains any number Gosu classes, enhancements, and other types of Gosu code files

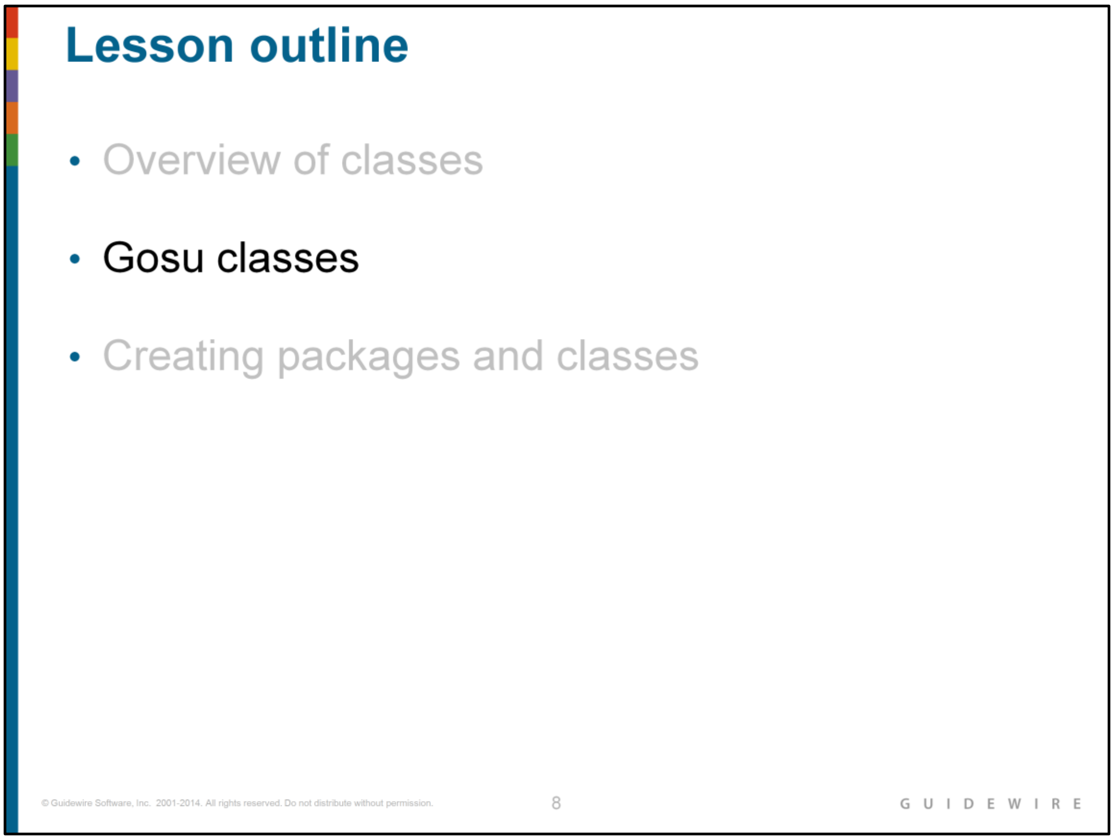
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 7 GUIDEWIRE

A package is an aspect of object oriented programming languages. It is a collection of classes gathered together for convenience often because the classes have a similar purpose such as they perform related logic and/or
requires access to each other (while classes outside package do not need access).

You reference a class in a package with a fully qualified name. A fully qualified name details the package hierarchy. In the slide example, the Square class exists in the hierarchy of the acme package. For example, the fully qualified name for the Square class is acme.ta.classes.Square.

Per package hierarchy, you can only have one unique class name.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



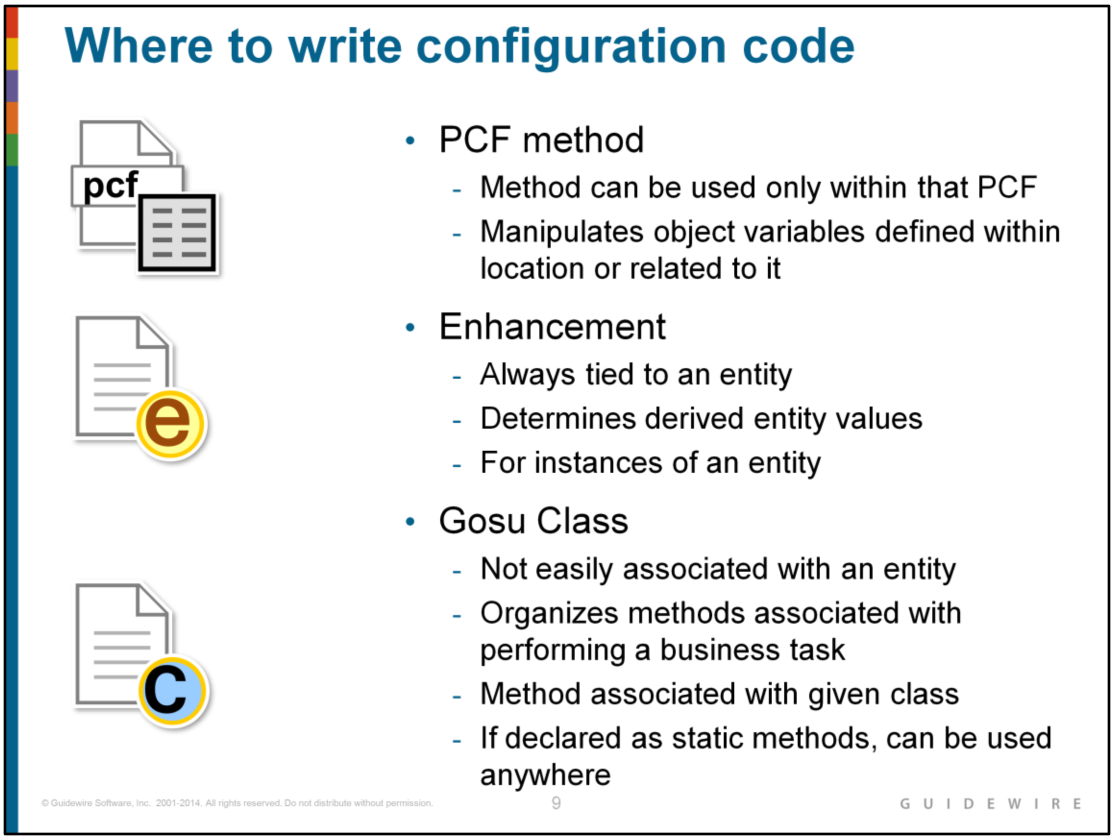
Lesson outline

- Overview of classes
- Gosu classes
- Creating packages and classes

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



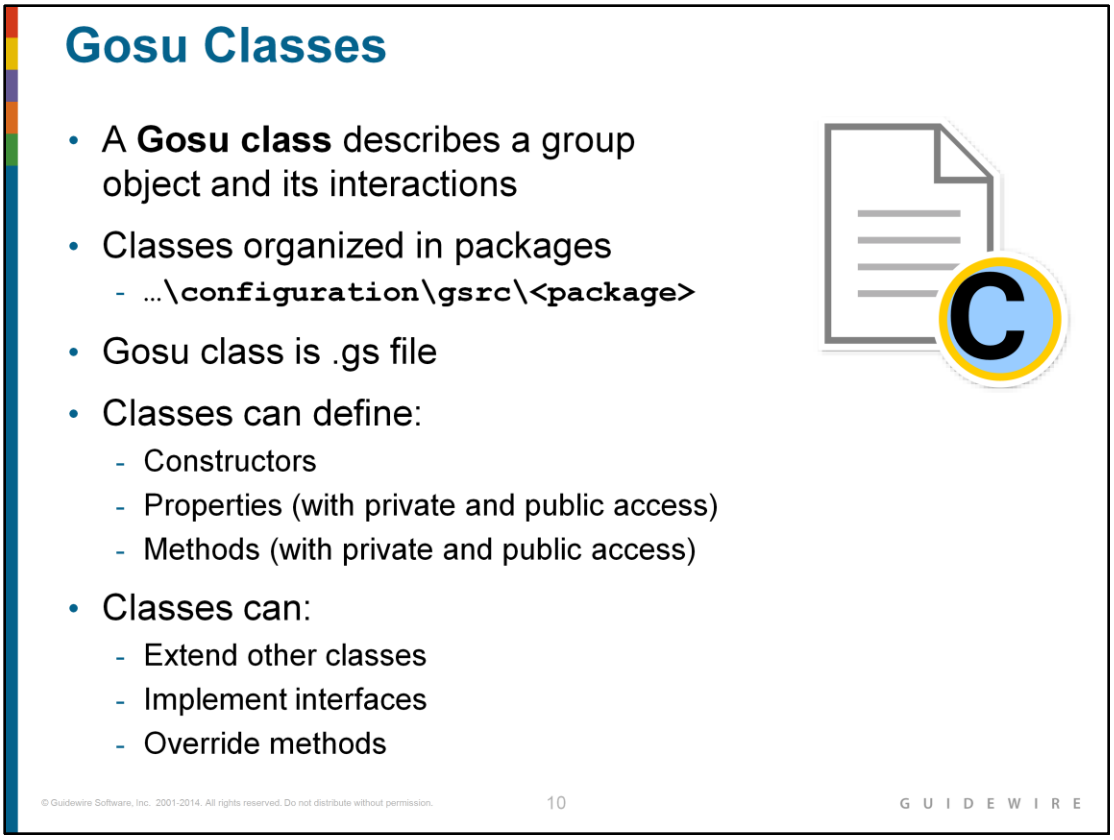
Where to write configuration code

- PCF method
  - Method can be used only within that PCF
  - Manipulates object variables defined within location or related to it
- Enhancement
  - Always tied to an entity
  - Determines derived entity values
  - For instances of an entity
- Gosu Class
  - Not easily associated with an entity
  - Organizes methods associated with performing a business task
  - Method associated with given class
  - If declared as static methods, can be used anywhere

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.
9
GUIDEWIRE

Class methods can also be non-static. In this case, the method can only be called from an instance of the class. Often configuration developers do not write an extensive amount of non-static class methods. Integration developers, however, typically do write both static and non-static class methods.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Gosu Classes

- A **Gosu class** describes a group object and its interactions
- Classes organized in packages
  - …**\configuration\gsrc\<package>**
- Gosu class is .gs file
- Classes can define:
  - Constructors
  - Properties (with private and public access)
  - Methods (with private and public access)
- Classes can:
  - Extend other classes
  - Implement interfaces
  - Override methods

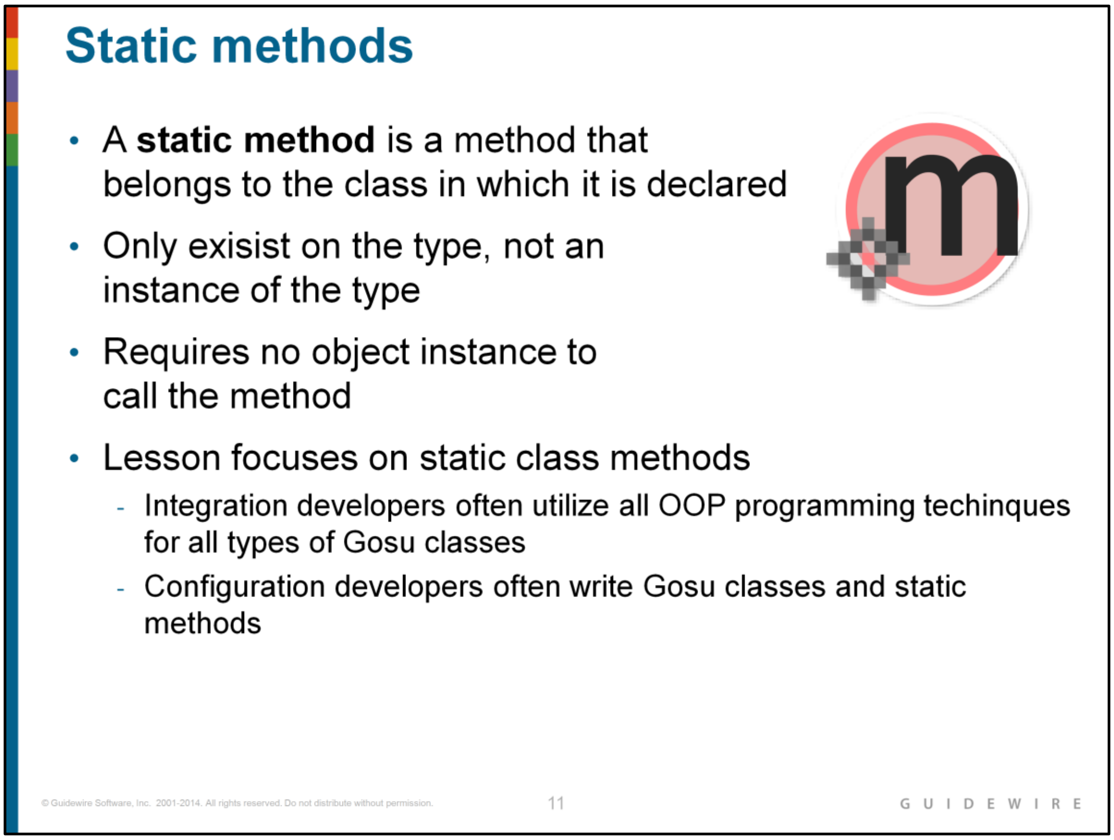
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 10 GUIDEWIRE

Gosu classes have a package structure which contains the individual classes and the classes are extendable. Using Gosu, you can write your own custom classes and call these classes from within Gosu. You create and reference Gosu classes by name.

For example, suppose that you want to define a class called MyClass in package MyPackage with a method called getName(). You would first create the class in the …\configuration\gsrc\ folder in Guidewire Studio. Then, in Gosu Scrathpad, you can instantiate the class as an object and call the method. For example:

```
var myObject = new MyPackage.MyClass()
var name = myObject.getName()
```

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Static methods

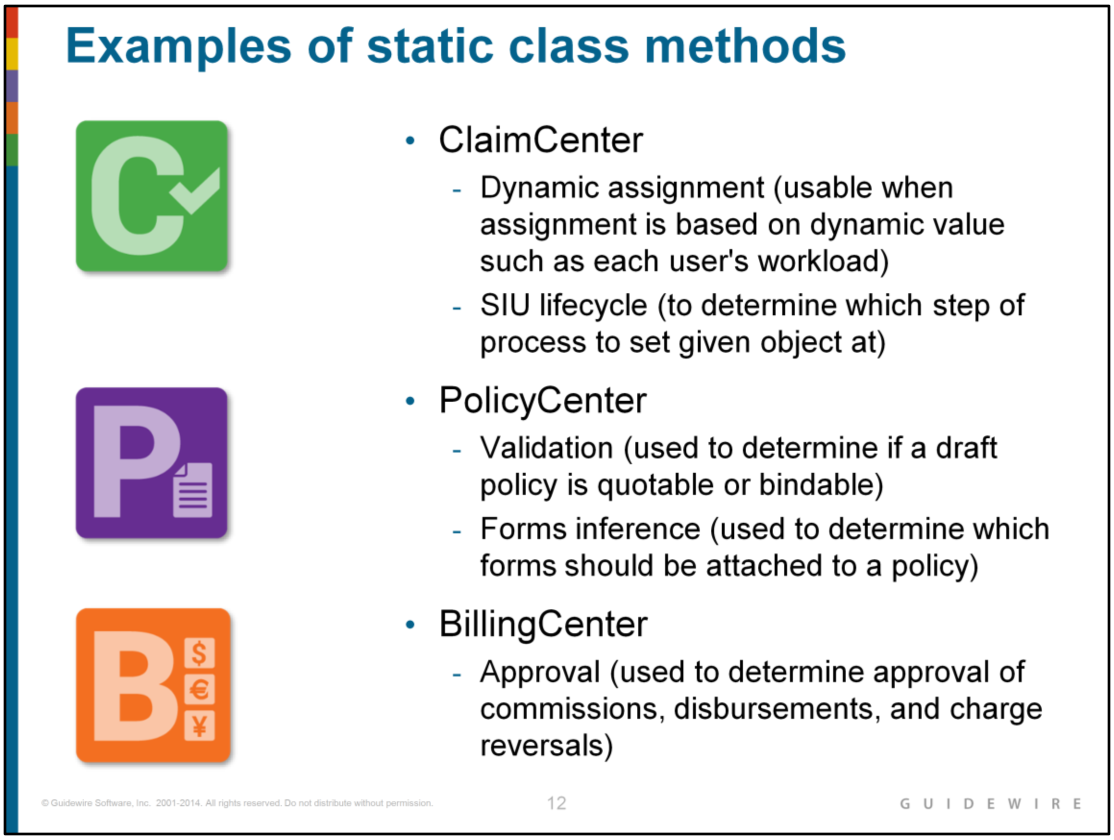
- A **static method** is a method that belongs to the class in which it is declared
- Only exisist on the type, not an instance of the type
- Requires no object instance to call the method
- Lesson focuses on static class methods
  - Integration developers often utilize all OOP programming techinques for all types of Gosu classes
  - Configuration developers often write Gosu classes and static methods

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

Most integration work involves full utilization of OOP capabilities, including:
- Extending existing classes
- Implementing interfaces
- Creating instances of classes
- Overriding methods
- Using try/catch blocks to handle exceptions

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Examples of static class methods

- ClaimCenter
  - Dynamic assignment (usable when assignment is based on dynamic value such as each user's workload)
  - SIU lifecycle (to determine which step of process to set given object at)
- PolicyCenter
  - Validation (used to determine if a draft policy is quotable or bindable)
  - Forms inference (used to determine which forms should be attached to a policy)
- BillingCenter
  - Approval (used to determine approval of commissions, disbursements, and charge reversals)

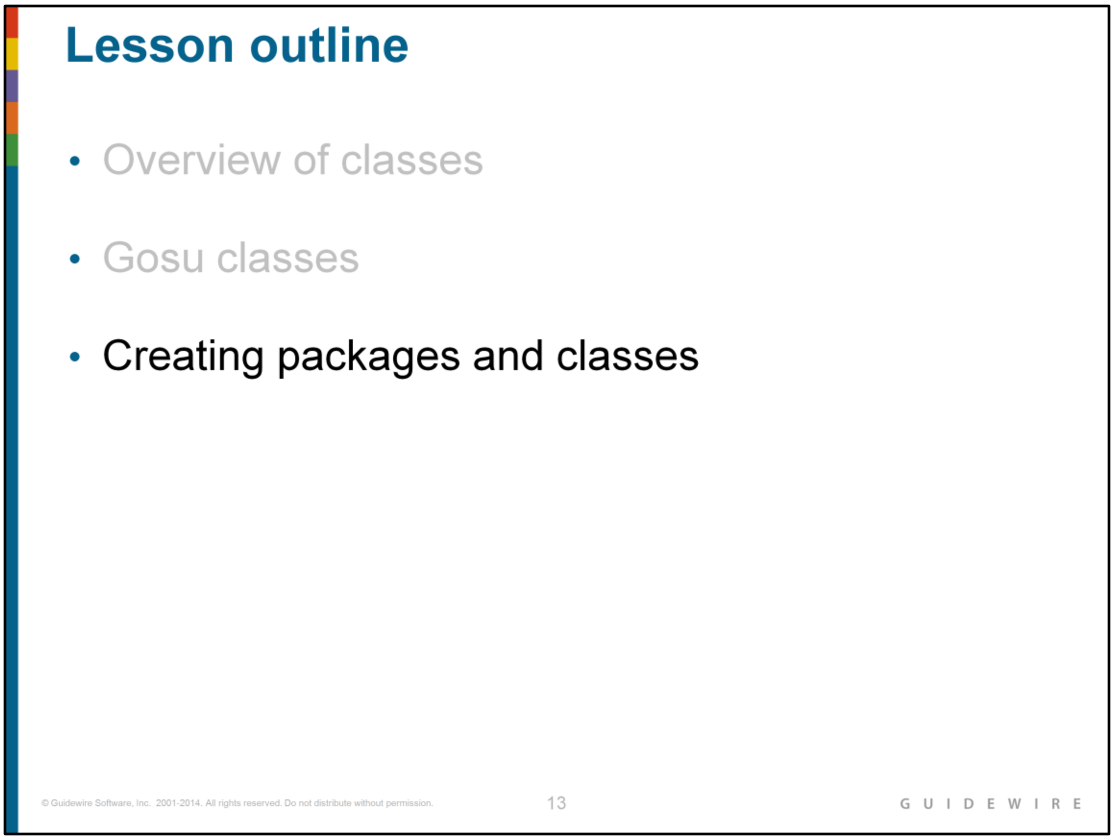
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

SIU stands for Special Investigations Unit, which is the part of a claims processing organization that evaluates potentially fraudulent claims. The examples can be found in the following packages for the given product's package hierarchy:

- ClaimCenter dynamic assignment: gw.api.assignment.examples package
- ClaimCenter SIU lifecycle: libraries package
- PolicyCenter validation: gw.validation package
- PolicyCenter forms inference: gw.lob.<specific line of business>.forms packages
- BillingCenter approval: gw.approval package

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



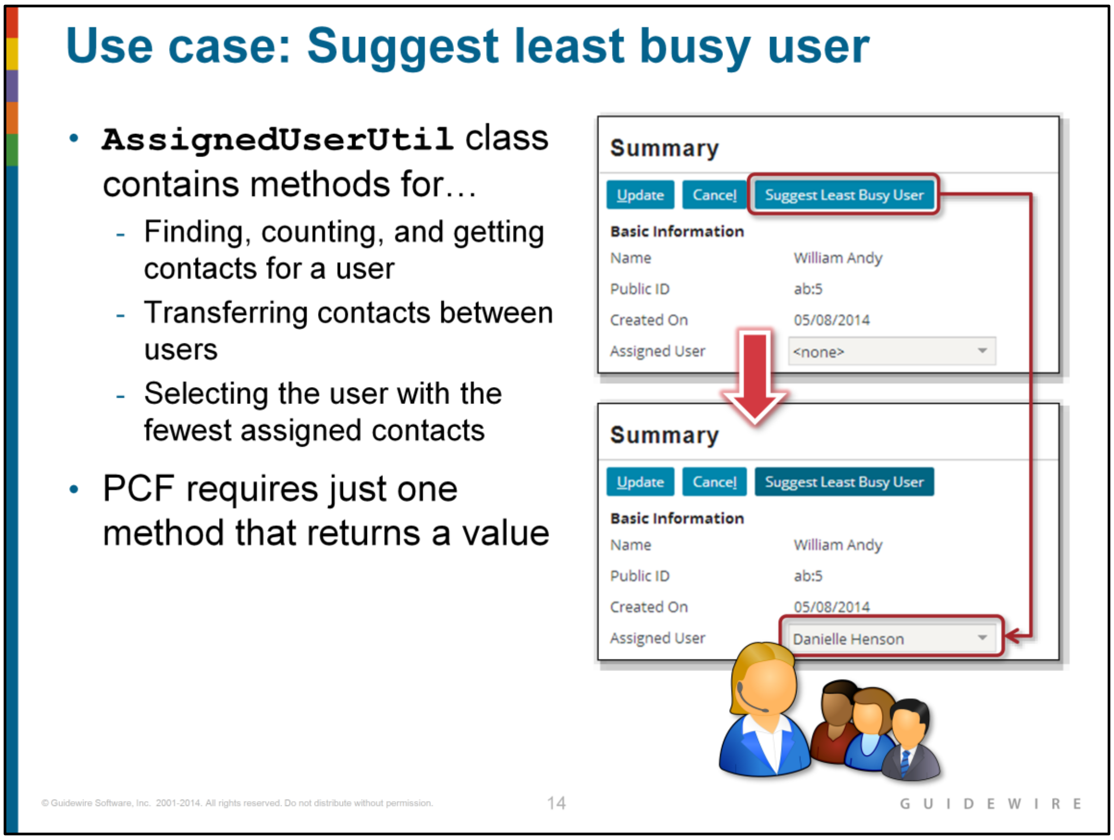
Lesson outline

- Overview of classes
- Gosu classes
- Creating packages and classes

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Use case: Suggest least busy user

- **`AssignedUserUtil`** class contains methods for…
  - Finding, counting, and getting contacts for a user
  - Transferring contacts between users
  - Selecting the user with the fewest assigned contacts
- PCF requires just one method that returns a value

Summary

Update | Cancel | Suggest Least Busy User

Basic Information

Name	William Andy
Public ID	ab:5
Created On	05/08/2014
Assigned User	<none>

Summary

Update | Cancel | Suggest Least Busy User

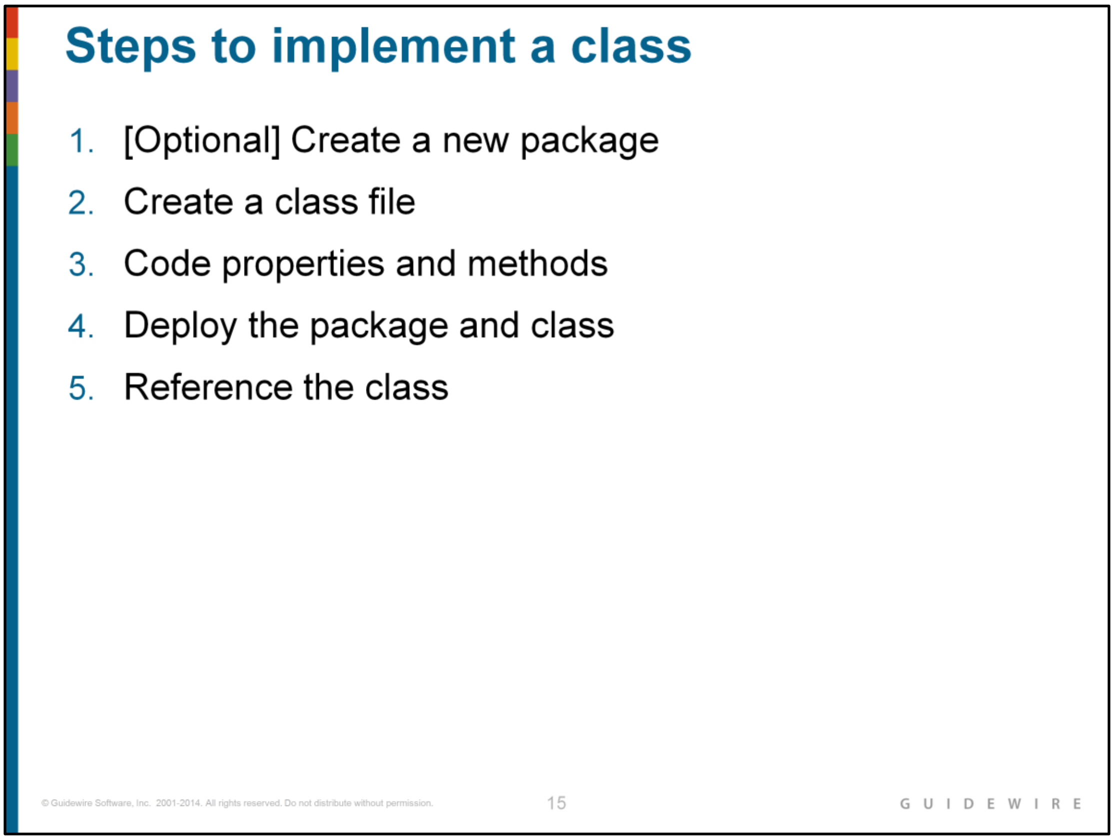
Basic Information

Name	William Andy
Public ID	ab:5
Created On	05/08/2014
Assigned User	Danielle Henson

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 14 GUIDEWIRE

As a sample business requirement, TrainingApp needs several methods that for working with contacts and assigned users. Because several methods are needed and these methods are called from multiple types of configuration objects, all methods will be declared in Gosu class.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



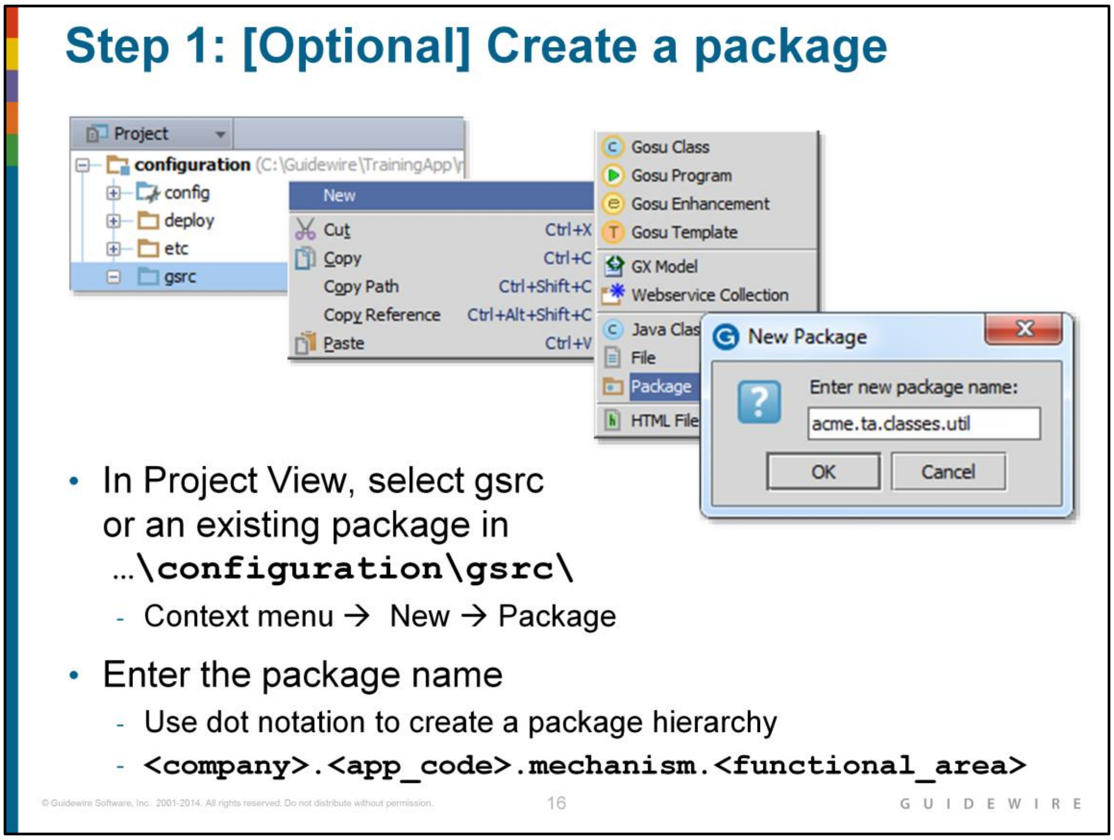
Steps to implement a class

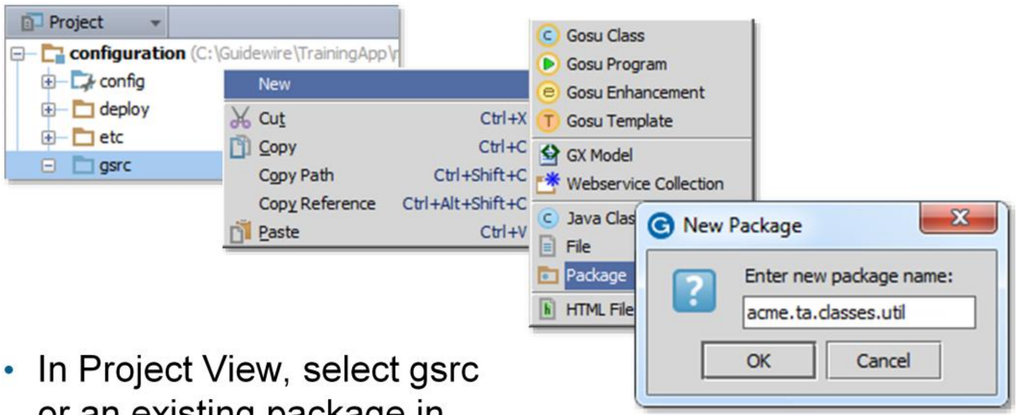
1. [Optional] Create a new package
2. Create a class file
3. Code properties and methods
4. Deploy the package and class
5. Reference the class

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.


Step 1: [Optional] Create a package



- In Project View, select gsrc or an existing package in …**`\configuration\gsrc\`**
  - Context menu → New → Package
- Enter the package name
  - Use dot notation to create a package hierarchy
  - **`<company>.<app_code>.mechanism.<functional_area>`**

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 16 GUIDEWIRE

In the slide example, the package reference is for illustration purposes and does not exist in TrainingApp. In the slide example, the dot notation creates a package hierarchy of four packages: acme, ta, classes, and util. Packages are always arranged in a hierarchy. When naming a package, the package name must be unique for the given parent package or \gsrc folder. Package names should be entirely lower case. Do not create child packages in com.guidewire or gw packages.

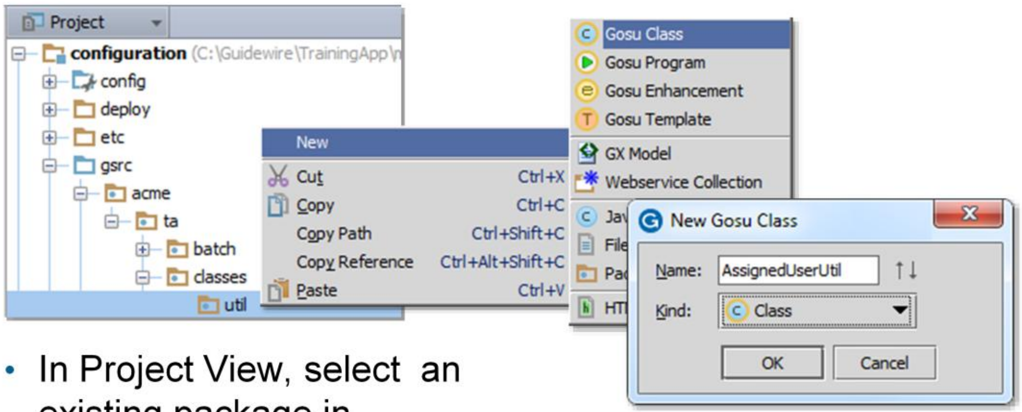
`<company>` is the company's name or a suitable abbreviation.
`<app-abbrv>` is the Guidewire application's two-letter abbreviation, for example PolicyCenter is pc, BillingCenter is bc, ClaimCenter is cc, ContactManager is ab or cm.
`<mechanism>` is often batch, messaging, plugin (for predefined non-messaging, non-startable plugins), startable, webservice, or class(es)
`<functional_area>` is the functional area of the application.

The naming convention for packages for Gosu classes is:
`<company>.<app-abbrv>.<mechanism>.<functional_area>`

A package is a collection of classes grouped together because they perform related logic, and/or they should have access to each other while classes outside the package should not (though this usage of packages is less common).

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.

Step 2: Create a class file



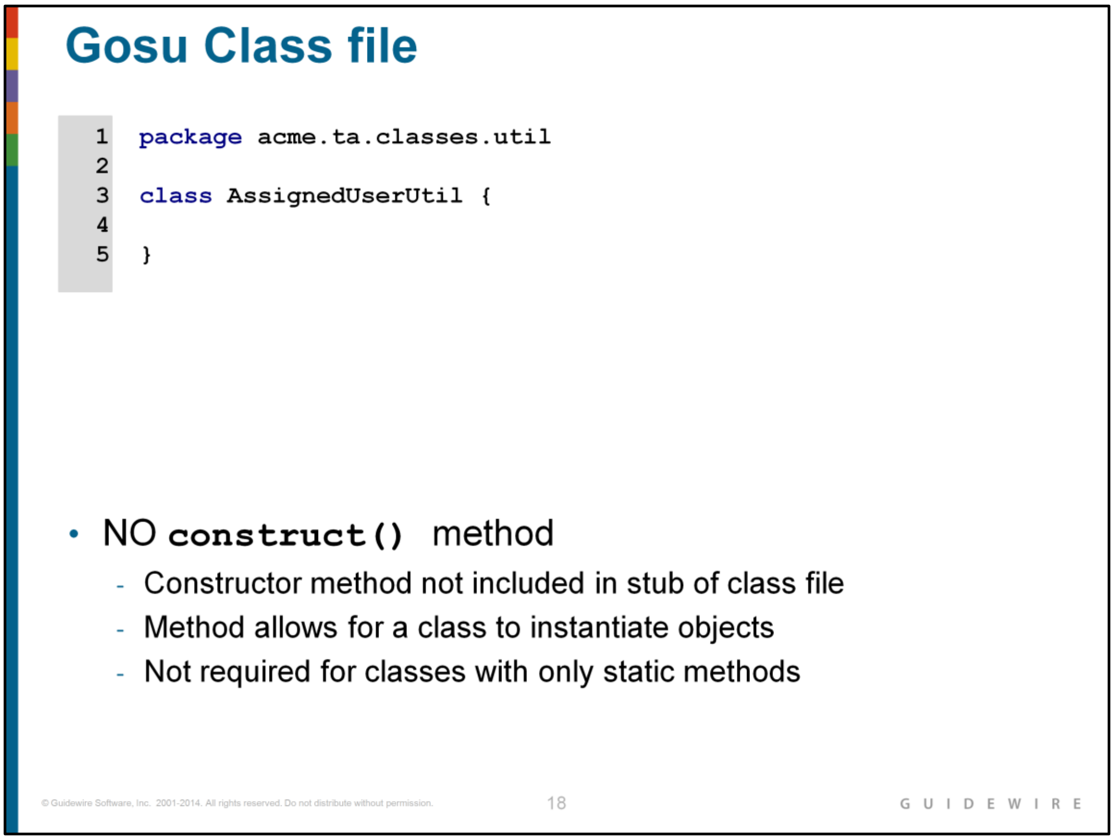
- In Project View, select an existing package in …`\configuration\gsrc\`
  - Context menu → New → Gosu Class
- Use meaningful name with Pascal Case
  - Capitalize the first letter in the identifier and the first letter of each subsequent concatenated word

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 17 GUIDEWIRE

In the slide example, the package reference is for illustration purposes and does not exist in TrainingApp. The recommended capitalization convention for Gosu class name is to use Pascal Case. In Pascal Case, the first letter in the identifier and the first letter of each subsequent concatenated word are capitalized.

When creating a class in Guidewire Studio, it is possible to create a hierarchy of packages for a given class using the dot notation for the class name. For example, in the New Class dialog, when you enter `acme.ta.classes.util.AssignedUserUtil`. Guidewire Studio will create a package hierarchy of four packages: acme, ta, classes, and util, and the AssignedUserUtil.gs class file.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Gosu Class file

```
package acme.ta.classes.util

class AssignedUserUtil {

}
```

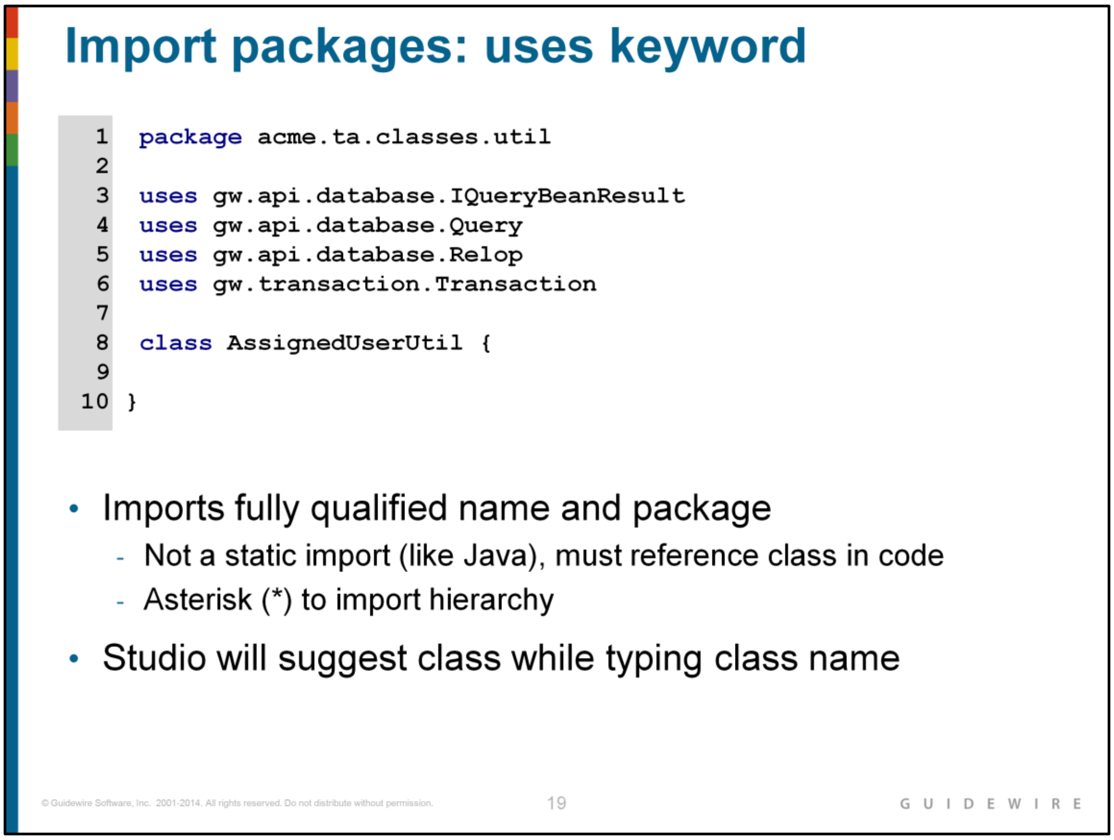
- NO **`construct()`** method
  - Constructor method not included in stub of class file
  - Method allows for a class to instantiate objects
  - Not required for classes with only static methods

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.
18
GUIDEWIRE

In the slide example, the fully qualified name (package and class) is for illustration purposes and does not exist in TrainingApp. You can find the class in trainingapp.base.AssigneUserUtil.

Because configuration-level classes typically consist only of static methods, instances of the class are never created and there is no need to specify a constructor. A constructor is defined with the keyword construct. The construct method allows for a class to instantiate objects.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Import packages: uses keyword

```
package acme.ta.classes.util

uses gw.api.database.IQueryBeanResult
uses gw.api.database.Query
uses gw.api.database.Relop
uses gw.transaction.Transaction

class AssignedUserUtil {

}
```

- Imports fully qualified name and package
  - Not a static import (like Java), must reference class in code
  - Asterisk (*) to import hierarchy
- Studio will suggest class while typing class name

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

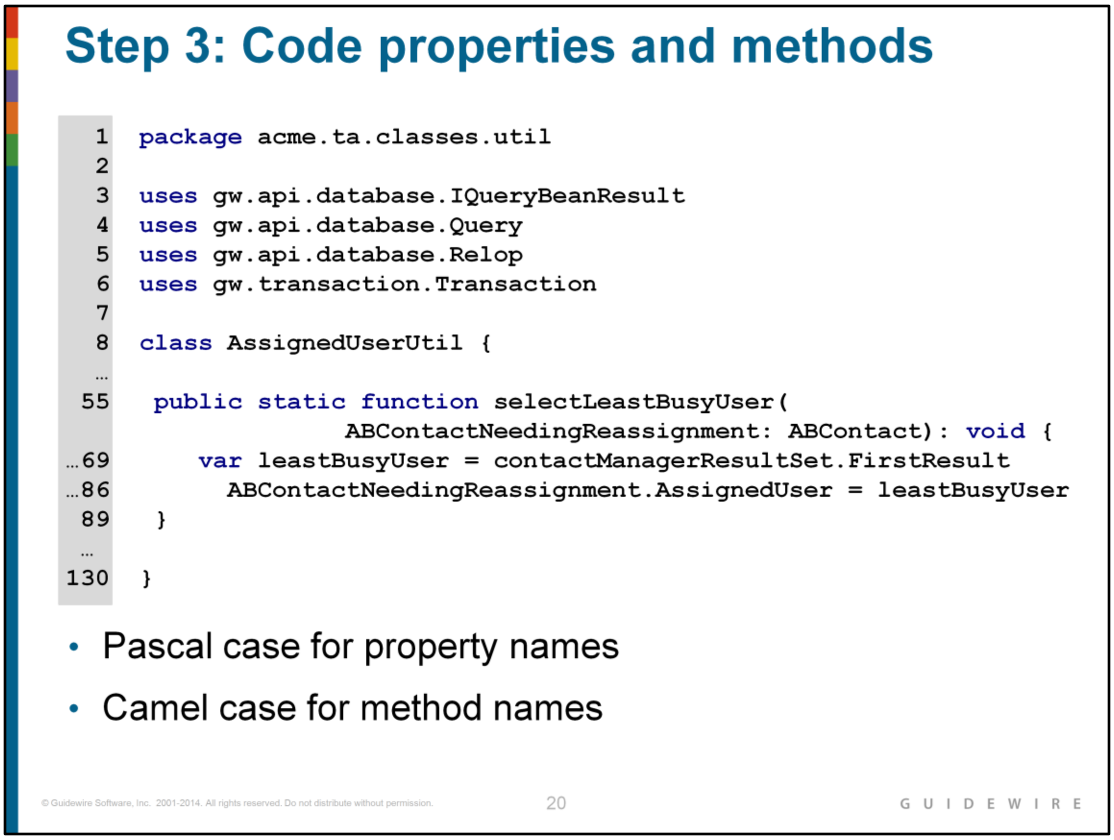
In the slide example, the fully qualified name (package and class) is for illustration purposes and does not exist in TrainingApp. You can find the class in trainingapp.base.AssigneUserUtil.

You do not need to always reference a class by fully qualified name. You can import the class with the **uses** operator (keyword). The uses operator behaves in a similar fashion to Java language’s import command, however, explicit types always have precedence over wildcard namespace references. In Gosu, you can include all classes in a package namespace by using an asterisk (*) character to indicate the hierarchy.

While the **uses** operator is technically an unary operator in that it takes a single operand, the functionality it provides is only useful with a second statement. In other words, the only purpose of using a uses expression is to simplify other lines of code in which you can omit the fully-qualified type name. For this reason, you will often call it a keyword instead of operator.

You can type in the class name and method and Studio will try to resolve the fully qualified name for you if the package or namespace has not been already declared. Use ALT+ENTER over the red squiggles to see the Import Class options.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Step 3: Code properties and methods

```
  1  package acme.ta.classes.util
  2
  3  uses gw.api.database.IQueryBeanResult
  4  uses gw.api.database.Query
  5  uses gw.api.database.Relop
  6  uses gw.transaction.Transaction
  7
  8  class AssignedUserUtil {
  …
 55   public static function selectLeastBusyUser(
                   ABContactNeedingReassignment: ABContact): void {
…69      var leastBusyUser = contactManagerResultSet.FirstResult
…86        ABContactNeedingReassignment.AssignedUser = leastBusyUser
 89    }
  …
130  }
```

- Pascal case for property names
- Camel case for method names

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

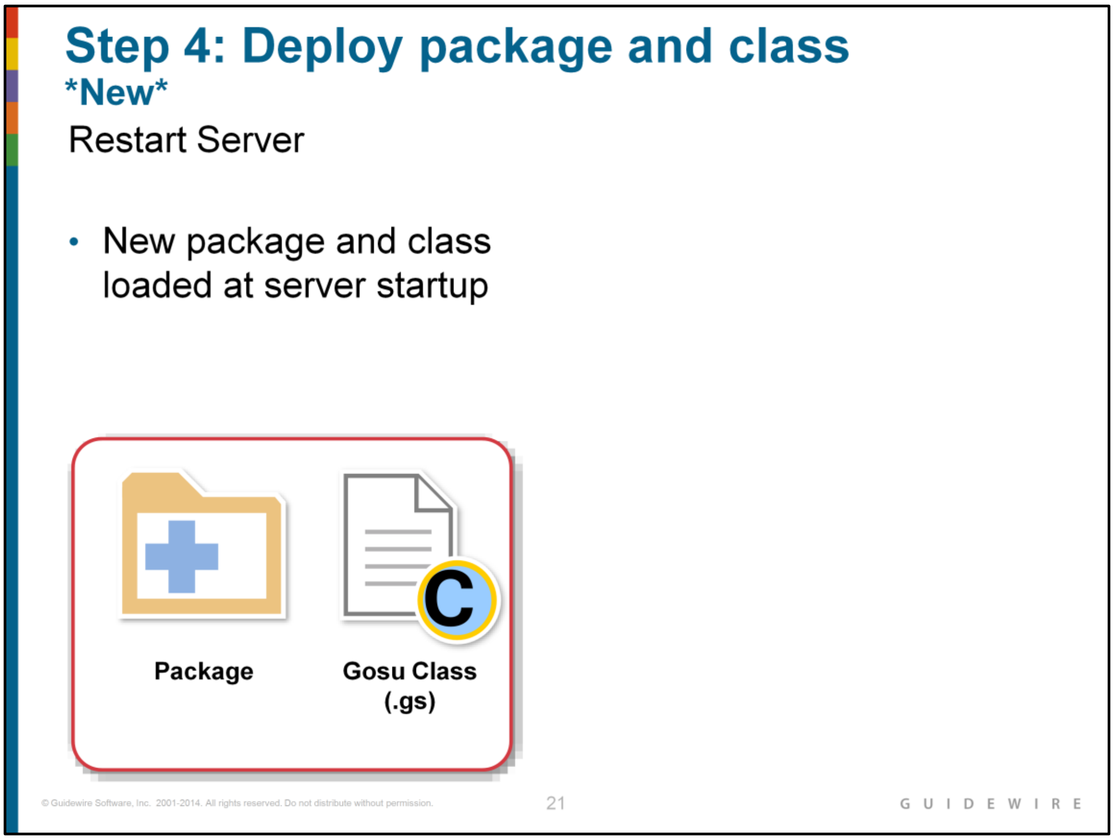
GUIDEWIRE

In the slide example, the fully qualified name (package and class) is for illustration purposes and does not exist in TrainingApp. You can find the class in trainingapp.base.AssigneUserUtil.
Line 1: declares the package in which the class exists.
Lines 3-6: imports other classes by fully qualified name for this class to reference using only the class name.
Line 8: declares the class name.
Lines 55-89: define a static method. The method does not return a value.

The recommended capitalization convention for properties is to use Pascal Case. In Pascal Case, the first letter in the identifier and the first letter of each subsequent concatenated word are capitalized. A property is used to get and set an object value. You can precede the property keyword with one or more modifier keywords. Guidewire modifiers include public and private, which are access modifiers. A property is public by default, meaning that it can be referenced from anywhere in a Guidewire application that uses Gosu.

The recommended capitalization convention for methods is to use Camel Case. In Camel Case, the first letter of an identifier is lowercase and the first letter of each subsequent concatenated word is capitalized. A method can return any type of value available in Gosu, including Boolean, String, and Integer. Like the property keyword, you can precede the function keyword with one or more modifier keywords. Guidewire modifiers include public and private, which are access modifiers. A method is public by default, meaning that it can be referenced from anywhere in a Guidewire application that uses Gosu. A static member means that the member exists only on the single type itself, not on *instances* of the type. Access static members on Java types just as you would native Gosu types. For Gosu code that accesses static members, you must qualify the class that declares the static member.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Step 4: Deploy package and class

New

Restart Server

- New package and class loaded at server startup

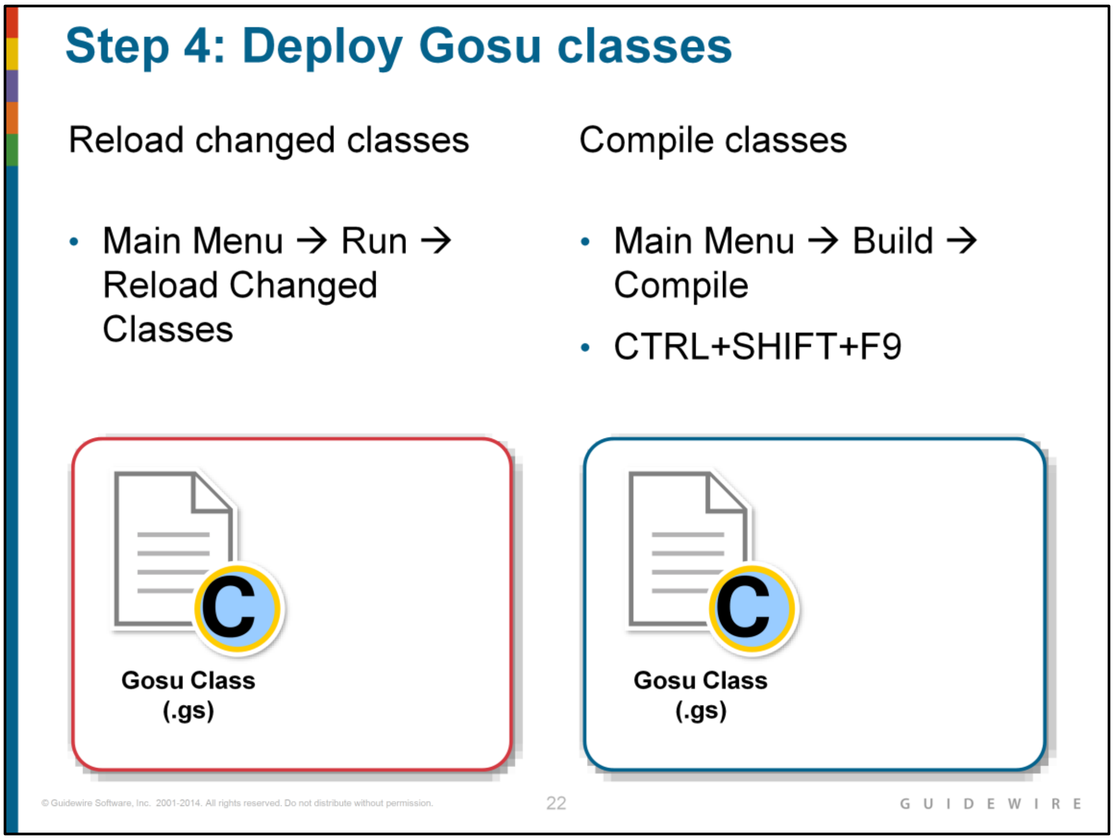
Package

Gosu Class (.gs)

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Step 4: Deploy Gosu classes

Reload changed classes

- Main Menu → Run → Reload Changed Classes

Compile classes

- Main Menu → Build → Compile
- CTRL+SHIFT+F9

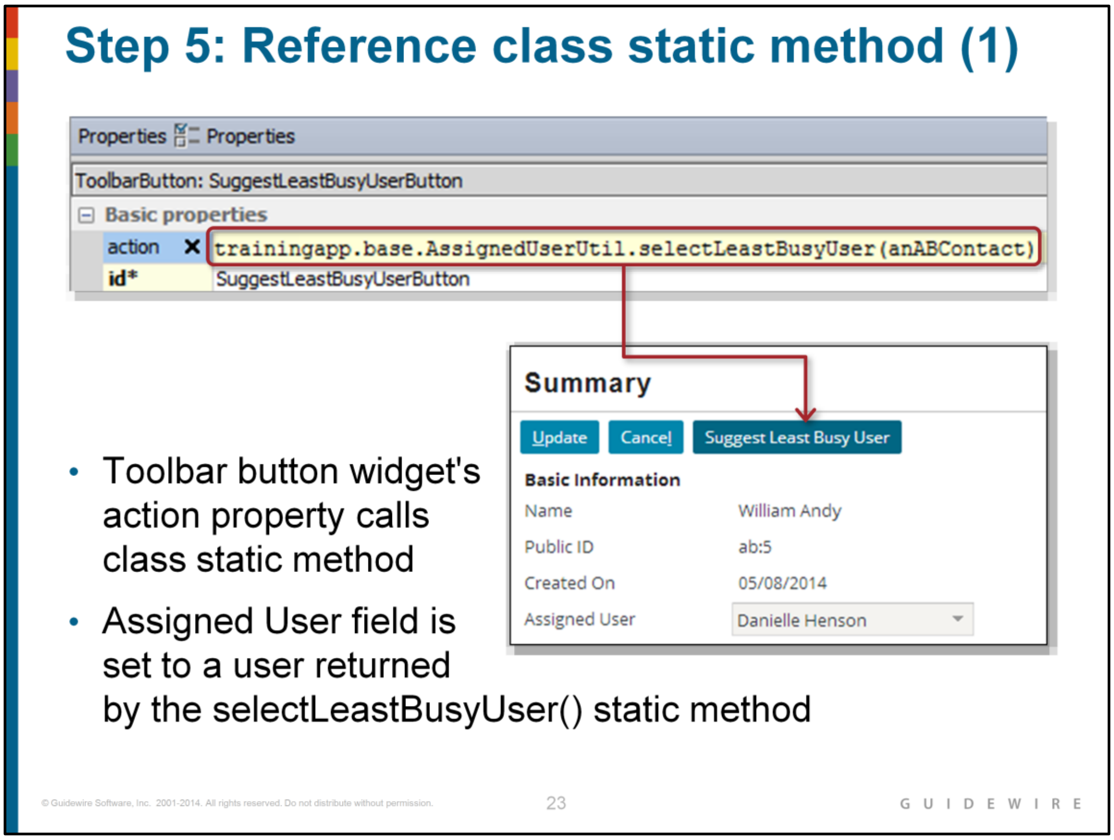
Gosu Class (.gs)
Gosu Class (.gs)

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. 22 GUIDEWIRE

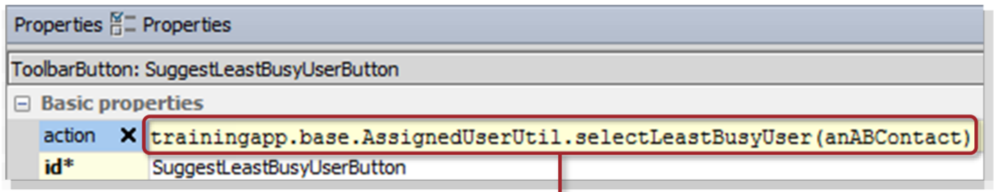
You can deploy new and modified Gosu classes when the server is running in run or debug server process. If the class exists in a new package, then you must restart the server.

If your modified enhancement (.gs) contains a new displaykey, then you will also need to reload PCF files using ALT+SHIFT+L, the Guidewire API and/or internal server tools.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Step 5: Reference class static method (1)

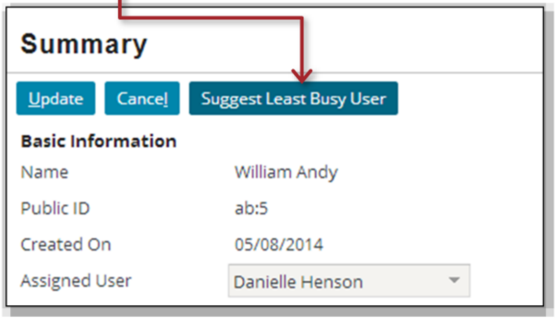


Properties | Properties

ToolbarButton: SuggestLeastBusyUserButton

Basic properties

action	`trainingapp.base.AssignedUserUtil.selectLeastBusyUser(anABContact)`
id*	SuggestLeastBusyUserButton



Basic Information

Name	William Andy
Public ID	ab:5
Created On	05/08/2014
Assigned User	Danielle Henson

- Toolbar button widget's action property calls class static method
- Assigned User field is set to a user returned by the selectLeastBusyUser() static method

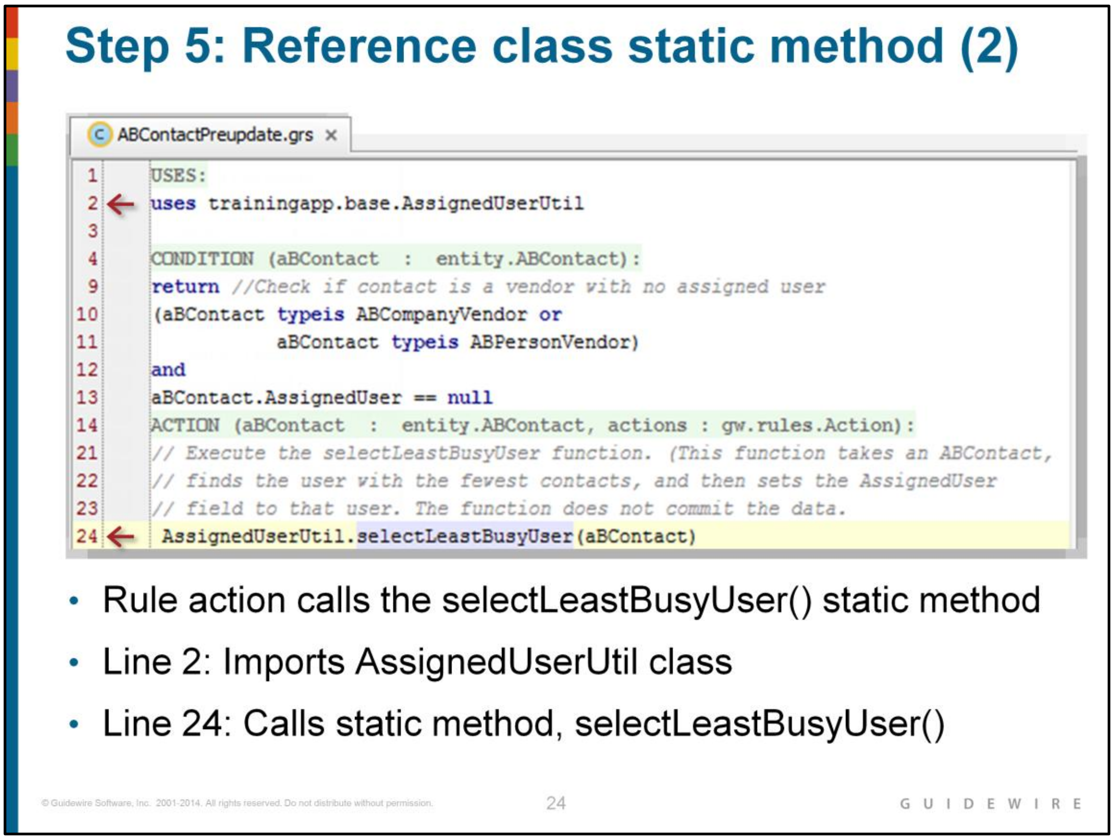
© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

The `selectLeastBusyUser()` method select the least busy user for a given anABContact. In the slide example, the toolbar button widget 's action property references the static method.

You can find the example in TrainingApp in ABContactSummaryPage.pcf.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Step 5: Reference class static method (2)

ABContactPreupdate.grs

```
1       USES:
2       uses trainingapp.base.AssignedUserUtil
3
4       CONDITION (aBContact  :  entity.ABContact):
9       return //Check if contact is a vendor with no assigned user
10      (aBContact typeis ABCompanyVendor or
11                aBContact typeis ABPersonVendor)
12      and
13      aBContact.AssignedUser == null
14      ACTION (aBContact  :  entity.ABContact, actions : gw.rules.Action):
21      // Execute the selectLeastBusyUser function. (This function takes an ABContact,
22      // finds the user with the fewest contacts, and then sets the AssignedUser
23      // field to that user. The function does not commit the data.
24       AssignedUserUtil.selectLeastBusyUser(aBContact)
```

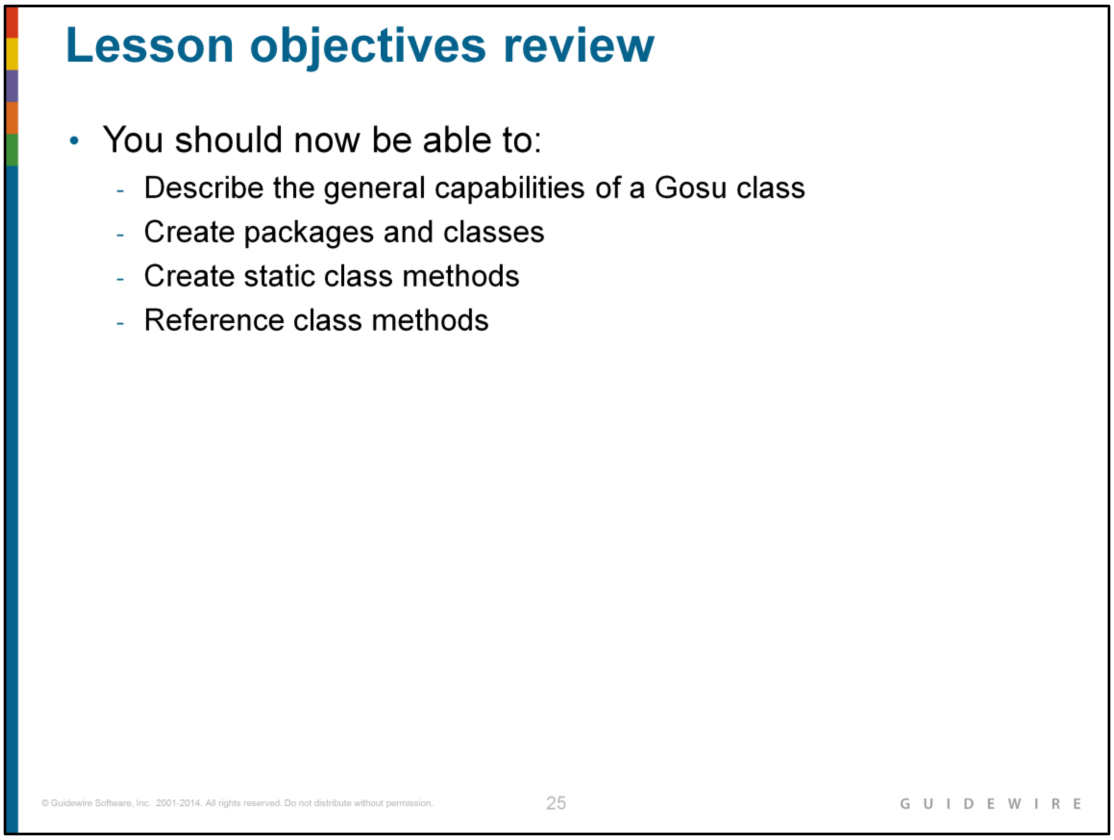
- Rule action calls the selectLeastBusyUser() static method
- Line 2: Imports AssignedUserUtil class
- Line 24: Calls static method, selectLeastBusyUser()

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission. GUIDEWIRE

The `selectLeastBusyUser()` method select the least busy user for a given aBContact. In the slide example, the rule action calls the static method.

You can find the example in TrainingApp in the "ABPU3000 - Subtype AB—Vendor" rule. The rule is in the ABContact Preupdate rule set in the Preupdate rule set category.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.

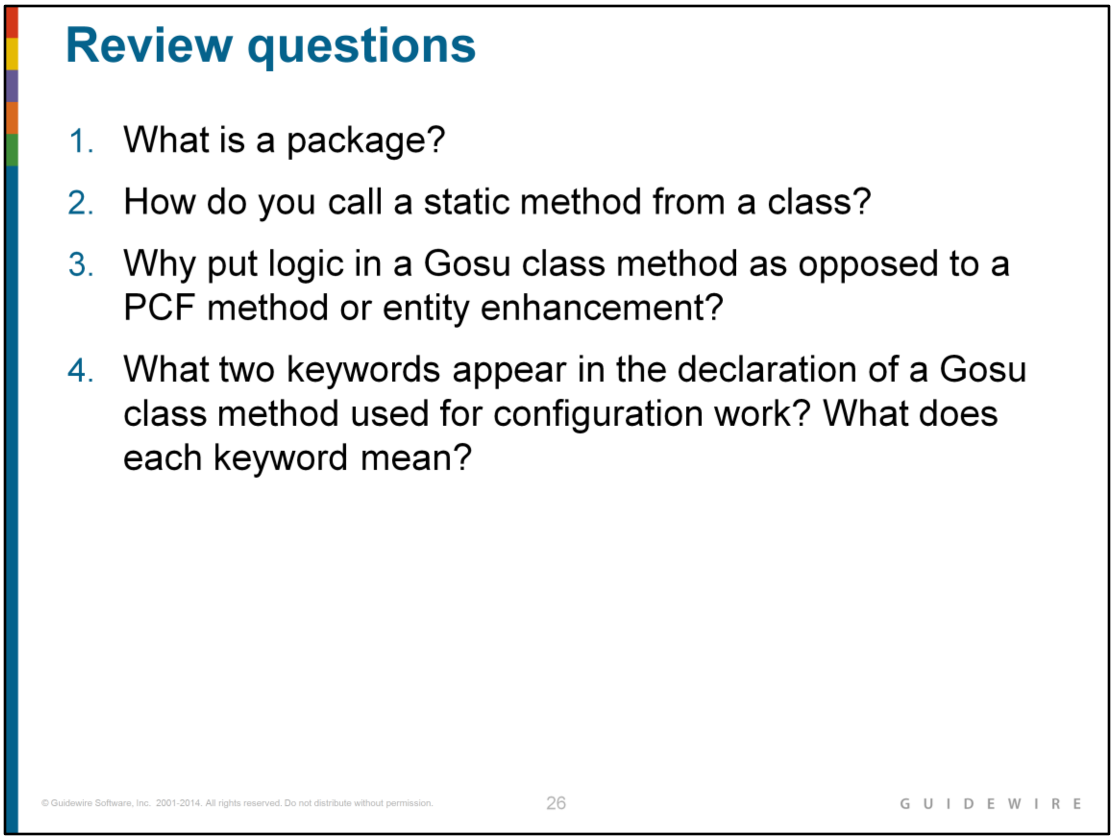


Lesson objectives review

- You should now be able to:
  - Describe the general capabilities of a Gosu class
  - Create packages and classes
  - Create static class methods
  - Reference class methods

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.


Review questions

1. What is a package?
2. How do you call a static method from a class?
3. Why put logic in a Gosu class method as opposed to a PCF method or entity enhancement?
4. What two keywords appear in the declaration of a Gosu class method used for configuration work? What does each keyword mean?

© Guidewire Software, Inc. 2001-2014. All rights reserved. Do not distribute without permission.

GUIDEWIRE

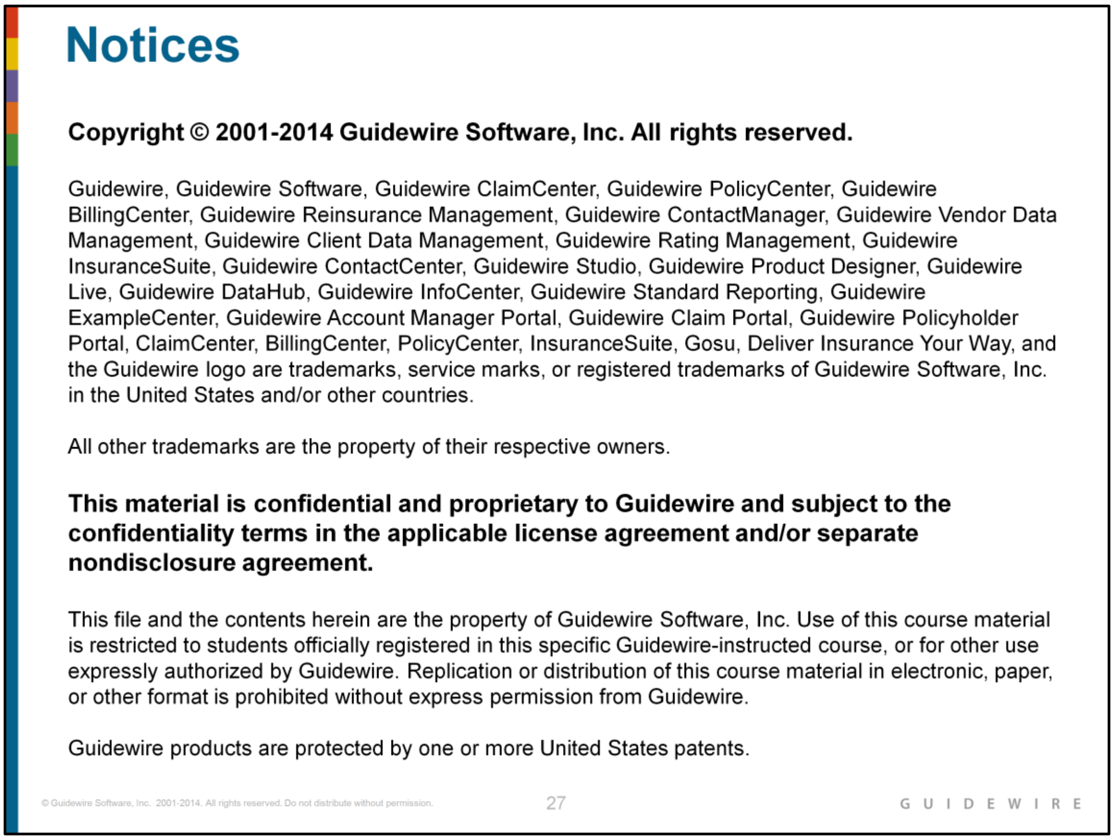
Answers
1) A package is a group of classes gathered together either because they execute a common business task, or because access to the classes must be limited only to the classes in the package, or both.

2) You call a static method from referencing the method name from the class. You must either reference the fully qualified name of the class or import the class using the uses keyword followed by the reference to the package path.

3) The logic in the class can managed in one place, yet multiple classes, entities, PCFS, anad Gosu enhancements can reference the Gosu class.

4) The public keyword identifies that the method is available to any code that invokes it outside of the class itself. The static keyword identifies that the method belongs to the class itself and can be declared even if there is no instance of the class.

© Guidewire Software, Inc. 2001-2014. All rights reserved.
Do not distribute without permission.



Notices

Copyright © 2001-2014 Guidewire Software, Inc. All rights reserved.

Guidewire, Guidewire Software, Guidewire ClaimCenter, Guidewire PolicyCenter, Guidewire BillingCenter, Guidewire Reinsurance Management, Guidewire ContactManager, Guidewire Vendor Data Management, Guidewire Client Data Management, Guidewire Rating Management, Guidewire InsuranceSuite, Guidewire ContactCenter, Guidewire Studio, Guidewire Product Designer, Guidewire Live, Guidewire DataHub, Guidewire InfoCenter, Guidewire Standard Reporting, Guidewire ExampleCenter, Guidewire Account Manager Portal, Guidewire Claim Portal, Guidewire Policyholder Portal, ClaimCenter, BillingCenter, PolicyCenter, InsuranceSuite, Gosu, Deliver Insurance Your Way, and the Guidewire logo are trademarks, service marks, or registered trademarks of Guidewire Software, Inc. in the United States and/or other countries.

All other trademarks are the property of their respective owners.

This material is confidential and proprietary to Guidewire and subject to the confidentiality terms in the applicable license agreement and/or separate nondisclosure agreement.

This file and the contents herein are the property of Guidewire Software, Inc. Use of this course material is restricted to students officially registered in this specific Guidewire-instructed course, or for other use expressly authorized by Guidewire. Replication or distribution of this course material in electronic, paper, or other format is prohibited without express permission from Guidewire.

Guidewire products are protected by one or more United States patents.