



创建、配置和运行模型到模型变换

目录

创建、配置和运行模型到模型变换 1

简介: 创建、配置和运行模型到模型变换 1

模块 1: 创建模型到模型变换映射项目和优化模型到模型映射 3

课程 1: 创建模型到模型变换映射项目 3

课程 2: 检查映射项目 4

课程 3: 创建和优化类到类映射声明 5

课程 4: 创建和优化类到接口映射声明 8

课程 5: 创建和优化操作到操作映射声明所需的映射声明 9

课程 6: 创建和优化包到包映射声明 13

课程 7: 创建和优化模型到模型映射声明 15

模块 2: 创建变换代码和运行变换 17

课程 1: 生成和编译变换源代码 17

课程 2: 配置运行时工作台 18

课程 3: 创建变换配置 19

课程 4: 运行变换 20

总结: 创建、配置和运行模型到模型变换 21

创建、配置和运行模型到模型变换

本教程描述如何创建“模型到模型”变换。可以通过创建变换映射项目来创建“模型到模型”变换。“模型到模型”变换可将某个模型变换为具有不同抽象级别的另一模型。本教程还描述如何在映射项目中创建和优化映射模型，以及如何生成和测试“模型到模型”变换代码。

学习目标

应当按指定的顺序完成本教程中的每个课程。本教程说明如何完成下列任务：

- 创建“模型到模型”变换映射项目
- 使用变换映射工具在“模型到模型”变换映射项目中创建和优化映射模型
- 生并和编译“模型到模型”变换源代码
- 在运行时工作台中配置和运行“模型到模型”变换

所需时间

约 60 分钟

相关信息

编写模型到模型变换



查看 PDF 版本

要查看此文件，必须在系统上安装 Adobe® Acrobat® Reader。

简介：创建、配置和运行模型到模型变换

本教程说明如何使用在 IBM® Rational® 建模产品中提供的映射工具创建、配置和运行“模型到模型”变换。在创建变换映射项目并且在项目中创建和优化映射模型之后，可以生成“模型到模型”变换的代码。在生成变换源代码之后，可以在运行时工作台中配置和运行变换。

在本教程中，将创建一个“模型到模型”变换，它把源模型中的类变换为目标模型中的接口和实现类。生成的实现类需要源类中操作的副本，而生成的接口只需要源类的公用操作的副本。

此教程可能需要一些可选可安装组件。要确保您安装了适当的可选组件，请参阅“系统要求”列表。

变换是一种模式实现策略，它采用一个源元素或一组元素，并将它们变成一个新的目标元素或一组元素。变换使您能够在模型与代码之间转换，以及在不同抽象级别的模型之间转换。创建变换也称为变换编写，是一个模型驱动的过程，它使您能够创建包含详细的实现信息的变换，或者创建指定模型或元模型之间的映射关系的变换。

“模型到模型”变换映射项目使您能够指定源和目标元模型，以及创建定义元模型中各元素之间的关系的映射模型。可以增量式生成实现关系的可扩展变换源代码。使用此级别的抽象使您能够着重于问题域而不是解决方案域。

“模型到模型”变换编写过程包括以下高级步骤：

1. 可以创建包含映射模型的“模型到模型”变换映射项目。映射项目可包含多个映射模型。创建映射项目时，变换服务将注册一个变换。每个变换都有一个变换提供程序，一个称为 **MainTransform** 的变换和项目中的每个映射声明一个变换。
2. 可以将映射声明（也称为映射）添加至映射模型。映射模型可包含一个或多个映射声明。
3. 可以将映射规则添加至映射模型中的映射声明。
4. 可以从映射项目中的一个或多个映射模型生成变换源代码。“模型到模型”变换编写工具为映射项目中的每个映射模型生成一个变换。对于每个映射声明，编写工具生成实现变换的 **Java™** 源文件。对于映射声明中的每个移动或定制映射规则，将在变换源代码中生成规则。对于映射声明中的每个子映射映射规则，将在变换源代码中生成内容抽取器。

学习目标

本教程包含两个模块，必须按顺序将它们完成。在这些模块中，将执行下列任务：

- 创建包含变换框架和映射模型的模型到模型变换编写项目
- 在映射模型中创建映射声明
- 通过完成下列任务优化映射声明：
 - 指定每个映射声明的输入对象和输出对象
 - 通过定义映射声明中输入/输出对象的属性之间的关系创建映射规则
- 生成和编译变换源代码
- 配置运行时工作台以测试模型到模型变换
- 在运行时工作台中，创建和应用运行模型到模型变换的变换配置

所需时间

完成本教程大约需要 60 分钟。如果要研究其他与本教程相关的概念，那么完成本教程的时间可能会延长。

技能级别

高级

适用对象

本教程的适用对象是开发人员。

系统要求

要完成本教程，必须安装变换编写组件。

还必须启用“建模”功能。

先决条件

要完成本教程，您应熟悉下列概念：

- Eclipse 建模框架（EMF）
- Eclipse 插件项目
- Ecore 模型
- Eclipse 工作台

模块 1：创建模型到模型变换映射项目和优化模型到模型映射

在此模块中，将创建包含多个映射声明的“模型到模型”变换映射项目。在每个映射声明中，您可以创建定义如何使映射声明的输入元素和输出元素的特性相关的映射规则。还可以了解项目中的文件，以及如何使用项目中的文件生成变换代码。

学习目标

本模块中的课程描述映射模型、映射声明和映射规则，并说明如何完成下列任务：

- 创建映射项目
- 创建和优化映射声明
- 管理映射规则

所需时间

完成本模块大约需要 40 分钟。

课程 1：创建模型到模型变换映射项目

本课程说明如何创建“模型到模型”变换映射项目。

“模型到模型”变换映射项目是一个标准 Eclipse 插件，它指定的变换提供程序是定义变换的机制。映射项目还至少包含一个映射文件；映射文件也称为映射模型。创建映射项目时，将自动在该项目中创建一个映射模型。

了解有关变换映射项目的更多信息：

“模型到模型”变换映射项目也称为映射项目，是对名为 `com.ibm.xtools.transform.core.transformationProviders` 的扩展点进行扩展的 Eclipse 插件。在变换映射项目中创建“模型到模型”变换使您能够着重于指定所选源和目标模型或元模型中的各元素如何相关，而不是创建表示变换的实现详细信息的代码。

映射项目可包含多个映射文件（也称为映射模型）。修改映射模型时，可以迭代方式生成变换源代码。生成变换源代码时，将会自动注册一个名为 `MainTransform` 的外部可视的变换，并且将为映射模型中的每个映射声明生成变换的 Java 源代码。

创建映射项目时，可以指定一个或多个源和目标元模型。可以指定元模型（文件扩展名为 `.ecore`）或 UML 概要文件（文件扩展名为 `.epx` 或 `.uml`）。如果在创建项目时指定源和目标元模型，就会自动将必需的依赖项添加至插件清单文件。如果在创建映射项目后通过使用编辑器区域中的命令添加元模型，必须将任何必需的新依赖项添加至插件清单文件。

要创建“模型到模型”变换映射项目：

1. 打开插件开发透视图：单击**窗口** → **打开透视图** → **其他**。在打开透视图窗口中，单击**插件开发**，然后单击**确定**。
2. 单击**文件** → **新建** → **项目**。
3. 在新建项目向导的选择向导页上，单击**变换编写**，然后单击**模型到模型映射变换项目**。
4. 单击**下一步**。
5. 在插件项目页上的**项目名称**字段中，输入 `Generalize Classes`。对于该页上的其他字段，接受缺省值。
6. 单击**下一步**。
7. 在插件内容页上，复审各值，然后单击**下一步**。
8. 在模板页上，从**可用的模板**列表中，选择具有变换映射的插件。

9. 单击下一步。
10. 在新建变换映射页上的**映射名称**字段中，如果尚不存在值，请输入 `Generalize_Classes`。此字段指定映射模型的名称；该映射模型位于项目的 `model` 文件夹中，并且文件扩展名为 `.mapping`。
11. 在**包名**字段中，如果名称尚不存在，请输入 `generalize_classes`。在后面的课程中，生成变换源代码时，就会在名为 `generalize_classes.transforms` 的文件夹中创建变换的 Java 源代码。
12. 要指定输入模型和输出模型，请完成以下步骤：
 - a. 在新建变换映射页上“输入模型”区域的旁边，单击**添加模型**。
 - b. 在装入资源对话框中，单击适当的按钮以浏览至模型。对于本教程，请单击**浏览已注册的包**，选择最新版本的UML2 模型（有关其命名约定，请访问以下网址：<http://www.eclipse.org/uml2/2.x.y/UML>），然后单击**确定**。此步骤指定：变换接受 UML.ecore 元模型作为变换源。
 - c. 在新建变换映射页上“输出模型”区域的旁边，单击**添加模型**。
 - d. 重复步骤 12b，指定变换输出是类型为 `UML.ecore.metamodel` 的模型。此步骤指定：变换输出为 UML.ecore 元模型。
 - e. 单击**确定**。

如果在创建项目时指定输入元模型和输出元模型，就会自动将必需的依赖项添加至插件清单文件。如果在创建映射项目后通过使用编辑器区域中的命令添加元模型，必须将任何必需的新依赖项添加至插件清单文件。

13. 单击**完成**。

这就在工作空间中创建了映射项目。在下一课中，将检查映射项目的结构。

课程 2：检查映射项目

在创建映射项目后，可以使用包资源管理器视图检查项目的结构。

要检查映射项目的内容：

1. 在包资源管理器视图中，展开 **Generalize Classes** 映射项目并观察生成的文件。
2. 浏览至 `model` 文件夹。注意此文件夹如何包含名称与变换映射项目相同的映射模型；该文件的文件扩展名也是 `.mapping`。映射项目可以包含多个映射模型。在本教程的后面部分您将把映射声明添加至此映射模型。

了解有关映射模型的更多信息：

映射模型也称为映射文件，是 Eclipse 建模框架（EMF）元模型（也称为 Ecore 模型，包含对要映射的元模型的引用）的实例。创建映射项目时，编写工具将使用您指定的输入/输出模型在项目中创建映射模型。映射模型的文件扩展名为 `.mapping`。

映射模型以 XML 文件的形式存储和序列化。“问题”视图显示有关映射模型的详细错误信息。在此视图中，双击某项以在文本编辑器中打开映射模型，并查看包含错误的行。此故障诊断方法通常比通过在编辑器区域中查看映射模型而进行故障诊断容易一些。

3. 在包资源管理器视图中，展开 `src` 文件夹。`generalize_classes.transforms` 文件夹包含生成的变换的源代码。此时，只存在一个名为 `MainTransform` 的缺省变换。在本教程的后面部分，您将创建映射声明并重新生成变换源代码。对于映射模型中的每个映射声明，变换编写框架将生成名为 `nTransform` 的 Java 类，其中 `n` 表示映射声明的名称。这些 Java 类一起组成变换代码。
4. 在包资源管理器视图的 `model` 文件夹中，双击映射模型。变换映射编辑器将会打开，该编辑器使您能够创建映射声明以及优化每个映射声明中的映射规则。将在此模块接下来的课程中完成这些任务。

课程 3: 创建和优化类到类映射声明

本课程说明如何创建指定 UML 类作为输入/输出对象的映射声明。此“类到类”映射声明包含一些映射规则，当运行生成的变换时，这些映射规则将创建源模型中的类及其操作的副本，并将副本放置在目标模型中。还可以创建“操作到操作”映射声明，即在“类到类”映射声明中调用的子映射。

映射声明也称为映射，指定如何创建或更新给定输入对象的输出对象。映射声明使您能够指定输入对象中的属性如何与输出对象中的属性对应。每个映射声明都指定输入类型和输出类型，从添加至映射模型的元模型中选择这些类型。

对于映射模型中的每个映射声明，变换编写框架将生成名为 *nTransform.java* 的 Java 源文件，其中 *n* 表示映射声明的名称。这些 Java 文件一起组成变换代码。除了生成变换的实现代码之外，变换编写框架还生成代码以向变换服务注册变换。在创建映射声明之后，可以增量式添加映射规则以及生成映射规则的源代码或实现。不必在生成源代码之前定义所有映射规则。

了解有关映射声明的更多信息:

映射声明通常遵循以下命名约定: *x2y*，其中 *x* 表示输入对象类型，而 *y* 表示输出对象类型。例如，名为 *Package2EPackage* 的映射声明指定使用 *Package* 作为输入对象和使用 *EPackage* 作为输出对象的映射声明。

元素之间的映射建立其属性之间的对应关系，这样它们之间可以交换数据。大多数映射能够进一步处理源和目标之间的数据。例如，通过创建将值指定给目标的定制代码，可以选择指定计算或对数据进行其他修改。

要在映射模型中创建“类到类”映射声明:

1. 如果 *.mapping* 文件尚未打开，请在包资源管理器视图的 *model* 文件夹中，双击 *.mapping* 文件。
2. 在变换映射编辑器的“映射根”部分中，右键单击 **Generalize_Classes**；然后单击**创建映射**。
3. 在新建映射窗口的**映射名称**字段中，输入 *Class2Class*；然后单击**确定**。映射就会显示在“大纲”视图中，并且在编辑器区域中“映射根”部分下面打开。

将输入/输出对象添加至类到类映射声明

在创建映射声明之后，必须向其添加输入对象和输出对象。在本课中，将指定一个 UML 类作为输入/输出对象。

注: 在创建“模型到模型”变换时，要添加在映射使用的输入或输出元模型中不可用的对象类型，那么可以将适当的 UML 概要文件或 *Ecore* 元模型添加至映射的作用域。在添加输入窗口中，单击**添加模型**并指定相应的模型。

要将输入对象和输出对象添加至“类到类”映射声明:

1. 单击**添加输入对象**图标，即正在编辑的映射的工具栏上最左边的那个图标:



2. 在添加输入窗口的“元素”窗格中，选择元模型对象。“元素”窗格就会显示指定为映射模型的源或目标的一个或多个元模型中的元素。对于本教程，在“元素”窗格中，展开 **uml**，单击**类**，然后单击**确定**。
3. 单击**添加输出对象**图标，即正在编辑的映射的工具栏左边的第二个图标:



4. 在“添加输出”窗口的“元素”窗格中，展开 **uml**，单击**类**，然后单击**确定**。
5. 单击**文件** → **保存**。

现在，可以定义类输入/输出对象的属性之间的映射规则了。

定义类输入/输出对象的属性之间的映射规则

在将输入/输出对象添加至映射声明之后，就可以定义属性之间的映射规则了。映射规则也称为映射，指定如何根据输入对象的属性的值将值指定给输出对象的属性。

在“类到类”映射声明中，请完成以下步骤：

- 创建一个移动映射规则，该映射规则在目标模型中创建类

创建输入/输出对象的 `name` 属性之间的映射规则。目标类的名称与输入模型中类的名称相同；也可以将此操作看作创建类的副本。在后面的课程中，将把映射操作添加至映射规则。

- 创建一个子映射映射规则，该映射规则将对类中的每个操作在目标模型中的类中创建相应的操作。

可以创建输入/输出对象的 `ownedOperation` 属性之间的子映射映射规则。对于 `ownedOperation` 集合中的每个操作，目标模型中生成的操作的 `name` 和 `visibility` 将与输入模型中操作的 `name` 和 `visibility` 相同。

了解有关映射规则的更多信息：

映射规则也称为映射，指定如何根据输入对象的属性的值将值指定给输出对象的属性。可以在输入/输出对象之间创建多个映射规则，还可以在输入/输出对象的属性之间创建多个映射规则，如本教程后面的课程所示。可以创建以下类型的映射规则：

移动 移动，也称为简单映射规则，是最基本的映射规则类型。输入/输出属性必须是兼容的数据类型。输入/输出属性都包含多个值，或者都不包含多个值。例如，如果输出对象的属性是 `String`，并且无须定制代码就可将输入对象的属性转换为 `String`，请选择此选项。此类型的映射规则支持一个源元素或属性与一个目标元素或属性之间的映射。为移动映射规则生成的变换源代码实现将一个输入属性的值复制到一个输出属性的规则。

子映射

子映射是从一个映射中调用另一个映射。被调用的子映射可以（但不一定）在定义调用它的映射的同一映射文件中定义。子映射使您能够将输入模型中的复杂类型映射至输出模型中的复杂类型。您创建的子映射可以调用存在于任何映射文件中的映射。在独立的映射文件中定义子映射鼓励复用映射；但是，创建多个映射文件可能会增加项目维护难度。子映射也可包括其他子映射，这会生成分层结构。

子映射映射规则支持以下类型的映射：

- 输入对象和输出对象之间，或输入/输出对象的属性之间的 1 对 1 映射
- 输入/输出对象的属性之间的 1 对 m 或 m 对 1 映射
- 输入/输出对象的属性之间的 m 对 n 映射

如果子映射规则指定 1 对 m 映射，那么生成的变换将来自单个属性的对象追加至列表；对于 m 对 1 映射，变换从列表中抽取对象并将它插入到单个属性中。

还可以在映射声明中的输入对象和输出对象之间创建子映射映射规则。

对于映射声明中的每个子映射，将在包含变换中生成名为 `getInputFeatureToOutputFeature_UsingMap_Extractor` 的抽取器，其中 `InputFeature` 表示输入属性的名称，`OutputFeature` 表示输出属性的名称，而 `Map` 表示映射声明的名称。

定制 此映射规则类型使您能够指定计算输出属性的值的定制代码。例如，选择此映射类型以将输出对象中属性的值设置为等于多个输入对象属性的并置。

可以使用 Eclipse 提供的“对象约束语言”（OCL）API 指定语义改进。

定制映射规则支持以下类型的映射：

- 输入/输出对象之间，或者输入/输出对象的属性之间的 1 对 n 映射
- 输入/输出对象之间，或者输入/输出对象的属性之间的 m 对 n 映射
- m 对 1 映射，其中 m 对 1 表示以下映射之一：
 - 从多重性设置为 m 的单个输入属性到多重性设置为 1 的单个输出属性的映射
 - 从多个输入属性到单个输出属性的映射
 - 从多个输入对象到单个输出对象的映射

被继承的映射

只能在输入对象和输出对象之间创建此映射规则。

进行继承的映射声明继承在被继承的映射声明中定义的映射规则。可以通过定义具有以下性质的映射规则来覆盖被继承的映射规则：

- 输入对象属性和输出对象属性与被继承的映射规则相同。
- 覆盖的和被继承的映射规则是具有匹配的抽取器的子映射映射，或者覆盖的和被继承的映射规则都是移动映射或定制映射。

如果创建被继承的映射映射规则，必须指定包含想要继承的映射规则的映射声明。不能在映射声明中指定多个被继承的映射。

被继承的映射规则以及根据该映射生成的规则或抽取器，就处理顺序中的相对位置而言，与被覆盖的映射规则及其生成的规则或抽取器一致。

对于映射声明中的每个移动或定制映射规则，会将规则添加至生成的变换源代码。对于每个子映射映射规则，将在变换源代码中生成内容抽取器。创建映射规则时，其类型由您选择的输入/输出属性确定。例如，如果输入/输出属性与基本类型兼容，如字符串或整数，请指定移动映射规则。如果输入/输出属性是复杂类型，请指定子映射规则。如果移动和子映射都不是适当的映射规则类型，请指定定制映射规则。

要在 Class2Class 映射声明中定义映射规则：

1. 在输入/输出对象的 `name` 属性之间创建移动映射规则：
 - a. 在编辑器区域的类输入对象中，单击 `name` 属性。
 - b. 将 `name` 属性的句柄拖至类输出对象的 `name` 属性。
2. 在输入/输出对象的 `ownedOperation` 属性之间创建子映射映射规则：
 - a. 在编辑器区域的类输入对象中，单击 `ownedOperation` 属性。
 - b. 将 `ownedOperation` 属性的句柄拖至类输出对象的 `ownedOperation` 属性。因为 `ownedOperation` 属性是集合，所以在缺省情况下，将创建子映射映射规则。
3. 单击**文件** → **保存**。

现在，可以创建“操作到操作”映射声明了。

创建操作到操作映射声明

在本课中，此时，在编辑器区域中，子映射具有有红色圆圈圈起来的 `X`（表示错误）的修饰符。将鼠标指针放在此修饰符上以查看错误消息。错误消息说明必须选择要调用的子映射映射规则的映射声明。要解决此错误，可创建“操作到操作”映射声明。

要创建“操作到操作”映射声明：

1. 如果属性视图未打开，请在编辑器区域中右键单击在前一部分的步骤 2 中创建的子映射元素，然后单击在**属性中显示**。

2. 在属性视图的**详细信息**选项卡上，在**映射**字段旁边，单击**新建**。
3. 在新建映射窗口的**映射名称**字段中，输入 `Operation2Operation`。
4. 如果**输入**字段不包含值 `Operation`，请单击**浏览**，然后在添加输入窗口的“元素”窗格中，单击 **Operation**。
5. 如果**输出**字段不包含值 `Operation`，请单击**浏览**，然后在添加输出窗口的“元素”窗格中，单击 **Operation**。
6. 单击**确定**。这就除去了错误修饰符，并且将在大纲视图中显示“操作到操作”映射声明。在后面的课程中，将在此映射声明中创建映射规则。
7. 单击**文件** → **保存**。

在下一课中，将定义“类到接口”映射声明。

课程 4：创建和优化类到接口映射声明

本课程说明如何在映射模型中创建“类到接口”映射声明。此映射声明包含一些映射规则，这些映射规则将 1) 创建一个接口，该接口的名称派生自源模型中类的名称；2) 仅复制源模型中类的公用方法。

要在映射模型中创建“类到接口”映射声明：

1. 如果未在映射编辑器中打开 `Generalize_Classes.mapping` 文件，请在包资源管理器视图中双击该文件。
2. 在变换映射编辑器的“映射根”部分中，右键单击 **Generalize_Classes**；然后单击**创建映射**。
3. 在新建映射窗口的**映射名称**字段中，输入 `Class2Interface`；然后单击**确定**。映射就会显示在大纲视图中，并且在编辑器区域中“映射根”部分下面打开。

注：要查看未在编辑器区域中打开的映射声明的详细信息，在大纲视图中，展开该映射声明的名称。要打开另一个映射声明，在大纲视图中，单击映射声明的名称。

将输入/输出对象添加至类到接口映射声明

在创建“类到接口”映射声明之后，将 UML 类指定为输入对象，将 UML 接口指定为输出对象。

要将输入对象和输出对象添加至“类到接口”映射声明：

1. 单击**添加输入对象**图标，即正在编辑的映射的工具栏上最左边的那个图标。
2. 在添加输入窗口的“元素”窗格中，展开 **uml**，单击**类**，然后单击**确定**。
3. 单击**添加输出对象**图标，即正在编辑的映射的工具栏左边的第二个图标。
4. 在添加输出窗口的“元素”窗格中，展开 **uml**，单击**接口**，然后单击**确定**。
5. 单击**文件** → **保存**。

现在，可以定义类输入对象和接口输出对象的属性之间的映射规则了。

定义输入/输出对象的属性之间的映射规则

在将输入/输出对象添加至“类到接口”映射声明之后，可以定义输入/输出对象的属性之间的映射规则。

对于本课程，将创建一个定制映射规则，而该规则将创建名为 `IClassName` 的接口。也可以创建一个调用在第 5 页的『课程 3：创建和优化类到类映射声明』中创建的“操作到操作”子映射的子映射映射规则，然后将输入过滤器添加至该子映射映射规则，以指定仅将公用操作变换为目标接口。

要定义输入/输出对象的属性之间的映射规则：

1. 创建输入/输出对象的 **name** 属性之间的定制映射规则：

- a. 在编辑器区域的类输入对象中，单击 **name** 属性。
 - b. 将 **name** 属性的句柄拖至接口输出对象中的 **name** 属性。这就创建了移动映射规则。
 - c. 在连接 **name** 属性的**移动**元素上，单击向下箭头，然后单击**定制**。
 - d. 右键单击**定制**元素；然后单击**在属性中显示**。
 - e. 在属性视图的**详细信息**选项卡上，单击**直接插入**，然后在方法特征符下面的文本区域中，输入以下代码：`Interface_tgt.setName("I"+Class_src.getName());`

注：要调用内容辅助功能，在输入代码时，按 **Alt+I**。
 - f. 单击**应用**。
2. 在输入/输出对象的 **ownedOperation** 属性之间创建子映射映射规则：
 - a. 在编辑器区域的类输入对象中，单击 **ownedOperation** 属性。
 - b. 将 **ownedOperation** 属性的句柄拖至接口输出对象的 **ownedOperation** 属性。因为 **ownedOperation** 属性是集合，所以在缺省情况下，将创建子映射映射规则。
 - c. 如果属性视图没有打开，请右键单击新的**子映射**元素；然后单击**在属性中显示**。
 - d. 如果尚未选择 **Operation2Operation**，请在属性视图的**详细信息**选项卡中，从**映射列表**中进行选择。
 3. 对于在步骤 2 中创建的子映射映射规则，添加只将公用操作变换为目标接口的输入过滤器：
 - a. 在“属性”视图中的**输入过滤器**选项卡上，单击**过滤器输入元素**，然后单击**直接插入**。
 - b. 在方法特征符下面的**代码**选项下面的文本区域中，输入以下代码：`if (ownedOperation_src.getVisibility().equals(VisibilityKind.PUBLIC_LITERAL)) { return true; } return false;`

注：为输入或输出过滤器指定的代码必须返回布尔值。只能指定方法主体；变换编写框架定义方法特征符。要查看有效变量名称的列表，在**直接插入**按钮下面的文本区域中，按 **Alt+I**。
 - c. 单击**应用**。
 4. 单击**文件** → **保存**。

在下一课中，将在 **Generalize_Classes.mapping** 模型中创建几个映射声明；这些映射声明定义运行生成的变换时如何变换参数和基本类型。

课程 5：创建和优化操作到操作映射声明所需的映射声明

本课程说明如何创建“操作到操作”映射声明变换操作中的元素所需的映射声明。例如，将创建一个映射声明，把源模型中的参数变换为目标模型中的参数；还将创建一个映射声明，定义如何将源模型中的基本类型变换为目标模型中的基本类型。在本课中，还将为“操作到操作”映射声明创建映射规则，这些规则调用在本课中创建的映射声明。

下表列示了将在本课中创建的映射声明和映射规则：

映射声明	输入/输出对象类型	被映射的属性	进行映射的规则类型	进行映射的规则描述
Primitive2Primitive	UML 基本类型	name	移动	在子映射调用 Primitive2Primitive 映射声明的元素中创建基本类型

映射声明	输入/输出对象类型	被映射的属性	进行映射的规则类型	进行映射的规则描述
Parameter2Parameter	UML 参数	name, visibility	移动	在子映射调用 Parameter2Parameter 映射声明的元素中创建参数；生成的参数的 name 和 visibility 与源输入元素中的参数的 name 和 visibility 相同
		type	子映射	对于子映射调用此映射声明的元素中的每个类型，存在以下结果： <ul style="list-style-type: none"> • 如果参数是基本类型，那么此规则调用 Primitive2Primitive 映射声明 • 如果参数的类型是类，那么此规则调用 Class2Class 和 Class2Interface 映射声明
Operation2Operation	UML 操作	name, visibility	移动	在子映射调用 Operation2Operation 映射声明的元素中创建操作；生成的操作的 name 和 visibility 与源输入元素中操作的 name 和 visibility 相同
		ownedParameter	子映射	对于操作中的每个参数，此子映射调用 Parameter2Parameter 映射声明

在本教程中，Paramater2Parameter 映射声明调用 Primitive2Primitive 映射声明以创建类型为基本类型的参数。

要在映射模型中创建“基本类型到基本类型”映射声明：

1. 如果 Generalize_Classes.mapping 文件尚未打开，请在包资源管理器视图中的 model 文件夹中，双击该文件。
2. 在变换映射编辑器的“映射根”部分中，右键单击 **Generalize_Classes**；然后单击**创建映射**。
3. 在新建映射窗口的映射名称字段中，输入 Primitive2Primitive 作为映射的名称；然后单击**确定**。映射就会显示在“大纲”视图中，并在编辑器区域中的“映射根”下面打开。

将输入/输出对象添加至 Primitive2Primitive 映射声明

在创建 Primitive2Primitive 映射声明之后，必须向其添加输入对象和输出对象。在本课中，将指定一个 UML 基本类型作为输入/输出对象。

要将输入对象和输出对象添加至 Primitive2Primitive 映射声明：

1. 单击**添加输入对象**图标，即正在编辑的映射的工具栏上最左边的那个图标。
2. 在“添加输入”窗口的“元素”窗格中，展开 **uml**，单击**基本类型**，然后单击**确定**。

3. 单击**添加输出对象**图标，即正在编辑的映射的工具栏左边的第二个图标。
4. 在“添加输出”窗口的“元素”窗格中，展开 **uml**，单击**基本类型**，然后单击**确定**。
5. 单击**文件** → **保存**。

现在，可以定义基本类型输入/输出对象的 **name** 属性之间的映射规则了。

定义基本类型输入/输出对象的属性之间的映射规则

对于本课程，创建一个移动映射规则，该映射规则在目标模型中创建基本类型。生成的基本类型的名称与输入模型中基本类型的名称相同；也可以将此操作看作创建基本类型的副本。

要定义移动映射规则以定义基本类型输入/输出对象的 **name** 属性之间的关系：

1. 在编辑器区域的基本类型输入对象中，单击 **name** 属性。
2. 将 **name** 属性的句柄拖至输出对象中的 **name** 属性。
3. 单击**文件** → **保存**。

现在，可以创建“参数到参数”映射声明了。

创建参数到参数映射声明

本课程说明如何创建指定 UML 参数作为输入/输出对象的映射声明。此“参数到参数”映射声明包含一些映射规则，当运行生成的变换时，这些映射规则将在目标模型中创建一个参数，该参数的 **name**、**visibility** 和 **type** 与子映射调用了此映射声明的元素中参数的 **name**、**visibility** 和 **type** 相同。在本教程中，**Operation2Operation** 映射声明包含调用此映射声明的子映射映射规则。

要创建“参数到参数”映射声明：

1. 如果 **Generalize_Classes.mapping** 文件尚未打开，请在包资源管理器视图中的 **model** 文件夹中，双击该文件。
2. 在变换映射编辑器的“映射根”部分中，右键单击 **Generalize_Classes**；然后单击**创建映射**。
3. 在新建映射窗口的**映射名称**字段中，输入 **Parameter2Parameter**，然后单击**确定**。映射就会显示在大纲视图中，并在编辑器区域中的“映射根”下面打开。

将输入/输出对象添加至 **Parameter2Parameter** 映射声明

在创建映射声明之后，必须向其添加输入对象和输出对象。在本课中，将指定一个 UML 参数作为输入/输出对象。

要将输入对象和输出对象添加至 **Parameter2Parameter** 映射声明：

1. 单击**添加输入对象**图标，即正在编辑的映射的工具栏上最左边的那个图标。
2. 在“添加输入”窗口的“元素”窗格中，展开 **uml**，单击**参数**，然后单击**确定**。
3. 单击**添加输出对象**图标，即正在编辑的映射的工具栏左边的第二个图标。
4. 在“添加输出”窗口的“元素”窗格中，展开 **uml**，单击**参数**，然后单击**确定**。
5. 单击**文件** → **保存**。

现在，可以定义参数输入/输出对象的属性之间的映射规则了。

在 **Parameter2Parameter** 映射声明中定义映射规则

在将输入/输出对象添加至映射声明之后，就可以定义属性之间的映射规则了。

在此部分中，将创建子映射规则和移动映射规则，当运行生成的变换时，这些规则将在目标模型中创建一个参数，该参数的 `name`、`visibility` 和 `type` 与源模型的元素中参数的 `name`、`visibility` 和 `type` 相同。在本教程中，`Operation2Operation` 映射声明调用此映射声明。

要在 `Parameter2Parameter` 映射声明中定义映射规则：

1. 在输入/输出对象的 `name` 属性之间创建移动映射规则：
 - a. 在编辑器区域的参数输入对象中，单击 `name` 属性。
 - b. 将 `name` 属性的句柄拖至输出对象中的 `name` 属性。
2. 重复步骤 1a 和 1b，但将 `name` 替换为 `visibility`。
3. 创建输入/输出对象的类型属性之间的子映射规则：
 - a. 在编辑器区域的参数输入对象中，单击 `type` 属性。
 - b. 将 `type` 属性的句柄拖至输出对象中的 `type` 属性。 因为此属性是复杂类型，所以在缺省情况下将创建子映射。
 - c. 如果属性视图尚未打开，请在编辑器区域中，右键单击新的子映射元素；然后单击在属性中显示。
 - d. 在属性视图中的详细信息选项卡上，从映射列表中选择 **Primitive2Primitive**。

此映射规则将生成代码；当遇到源模型中的基本类型参数时，此代码将调用 `Primitive2Primitive` 变换。如果源模型中的参数不是基本类型，那么不会将参数变换为目标模型。

4. 重复步骤 3a、3b 和 3c，然后完成以下步骤：
 - 在属性视图的详细信息选项卡上，从映射列表中选择 **Class2Class**。

此子映射映射规则将生成代码；如果源参数的类型为类，那么此代码将在目标模型中的元素中创建类型为类的参数。

5. 重复步骤 3a、3b 和 3c，然后完成以下步骤：
 - 在“属性”视图的详细信息选项卡中，从映射列表中选择 **Class2Interface**。

如果源参数的类型为类，那么此子映射映射规则将在目标模型中的元素中创建类型为接口的参数。

6. 单击文件 → 保存。

现在，可以在您在第 5 页的『课程 3：创建和优化类到类映射声明』中创建的“操作到操作”映射声明中定义映射规则了。

在 `Operation2Operation` 映射声明中定义映射规则

第 5 页的『课程 3：创建和优化类到类映射声明』说明了如何创建“操作到操作”映射声明。在本部分中，现在已准备好在此映射声明中创建映射规则。运行生成的变换时，这些映射规则将在目标模型中创建一个操作，该操作的 `name`、`visibility` 和参数与源模型中操作的 `name`、`visibility` 和参数相同。在本教程中，`Class2Class` 和 `Class2Interface` 映射声明调用此映射声明。

要在 `Operation2Operation` 映射声明中定义映射规则：

1. 如果 `Generalize_Classes.mapping` 文件尚未打开，请在包资源管理器视图中的 `model` 文件夹中，双击该文件。
2. 在大纲视图中，单击 **Operation2Operation** 映射声明。
3. 在输入/输出对象的 `name` 属性之间创建移动映射规则：
 - a. 在编辑器区域的操作输入对象中，单击 `name` 属性。
 - b. 将 `name` 属性的句柄拖至输出对象中的 `name` 属性。
4. 重复步骤 3a 和 3b，但将 `name` 替换为 `visibility`。
5. 在输入/输出对象的 `ownedParameter` 属性之间创建子映射规则：

- a. 在编辑器区域的操作输入对象中，单击 `ownedParameter` 属性。
 - b. 将 `ownedParameter` 属性的句柄拖至输出对象中的 `ownedParameter` 属性。 因为此属性是集合，所以在缺省情况下，将创建子映射。
 - c. 如果属性视图尚未打开，请在编辑器区域中，右键单击新的子映射元素；然后单击在属性中显示。
 - d. 在属性视图中详细信息选项卡上，从映射列表中选择 **Parameter2Parameter**。
6. 单击文件 → 保存。

在下一课中，将创建“包到包”映射声明。

课程 6: 创建和优化包到包映射声明

本课程说明如何创建“包到包”映射声明和多个映射规则。映射规则指定生成的变换如何处理源模型包含的包中嵌套的包或类元素。

包输入/输出对象包含名为 `packagedElement` 的特性。此特性是包含不同类型的有效 UML 对象的集合。在本课中创建的映射规则定义变换如何处理类型为包或类的集合元素。

在本课中，将创建以下映射规则：

- 在目标项目中创建包的移动映射规则；此包的名称与源模型中包的名称相同
- 如果元素的类型为类，那么调用 `Class2ClassTransform` 变换的子映射映射规则
- 如果元素的类型为类，那么调用 `Class2InterfaceTransform` 变换的子映射映射规则
- 如果元素的类型为包，那么调用 `Package2PackageTransform` 变换的子映射映射规则

第 5 页的『课程 3: 创建和优化类到类映射声明』说明了对于映射声明中每个移动映射规则，会把一个将属性值从源模型复制到目标模型的规则添加至生成的变换源代码。对于每个子映射映射规则，将在变换源代码中生成抽取指定集合中的元素的抽取器。如果当前输入对象是在映射声明中定义的输入类型的实例，那么会将子映射规则应用于对象。

运行生成的变换时，如果源模型包含包，那么将调用 `Package2PackageTransform` 变换并在目标模型中创建名称与源模型中包的名称相同的包。`Package2PackageTransform` 变换在 `packagedElement` 特性的集合中来回移动。对于类型为包（意味着源模型包含嵌套包）的每个集合元素，变换将调用 `Package2PackageTransform` 变换。对于类型为类的每个集合元素，变换调用规则以将类变换为目标输出模型中相应的类和接口。

要在映射模型中创建“包到包”映射声明：

1. 如果未在变换映射编辑器中打开 `Generalize_Classes.mapping` 文件，请在包资源管理器视图中双击该文件。
2. 在变换映射编辑器的“映射根”部分中，右键单击 **Generalize_Classes**；然后单击**创建映射**。
3. 在新建映射窗口的**映射名称**字段中，输入 `Package2Package`，然后单击**确定**。映射就会显示在大纲视图中，并在编辑器区域中的“映射根”下面打开。

将输入/输出对象添加至包到包映射声明

在创建映射声明之后，必须向其添加输入对象和输出对象。在本课中，将指定一个 UML 包作为输入/输出对象。

要将输入对象和输出对象添加至“包到包”映射声明：

1. 单击**添加输入对象**图标，即正在编辑的映射的工具栏上最左边的那个图标。
2. 在“添加输入”窗口的“元素”窗格中，展开 **uml**，单击**包**，然后单击**确定**。
3. 单击**添加输出对象**图标，即正在编辑的映射的工具栏左边的第二个图标。
4. 在“添加输出”窗口的“元素”窗格中，展开 **uml**，单击**包**，然后单击**确定**。

5. 单击文件 → 保存。

现在，可以定义输入/输出对象的特性之间的映射规则了。

定义包输入/输出对象的属性之间的映射规则

在将包输入/输出对象添加至映射声明之后，可以创建输入/输出对象的属性之间的映射规则。

注：生成变换源代码时，将为每个映射声明生成一个变换。每个生成的变换都包含一个 `getAccept_Condition()` 方法。在缺省情况下，如果当前输入元素是在映射声明中定义的输入对象的实例，那么此方法返回 `true`。如果方法返回 `false`，那么变换不会处理当前输入元素。在本课中，可以创建同时验证输入对象是否为输入类型的实例的输入过滤器。虽然这些输入过滤器是冗余的，但它们也是无害的，仅用于演示如何创建输入过滤器。子映射映射规则中的输入过滤器必须用 Java 编写并且返回布尔值。

要创建包输入/输出对象的属性之间的映射规则：

1. 通过将输入对象的 `name` 属性的句柄拖至输出对象的 `name` 属性，创建 `name` 属性之间的移动映射规则。
2. 创建输入/输出对象的 `packagedElement` 属性之间的子映射映射规则，并指定如果元素的类型为类，那么此子映射调用 `Class2ClassTransform` 变换：
 - a. 将输入对象的 `packagedElement` 属性的句柄拖至输出对象中的 `packagedElement` 属性。因为 `packagedElement` 属性是集合，所以在缺省情况下，映射编辑器将创建子映射映射规则。
 - b. 在编辑器区域中，右键单击新的子映射元素；然后单击在属性中显示。
 - c. 在属性视图的详细信息选项卡上，从映射列表中选择 **Class2Class**。运行生成的变换时，对于输入包中的每个元素，仅当元素的类型为类时，变换才会调用 `Class2ClassTransform` 变换。
3. 可选：在步骤 2 中创建的子映射规则中，创建验证当前对象是否为类的实例的输入过滤器：
 - a. 在编辑器区域中，右键单击在步骤 2 中创建的子映射元素。
 - b. 在属性视图的输入过滤器选项卡上，选择过滤器输入元素复选框。
 - c. 单击直接插入。
 - d. 在方法特征符下面的代码字段下面的文本区域中，指定以下代码：

```
return packagedElement_src instanceof org.eclipse.uml2.uml.Class;
```
 - e. 单击应用。
4. 创建输入/输出对象的 `packagedElement` 特性之间的子映射映射规则，并指定如果 `packagedElement` 集合中当前元素的类型为类，那么名为 `Package2PackageTransform` 的变换将类变换为目标模型中的接口。
 - a. 重复步骤 2a 创建 `packagedElement` 特性之间的另一个子映射。
 - b. 在编辑器区域中，右键单击新的子映射元素；然后单击在属性中显示。
 - c. 在属性视图的详细信息选项卡上，从映射列表中选择 **Class2Interface**。
5. 可选：在步骤 4 中创建的子映射规则中，创建验证当前对象是否为接口的实例的输入过滤器：
 - a. 在编辑器区域中，右键单击在 4 中创建的子映射元素；然后单击在属性中显示。
 - b. 重复步骤 3b、3c、3d 和 3e，指定相同的输入过滤器。
6. 创建处理嵌套包的子映射映射规则。如果 `packagedElement` 集合中的当前元素为包，那么变换将调用 `Package2PackageTransform` 变换。
 - a. 重复步骤 2a 创建 `packagedElement` 特性之间的另一个子映射。
 - b. 在编辑器区域中，右键单击新的子映射元素；然后单击在属性中显示。
 - c. 在属性视图的详细信息选项卡上，从映射列表中选择 **Package2Package**。

7. 可选： 在步骤 6 中创建的子映射规则中，创建验证当前输入元素是否为包的实例的输入过滤器：
 - a. 在编辑器区域中，右键单击在 6 中创建的子映射元素；然后单击**在属性中显示**。
 - b. 重复步骤 3b 和 3c，指定此规则具有输入过滤器。
 - c. 在方法特征符下面的**代码**字段下面的文本区域中，指定以下代码：`return packagedElement_src instanceof org.eclipse.uml2.uml.Package;`
 - d. 单击**应用**。
8. 单击**文件** → **保存**。

映射声明包含 `name` 属性之间的移动映射声明，以及 `packagedElement` 属性之间的多个子映射规则。

课程 7：创建和优化模型到模型映射声明

本课程说明如何在映射模型中创建“模型到模型”映射声明。此映射声明包含一个定制映射规则，当运行生成的变换时，该映射规则创建一个名称派生自源输入模型的目标输出模型。本课程还说明如何更改变换处理映射模型中的映射声明的顺序。

要重命名变换生成的目标模型，可以创建输入/输出对象之间的定制映射规则，而不是创建输入/输出对象的特性之间的映射规则。

要创建“模型到模型”映射声明：

1. 如果尚未在变换映射编辑器中打开映射模型，请在包资源管理器视图中双击 `model` 文件夹中的 `.mapping` 文件。
2. 在变换映射编辑器的“映射根”部分中，右键单击 **Generalize_Classes**；然后单击**创建映射**。
3. 在“新建映射”窗口的**映射名称**字段中，输入 `Model2Model`，然后单击**确定**。映射就会显示在“大纲”视图中，并在编辑器区域中的“映射根”下面打开。

将输入/输出对象添加至模型到模型映射声明

在创建“模型到模型”映射声明之后，必须向其添加输入/输出对象。在本课中，将指定一个模型作为输入/输出对象。

要将输入对象和输出对象添加至“模型到模型”映射声明：

1. 单击**添加输入对象**图标，即正在编辑的映射的工具栏上最左边的那个图标。
2. 在“添加输入”窗口的“元素”窗格中，展开 **uml**，单击**模型**，然后单击**确定**。
3. 单击**添加输出对象**图标，即正在编辑的映射的工具栏左边的第二个图标。
4. 在“添加输出”窗口的“元素”窗格中，展开 **uml**，单击**模型**，然后单击**确定**。
5. 单击**文件** → **保存**。

现在，可以定义输入/输出对象之间的映射规则了。

定义定制和子映射映射规则

在本课中，将创建以下映射规则：

- 定制映射规则，它重命名变换生成的目标模型
- 子映射映射规则，它调用 `Package2PackageTransform` 变换以变换模型中的包

要在映射声明中创建映射规则：

1. 创建重命名目标模型的定制映射规则：

- a. 在编辑器区域中，单击“模型”输入对象最上面的部分。将突出显示整个模型输入对象。
- b. 将“模型”输入对象的句柄拖至“模型”输出对象。
- c. 在新的子映射元素上，单击向下箭头，然后单击定制。
- d. 右键单击定制元素；然后单击在属性中显示。
- e. 在属性视图的详细信息选项卡上，单击直接插入。
- f. 在方法特征符下面的代码字段下面的文本区域中，指定以下代码（此代码在运行变换时实现定制映射规则）：`Model_tgt.setName(Model_src.getName()+"TgtModel");`
- g. 单击应用。

注：要重命名目标模型，还可以创建输入对象和输出对象的 `name` 属性之间的定制映射规则，并在详细信息选项卡上指定相同的代码，而不是创建输入/输出对象之间的映射规则。

2. 创建调用在第 13 页的『课程 6：创建和优化包到包映射声明』中创建的 `Package2Package` 映射的子映射映射规则：
 - a. 将 `packagedElement` 特性的句柄拖至模型输出对象的 `packagedElement` 属性。
 - b. 右键单击新的子映射元素；然后单击在属性中显示。
 - c. 在属性视图的详细信息选项卡上，从映射列表中选择 **Package2Package**。

3. 单击文件 → 保存。

您已经创建了本教程所需的所有映射声明和映射规则。现在，可以指定映射声明的处理顺序了。

更改映射声明的处理顺序

可以更改映射模型中映射声明的处理顺序。大纲视图按运行生成的变换代码时处理映射声明的顺序列示它们。通过更改处理顺序，可以指定可能由不太具体的映射声明处理和使用的输入对象的处理指示信息。

对于本教程，指定生成的变换按以下顺序处理映射声明：

- `Model2Model`
- `Package2Package`
- `Class2Class`
- `Class2Interface`
- `Operation2Operation`
- `Parameter2Parameter`
- `Primitive2Primitive`

注：在本教程中，`Model2Model` 和 `Package2Package` 映射声明的顺序非常重要。运行生成的变换时，因为模型是包，所以 `Package2Package` 变换接受模型或包。如果 `Package2Package` 变换是变换中列示的第一个变换，它将处理并使用模型输入对象。在这种情况下，`Model2Model` 变换不会处理模型输入对象；因此，在 `Model2Model` 变换中创建的定制规则不会运行。

要更改映射声明的处理顺序：

1. 如果大纲视图不可视，请单击窗口 → 显示视图 → 大纲。
2. 在大纲视图中，右键单击 `Model2Model` 映射声明；然后单击执行顺序，再单击上移。重复此步骤，直到 `Model2Model` 映射声明处于列表的顶部。
3. 对每个映射声明重复步骤 2，直到映射声明的顺序如上面的列表所示。
4. 单击文件 → 保存。

现在，可以生成变换源代码了。

模块 2: 创建变换代码和运行变换

在此模块中，将为“模型到模型”变换生成代码并通过在 Eclipse 工作台的单独实例中运行变换来测试变换。

学习目标

本模块中的课程说明如何完成下列任务：

- 生成和编译变换源代码
- 配置运行时工作台
- 创建变换配置
- 在运行时工作台中运行变换代码

所需时间

完成本模块大约需要 20 分钟。

先决条件

您应知道如何完成下列任务：

- 在项目中创建 UML 模型
- 执行基本的 UML 建模

课程 1: 生成和编译变换源代码

本课程说明如何生成和编译变换源代码。

在生成变换源代码之前，在 **Generalize Classes** 项目中，检查 `src` 文件夹中的内容。这些包和文件是在创建项目时生成的；其他文件是在编辑 `Generalize_Classes.mapping` 文件时创建的。

可以增量式添加映射规则以及为映射声明中的映射规则生成实现。不必在生成变换源代码之前定义所有映射规则。例如，在完成本教程之后，您可以添加创建从实现类到接口的实现关系的映射规则。

生成变换源代码时，对于映射模型中的每个映射声明，变换编写框架将生成名为 `nTransform.java` 的 Java 源文件，其中 `n` 表示映射声明的名称。这些 Java 文件一起组成变换代码。除了生成变换的实现代码之外，变换编写框架还生成代码以向变换服务注册变换。

要生成和编译变换源代码：

1. 如果插件开发透视图尚未打开，请将它打开：单击 **窗口** → **打开透视图** → **其他** → **插件开发**；然后单击 **确定**。
2. 如果未在映射编辑器中打开 `Generalize_Classes.mapping` 文件，请在包资源管理器视图中双击该文件。
3. 在映射编辑器区域的“映射根”部分中右键单击；然后单击 **生成变换源代码**。

提示：也可以通过完成下列步骤来生成变换源代码：在包资源管理器视图中，右键单击 `.mapping` 文件；然后单击 **变换** → **生成变换代码**。

在包资源管理器视图中，检查 `src` 文件夹中的包和文件。生成的变换显示在 `src/generalize_classes.transforms` 文件夹中。

4. 要编译生成的源代码，在包资源管理器视图中，单击 **Generalize Classes** 项目，然后单击 **项目** → **构建项目**。在缺省情况下，Eclipse 项目将在保存对项目的更改时自动构建。如果没有更改工作空间或项目的构建首选项，就不必完成此步骤。

提示： 要更改用于构建项目的首选项，单击窗口 → 首选项，展开常规，然后单击工作空间。在更改首选项之后，单击确定。

5. 检查 src 文件夹的内容是否存在错误。
6. 如果在 Class2Interface 变换代码中发生编译错误，那么可能需要导入 VisibilityKind 包：
 - a. 在包资源管理器视图中，在 src/generalize_classes.transforms/Class2InterfaceTransform.java 文件中，双击具有错误修饰符的方法。
 - b. 在编辑器区域的左页边距中，右键单击错误修饰符；然后单击快速修正和双击导入“**VisibilityKind**”（org.eclipse.uml2.uml）。
 - c. 单击文件 → 保存。
7. 在包资源管理器视图中，单击 **Generalize Classes** 项目，然后单击项目 → 构建项目。

复审项目的 src/generalize_classes.transforms 文件夹中的文件。会为映射模型中的每个映射声明生成一个 Java 变换。在 MainTransform.java 文件中，MainTransform 方法会按在“大纲”视图中指定的顺序添加每个生成的变换的实例。

现在，可以配置运行时工作台了。

课程 2：配置运行时工作台

本课程说明如何配置运行时工作台以及如何在运行时工作台创建源和目标 UML 模型。

可以创建和调用运行时工作台来测试和调试变换，这意味着不必在测试之前打包变换插件。

在创建并打开运行时工作台之后，必须创建源和目标模型以测试变换代码。变换将变换在源模型中创建的元素，并在目标模型中生成输出。

要配置运行时工作台：

1. 打开插件开发透视图：单击窗口 → 打开透视图 → 插件开发。
2. 单击运行 → 打开运行对话框。
3. 在运行窗口的左窗格中，单击 **Eclipse** 应用程序，然后单击新建启动配置图标，即工具栏左边的第一个图标。
4. 单击配置选项卡。
5. 单击使用现有的 **config.ini** 文件作为模板，并接受位置字段中的缺省值。这些值指定运行时实例为正在运行的产品的实例，而不是缺省 Eclipse 实例。缺省 Eclipse 运行时实例不能提供测试变换所需的足够功能。
6. 单击应用。
7. 单击运行。

注： 如果工作空间中的项目包含错误，那么会显示列示这些项目的对话框。要继续打开运行时工作台，请单击继续。

将打开新的运行时工作台。

注： 根据可用的系统资源的不同，可能需要花几分钟不等来打开运行时工作台。

在运行时工作台创建测试项目

在配置并打开运行时工作台后，必须创建一个包含下列各项的项目：

- 供变换进行变换的源 UML 模型；此模型包含一个包，该包又包含一个类
- 变换在其中生成输出的空目标模型

要创建包含源和目标模型的项目：

1. 在运行时工作台中，打开建模透视图：单击**窗口** → **打开透视图** → **建模**。
2. 创建名为 TransformationTest 的 UML 建模项目和名为 SourceModel 的 UML 模型：
 - a. 单击**文件** → **新建** → **项目**，展开**建模**，单击 **UML 项目**，然后单击**下一步**。
 - b. 在“创建模型项目”页面上的**项目名称**字段中，输入 TransformationTest。接受其他字段的缺省值，然后单击**下一步**。
 - c. 在“创建模型”页面上，如果尚未选择**常规**，请在“类别”窗格中单击它。
 - d. 在“模板”窗格中，单击**空白模型**。
 - e. 在**文件名字段**中，输入 SourceModel。
 - f. 单击**完成**。
 - g. 如果提示您切换到建模透视图，请单击**是**。
3. 在 TransformationTest 项目中，创建名为 TargetModel 的 UML 模型：
 - a. 在项目资源管理器视图中，右键单击 TransformationTest 项目；然后单击**新建** → **UML 模型**。
 - b. 在“UML 模型”向导的“创建模型”页上，接受缺省值，然后单击**下一步**。
 - c. 在第二个“创建模型”页上，如果尚未选择**常规**，请在“类别”窗格中单击它。
 - d. 在“模板”窗格中，单击**空白模型**。
 - e. 在**文件名字段**中，输入 TargetModel。
 - f. 单击**完成**。

TransformationTarget 项目现在包含生成的变换将变换的源模型以及变换在其中生成输出的目标模型。

4. 在 SourceModel 模型中，创建名为 BusinessClasses 的包；该包又包含名为 Employee 的类；Employee 类含有三个专用操作和一个公用操作：
 - a. 在项目资源管理器视图中，右键单击 **SourceModel** 模型；然后单击**添加 UML** → **包**。
 - b. 将该包命名为 BusinessClasses。
 - c. 右键单击 **BusinessClasses** 包；然后单击**添加 UML** → **类**。
 - d. 将该类命名为 Employee。
 - e. 右键单击 **Employee** 类；然后单击**添加 UML** → **操作**。
 - f. 将该操作命名为 readEmail。
 - g. 在属性视图中的**常规**选项卡上，在**可视性**区域中，单击**专用**。
 - h. 重复步骤 4e 并将新操作命名为 answerPhone。
 - i. 重复步骤 4g 将 answerPhone 操作指定为专用。
 - j. 重复步骤 4e 并将新操作命名为 performWork。
 - k. 重复步骤 4g 将 performWork 操作指定为专用。
 - l. 重复步骤 4e 并将新操作命名为 reportToManager(name:String)。
5. 单击**文件** → **保存**。

现在，可以创建用于运行变换的变换配置了。

课程 3：创建变换配置

本课程说明如何创建变换配置。变换配置指定变换生成预期的输出所需的信息。

可以应用变换配置以将指定的内容变换为目标模型中的不同内容。如果指定的源是模型，那么变换会解释源模型的元素并在指定的目标中生成输出。在大多数情况下，源模型保持不变。变换将生成一组新的文件或者修改一组现有的文件。

为了简化变换配置文件的使用，应将配置文件保存在包含变换的源的项目中。在本课中，将把变换配置文件保存在运行时工作台中的 **TransformationTest** 项目中。变换配置文件的文件扩展名为 **.tc**。

要创建变换配置：

1. 在运行时工作台中，单击**文件** → **新建** → **变换配置**。如果**变换配置**不是菜单项，请单击**文件** → **新建** → **其他** → **变换配置**。
2. 在新建变换配置向导的“名称和变换”页上，指定下列值：
 - a. 在**名称**字段中，输入 **FirstConfiguration**。
 - b. 从转发变换列表中，展开 **Generalize_Classes** 并选择 **Generalize_Classes** 变换。
 - c. 在**配置文件目标**字段中，如果尚未指定此字段，请输入 **/TransformationTest**。
 - d. 单击**下一步**。
3. 在“源和目标”页上，指定变换要进行变换的源模型，以及变换在其中生成输出的目标模型：
 - a. 在“选择的源”窗格中，单击 **TransformationTest** → **Models** → **SourceModel**
 - b. 在“选择的目标”窗格中，单击 **TransformationTest** → **Models** → **TargetModel**
4. 单击**下一步**并接受向导余下各页上的缺省值。
5. 单击**完成**。

这就在运行时工作台中的 **TransformationTest** 项目中创建了变换配置文件，并且该变换配置文件将显示在变换配置编辑器中。现在，可以运行变换了。

课程 4: 运行变换

本课程说明如何在运行时工作台中运行变换。应用变换配置时，将会创建变换的实例，并且变换将采用在变换配置中指定的属性运行。

运行变换时，它会创建一个临时模型，并将此模型与在变换配置中指定的目标模型进行比较。两个模型之间的差别显示在合并窗口中。

有关运行变换和指定合并策略的更多信息，请参阅下面的相关主题。

要应用调用 **Generalize_Classes** 变换的变换配置：

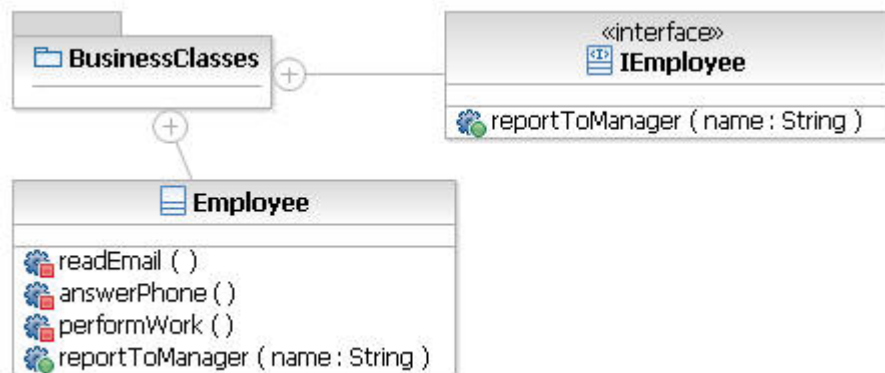
1. 如果该变换配置尚未打开，那么在运行时工作台项目中项目资源管理器视图内的 **TransformationTest** 项目中，双击 **FirstConfiguration.tc**；然后，在变换配置编辑器的首页上，单击**运行**。

提示：在 **TransformationTest** 项目中，还可以右键单击 **FirstConfiguration.tc**；然后单击**变换** → **Generalize Classes** 变换。

2. 当变换运行时，根据变换提供程序指定的缺省合并选项，可能会提示您接受对目标模型中的文件的更改。单击**确定**。
3. 单击**确定**以响应所显示的消息。
4. 在合并窗口中，查看对目标模型的建议更新，选择或清除复选框以接受或拒绝所建议更改，然后单击**确定**。
5. 单击**确定**以响应所显示的消息。

提示：要再次运行此变换，单击**建模** → **变换** → **运行上次运行的**。变换将使用来上次运行变换时使用的源元素。

现在，可以研究 TargetModel 模型中变换输出了。下图显示了所生成的 SourceModelTgtModel 输出模型中元素的可视表示。此模型包含一个名为 BusinessClasses 的包，该包又包含一个名为 Employee 的类和一个名为 IEmployee 的接口。



相关信息:

添加对模型到模型变换映射项目生成的模型的合并支持

运行和重新运行变换

总结：创建、配置和运行模型到模型变换

本教程说明了如何创建、配置和运行简单的模型到模型映射变换。

已学习的课程

本教程解释了模型到模型映射变换的概念并说明了如何完成下列任务:

- 创建包含变换框架和映射模型的模型到模型变换编写项目
- 在映射模型中创建映射声明
- 通过完成下列任务优化映射声明:
 - 指定每个映射声明的源输入对象和目标输出对象
 - 通过定义映射声明中输入/输出对象的属性之间的关系创建映射规则
- 生成和编译变换源代码
- 配置运行时工作台以测试模型到模型变换
- 在运行时工作台中，创建和应用运行模型到模型变换的变换配置

其他资源

要进一步了解本教程中的主题，请考虑以下源:

- 编写变换和变换扩展的联机帮助