Telelogic

Tau®

Java Tutorial

Tau®

Java Tutorial

Copyright Notice

IBM Patents and Licensing

Disclaimer of Warranty

Third-party Trademarks

Contents

Getting Started

Overview

This tutorial teaches you the basics of working with the Tau product in a Java coding environment, and introduces the concepts of requirements analysis and project implementation. In the tutorial, you model a simple application that calculates the population growth projections of senior citizens residing in a city. The growth projections will help the city's planner determine the appropriate size of a new senior center. The calculation example used in this tutorial is based on Fibonacci numbers. For more information on Fibonacci numbers, see **http://en.wikipedia.org/wiki/Fibonacci_number**.

Tutorial Objectives

When you have completed this tutorial, you will have had experience with:

- Using the Tau interface
- Creating a Java project
- Creating a use case diagram
- Creating an activity diagram
- Creating a class diagram
- Generating and editing Java code
- Running and compiling a Java application

Before You Begin

To complete the lessons in this tutorial, you must install the J2SE 5.0 Development Kit from Sun Microsystems. The kit is available at **http://java.sun.com/javase/downloads/index_jdk5.jsp**.

> **Note:** After you install the development kit, be sure to set the PATH system environment variable to the correct path of **javac.exe** in the bin directory of your installation.

Documentation Conventions

This document uses the following conventions:

- **Boldface** for names of GUI objects and controls, including selection choices; and emphasis. Examples:
  - From the **Default model view** drop-down list box, select **Java View**.
  - Click the **Activity flow final** symbol ⊗ on the **Drawing** toolbar.
  - If the Tau browser does not display, select **View** > **Browser**.
  - A project file, called **<project_name>.ttp**.
- `Courier font in 10 point` for pathnames, system messages, and items that you have to type. Examples:
  - The Output window displays the message `Animation session terminated`.
  - In the **Project name** box, replace the default project name with `<project name>`.
  - Type `show` for the function name, and press **Enter**.
- *Italics* for the first mention of a concept with an explanation.

About the Tau Product

Telelogic Tau provides standards-based Model Driven Development™ (MDD™) of complex systems and robust software for enterprise IT applications, including those utilizing Service Oriented Architectures. Tau's iterative requirements-based approach, comprehensive error-checking, and automated simulation increase developer productivity from initial requirements through final documentation and deployment.

Tau offers a large feature set for developers to employ key enabling technologies in a natural, easy-to-use tool environment. Tau makes a seamless and efficient environment for systems, software, and testability. It enables you to perform these tasks:

- **Analyze**, during which you can define, analyze, and validate the system requirements.
- **Design**, during which you can specify and design the architecture.
- **Implement**, during which you can automatically generate code, then build and run within the Tau product.

Tool Tips

This section describes time saving features in Tau that enable you to work more efficiently on a project. Wherever possible, the task-oriented procedures in this tutorial demonstrate the use of these features.

Using Shortcuts

Keyboard shortcuts (**CTRL** + **Arrow Key** for example) allow you to quickly navigate through the Tau interface. For a complete list of available shortcuts, see the Help topic “Editor Shortcuts.”

Using the Auto Placement Feature

You can use Tau’s auto placement feature to quickly add a series of elements in the drawing area. This feature is useful when drawing activity diagrams. For more information on this feature, see the Help topic “Add Symbols.”

Starting Tau

Windows

To start the Tau product in Windows, select **Start** > **Programs** > **Telelogic** > **Telelogic Tau *version number***.

Linux and Solaris

To start the Tau product on Linux and Solaris, type the following command:

```
<installation path>/bin/tau
```

Saving a Project in Tau

To save a project in Tau, on the main menu bar, select **File** > **Save All**. To configure Tau to automatically save changes to your project, follow these steps.

1. On the main menu bar, select **Tools** > **Options**.
2. On the **Save** tab, in the Auto-backup panel, select the **Activate** checkbox and specify an interval (every 5 minutes, for example.)
3. Click **OK**.

Closing Tau

To exit the Tau product, follow these steps.

1. Save your work. Do one of the following:
   - Press **CTRL+S**.
   - Click the Save button to save your work.
2. Choose **File** > **Exit** or click the **Close** button

Note

A red bar on the lefthand side of any file in your workspace indicates that you need to save your work before you exit Tau.

Project Files and Directories

The Tau product creates the following files and subdirectories in the project directory:

- A project file, called **<project_name>.ttp**
- A workspace, called <**project_name>.ttw** in which you can create a project and work on a project.
- Model files, called <**file_name.u2>** that contain the unit files for the project, including UML diagrams, packages, and code generation configurations.

Model Element Names

The names of model elements should follow conventions, such as these:

- Class names begin with an upper case letter, such as “System.”
- Operations and methods begin with lower case letters, such as “restartSystem.”
- Upper case letters to separate concatenated words, such as “checkStatus.”

Model Views

Tau enables you to construct a model in several views, each representing different abstract characteristics of your model. For detailed information on model views, see the Help topic “Views.”

Lesson 1: Creating a Java Project

When you create a Java project in Tau, it contains UML diagrams as well as Java packages, libraries, and add-ins. It also contains the required code generation configurations for the version of Java you are using. Tau creates a directory containing the *project files* in a specified location. The name you choose for your new project is used to name project files and directories, and it appears at the top level of the project hierarchy in the Tau browser.

Goals for this Lesson

In this lesson, you create a new Java project in Tau, configure a project workspace, and add the necessary Java packages for the exercises in this tutorial. You will learn about the following concepts:

- Project configuration settings
- Project workspaces
- Project directories

Exercise 1: Creating a Java Project

In this exercise, you create a new Java project and workspace in Tau.

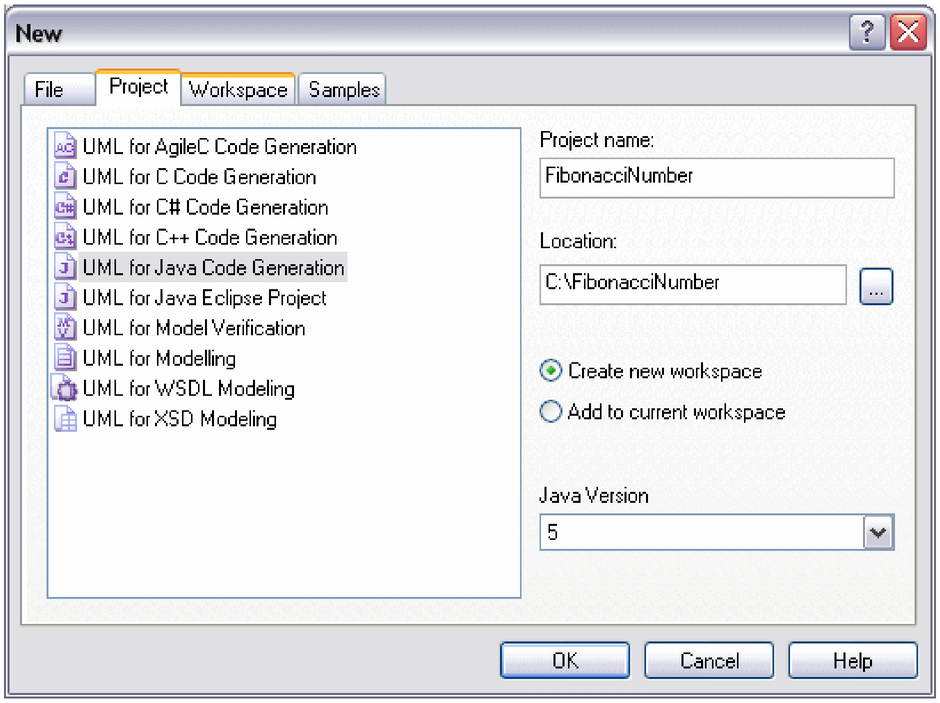
Task 1a: Creating a Java Project

To create a Java project, follow these steps:

1. Click **Start > Programs > Telelogic > Telelogic Tau *version number*.**
2. Do one of the following:

- Press **CTRL+N**, or
- On the main toolbar, select **File > New**, or
- On the Telelogic Tau Welcome page, in the **New Project** panel, click **Proceed**.

3. In the **New** dialog box, in the **Project** tab, choose **UML for Java Code Generation**.
4. In the **Project name** box, type `FibonacciNumber`.
5. In the **Location** box, enter a new directory name or browse to an existing directory.
6. Accept the default option **Create new workspace**.
7. From the Java version drop-down list box, select version **5**.Your dialog box should resemble the following figure.


New
File
Project
Workspace
Samples
UML for AgileC Code Generation
UML for C Code Generation
UML for C# Code Generation
UML for C++ Code Generation
UML for Java Code Generation
UML for Java Eclipse Project
UML for Model Verification
UML for Modelling
UML for WSDL Modeling
UML for XSD Modeling
OK
Cancel
Help

Project name: FibonacciNumber

Location: C:\FibonacciNumber

[✓ Create new workspace] Add to current workspace

Java Version: 5

8. Click **OK**.
9. Accept the Developer Wizard defaults and click **Next**, then click **Finish**. Tau creates a new project and workspace. In the Output window, the following message displays:

```
Java support loading ...

Done.
```

Note: The amount of time required to create the project and load the necessary Java files may vary depending on memory usage and CPU size. In general, this process takes several seconds.

Task 1b: Configuring the Standard Model View

To complete the exercises in this tutorial, you must work with your project in the Standard model view. To make sure you are using the Standard model view, follow these steps.

1. In the Tau browser, select the **.ttp** project file.
2. Do one of the following:
   - Press **ALT+4**.
   - Righ click **Reconfigure Model View.**
   - From the menu bar, select **View** > **Reconfigure Model View**.
3. In the Reconfigure View dialog box, select **Standard View**.
4. Click **OK**.

Exercise 2: Creating Java Packages

In this exercise, you create two Java packages in your workspace. The first package is for analysis and requirements diagrams that provide a high level overview of your application. The second package is for a class diagram. In subsequent lessons, you create a use case diagram and an activity diagram in the analysis package, and a class diagram in the class package. Later in the tutorial, you *export* the class package when you generate Java files from your model.

Note: By default, when you create your project, Tau adds a single Java package with the same name as the project. In the first task of this lesson, you rename the package included with the project. In the second task, you create a second Java package.

Task 2a: Renaming the Default Package

To rename the Java package, follow these steps.

1. In the Model view, select the **FibonacciNumber** package.
2. Press **F2**.
3. Type `Analysis`.

Task 2b: Creating a New Package

To create a new Java package, follow these steps:

1. In the Model view, right-click **Model**, then Select **New Model Element > Package**.
2. In the Create Model Root Element dialog box, in the **Element Name** field, type `Implementation`.
3. Right-click **Implementation**. On the drop-down menu, select **Convert to Java Package**. This will allow us to use this as a Java namespace and generate java code from this package.
4. Ensure that Java syntax is turned on for this package, by Right-clicking the package and making sure **Use Java syntax** has a check mark beside it.
5. On the main menu bar, select **File > Save All**.

Summary

In this lesson, you created a project and added Java packages to the project. You are ready to proceed to the next lesson, in which you begin your project by creating a use case diagram.

Lesson 2: Creating a Use Case Diagram

Use case diagrams show the behavior and capabilities of a system as it interacts with an external user or actor. A use case diagram also shows what a system will do and who will use it.

Goals for this Lesson

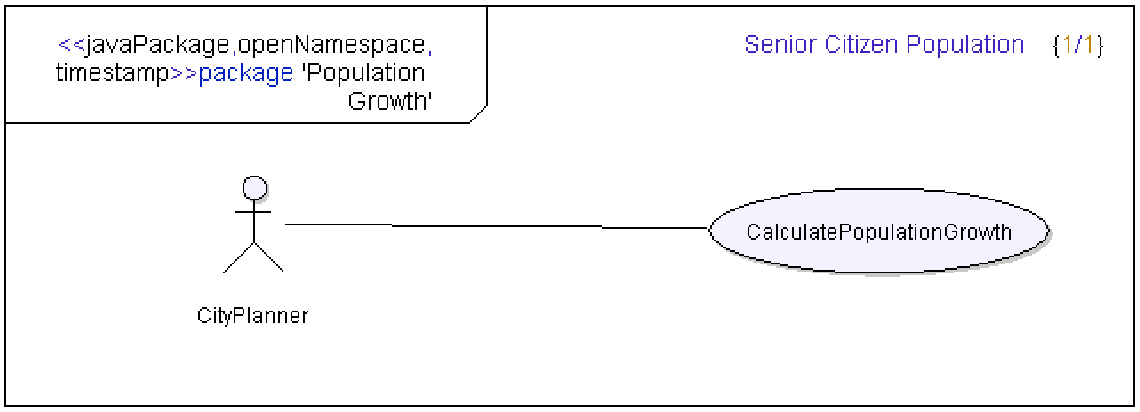
In this lesson, you create a simple use case diagram for the application you are modeling. As you work through this lesson, you will learn about the following elements in use case diagrams, and how to draw them:

- Actors
- Use Cases
- Associations

Exercise 1: Creating a Use Case Diagram

The following figure shows the use case diagram that you create in this exercise.

Use Case Diagram


<<javaPackage,openNamespace, timestamp>>package 'Population Growth'
Senior Citizen Population {1/1}
CityPlanner
CalculatePopulationGrowth

Task 1a: Creating a Use Case Diagram

To create a use case diagram, follow these steps.

1. Right-click **Analysis**.
2. Select **New Diagram > Use Case diagram**. Tau creates the use case diagram in your workspace and opens it in the drawing area.

Task 1b: Renaming a Use Case Diagram

In this task, you use the properties editor to rename the use case diagram. Follow these steps.

1. Right-click in the Model View **Usecase diagram1** and select **Properties**.
2. In the Edit Properties dialog box, in the **Name** field, replace the default name with `Senior Citizen Population`.
3. Click the **Close** button to exit the Properties dialog box.

Task 1c: Adding an Actor and a Use Case

In this task, you add an actor and a use case to the diagram. An *actor* is an external element outside of the system that interacts with the system. A *use case* illustrates the capabilities of a system and shows why a user interacts with the system.

The actor in your diagram is a city planner who is using a system to gather data on the expected population growth of senior citizens in the city. The use case shows the system using a Fibonacci algorithm to compute the growth projections the planner needs to help determine the appropriate size of the new senior center.

To add an actor and a use case to your diagram, follow these steps.

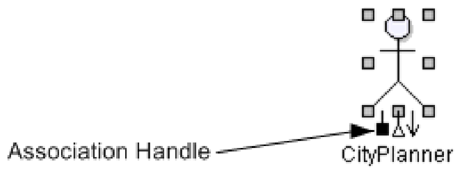
1. Click the actor symbol on the **Drawing** toolbar, then click in the drawing area. Tau adds an actor element in the drawing area.
2. Replace the default name with `CityPlanner`.
3. Click the use case symbol on the **Drawing** toolbar. Tau adds a use case element in the drawing area.
4. Replace the default name with `CalculatePopulationGrowth`.

Task 1d: Adding an Association

An *association line* shows a relationship between two elements in a use case diagram. In this task, you draw an association line that shows the interacting relationship between the city planner and the application use case. You can add an association line using the association symbol on the Drawing Tool menu, or by selecting the association "handle" on the **CityPlanner** element.

To draw an association line using the handle:

1. In the drawing area, select **CityPlanner**.
2. Click on the Association "handle" at the bottom of the **CityPlanner** element as shown in the following figure.



3. Click anywhere inside of the **CalculatePopulationGrowth** use case element. Tau adds an association line that connects the two elements.

To draw an association line using the symbol:

1. Click the **Association** symbol | on the **Drawing** Toolbar.
2. Click the right edge of **CityPlanner** and the left edge of **ComputePopulationGrowth**. Tau adds an association line that connects the two elements.
3. On the main menu bar, select **File > Save All**.

Your drawing should resemble the Use Case Diagram figure.

Summary

In this lesson, you created a use case diagram. You became familiar with the following elements of use case diagrams:

- Actors
- Use cases

- Associations

You are now ready to proceed to the next lesson, in which you create an activity diagram.

Lesson 3: Creating an Activity Diagram

An activity diagram show a particular type of behavior based on a sequence of activities. An activity diagram consists of different activity items connected to each other with arrows. The arrows are used to show the direction of activity flow in the diagram.

Goals for this Lesson

In this lesson, you create an activity diagram. Your activity diagram will show the sequence of activities that occur when the city planner uses the application to estimate the future population growth of senior citizens. The city planner starts the application, then enters a value that the application uses to calculate the population growth. In your activity diagram, the value supplied by the planner represents a number of years in the future. The response from the application is a population growth projection based on the number of years.

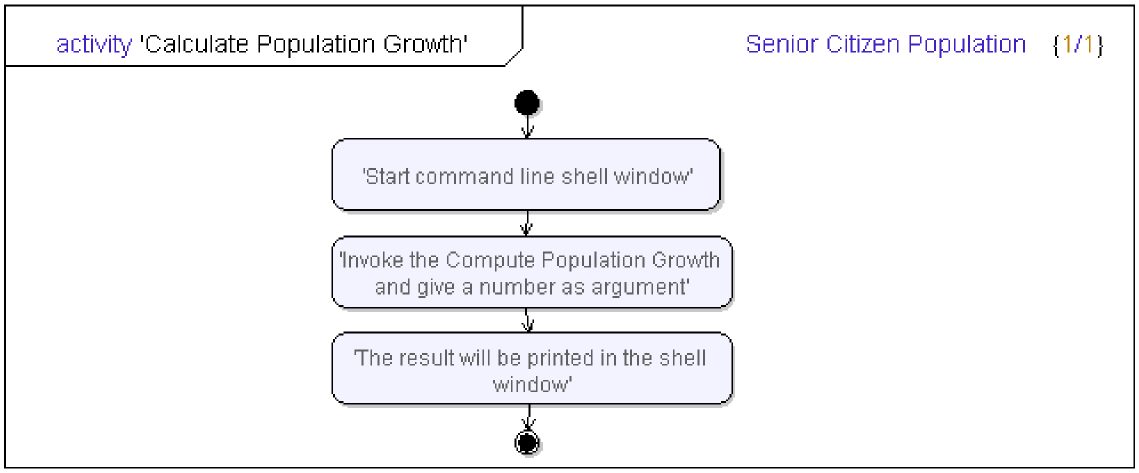
When you create an activity diagram using Tau, you will learn how to draw

- An initial node
- An action node
- An activity line
- A final node

Exercise 1: Creating an Activity Diagram

The following figure shows the Activity Diagram that you create in this exercise.

Activity Diagram


activity 'Calculate Population Growth'
Senior Citizen Population
{1/1}
'Start command line shell window'
'Invoke the Compute Population Growth and give a number as argument'
'The result will be printed in the shell window'

Task 1a: Configuring UML Settings

In this task, you configure UML editing settings so you can draw the elements of your activity diagram as shown in the Activity Diagram figure. By default, Tau is configured to draw the elements of an activity diagram horizontally. Follow these steps to change the setting to vertical so you can draw the elements as shown above.

1. On the menu bar, select **Tools** > **Options**, then select the **UML Advanced Editing** tab.
2. On the **UML Advanced Editing** tab, in the Activity diagrams panel, select **Vertical** from the **Autocreate** drop down list box.
3. Click **OK**.

Task 1b:Creating an Activity Diagram

To create an Activity diagram, follow these steps.

1. In the Tau browser, expand **Model**, then expand the **Analysis** package.
2. Select **New Diagram** > **Activity Diagram**. Tau creates the activity diagram in your workspace and opens it in the drawing area.

3. In the Edit Properties dialog box, in the **Name** field, replace the default name with `Senior Citizen Population`.

4. Click the Save button to save your work.

Task 1c: Drawing Activity Nodes

Nodes show a specific unit of behavior within an activity flow. In this task, you draw an initial node, three action nodes and a final activity node in your diagram. The nodes show the units of behavior that occur when the user starts the application and enters a value that the application uses to produce a population projection figure.

To draw activity nodes, follow these steps:

1. Click the **initial node** symbol on the **Drawing** toolbar. Tau adds an initial node element in the drawing area.

2. Press **Shift-Spacebar** and on the pop-up menu, click the **Activity/action** symbol, repeat this two times to result in three action nodes. Notice that each time you add an action node, Tau automatically includes an *activity flow arrow* between each node. The arrows show the direction of flow in the diagram.

3. Continue to press **Shift-Spacebar** and click the **Activity final** symbol on the **Drawing** toolbar.

4. Click each activity node and type the names as shown in the Activity Diagram figure.

5. On the main menu bar, select **File > Save All**.

You have finished drawing the activity diagram. Your diagram should resemble the Activity Diagram figure.

Summary

In this lesson, you created an activity diagram. You learned about the following elements in an activity diagram:

- Initial Nodes
- Action Nodes
- Final Nodes
- Associations

You are now ready to proceed to the next lesson, in which you create a class diagram.

Lesson 4: Creating a Class Diagram

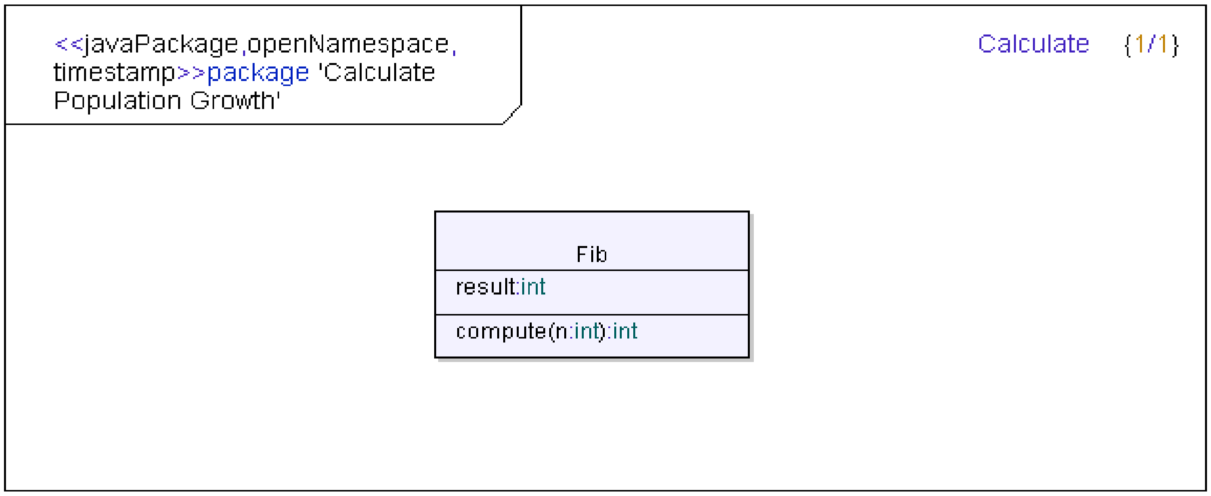
A class diagram shows the types of elements in a system and how they interact and relate to each other. Class relationships are typically shown with dependency or association lines.

Exercise 1: Creating a Class Diagram

In this exercise, you create a class diagram and draw a class in the diagram. The class contains an operation and an attribute for the application you are modeling.

Task 1a: Creating a Class Diagram

The following figure shows the class diagram that you create in this exercise.


<<javaPackage,openNamespace, timestamp>>package 'Calculate Population Growth'
Calculate {1/1}
Fib
result:int
compute(n:int):int

To create a class diagram, follow these steps:

1. In the browser, expand **Model**.
2. Right-click the **Implementation** package and select **New Diagram > Class Diagram**. Tau creates the class diagram in your workspace.

3. In the Edit Properties dialog box, in the **Name** field, replace the default name with `Calculate`.

4. Click the **Close** button to exit the Properties dialog box.

Task 1b: Drawing a Class

To draw a class, follow these steps:

1. Click the Class button on the **Drawing** toolbar, then click anywhere in the drawing area to add the class to the diagram.

2. Replace the default name of the class with `Fib`.

Task 1c: Adding Attributes and Operations

In this task, you add an attribute and an operation to the class you created in the previous task. The *attribute* you add will be the resulting number that is generated when the application performs a computation. The *operation* is the act of computing the number.

To add attributes to the **Fib** class, follow these steps:

1. Select the **Fib** class.

2. Place the cursor the middle panel of the class box and type `result:int` in the attribute text box.

3. Place the cursor in the bottom panel of the class box and type `compute( n:int):int` in the operation text box.

4. On the main menu bar, select **File** > **Save All**.

Summary

In this lesson, you created a class diagram. You learned about the following elements in a class diagram:

- Attributes
- Operations

You are now ready to proceed to the next lesson, in which you will generate Java code from the **Fib** class.

Lesson 5: Generating Java Code and More

Goals for this Lesson

In this lesson, you generate source code from the model you have created, and then you edit the source code. You learn about:

- Generating Java code
- Manually adding code
- Running a Java application

Exercise 1: Generating and Editing Java Code

In this exercise, you generate Java code from the **Fib** element you created in the previous lesson. After you generate the code from **Fib**, you manually add code to the `compute` operation. To generate the Java code, you export the package to a directory that you specify. Tau creates a file in that directory that contains the Java code.

Task 1a: Exporting a Java Package

In this task, you export the **Implementation** package to Java source code. Follow these steps:

1. Select **Implementation** in the Tau browser.
2. From the menu bar, choose **Java > Export > Package**.
3. In the Browse For Folder dialog box, do one of the following:
   - Click **OK** which will create a default folder named **Implementation** in the project directory, or
   - Use the scroll bar to specify an alternate directory, then click **Make New Folder**.

   Tau creates a file named **Fib.java** that contains the Java code generated from the **Implementation** package. Tau also creates a java build artifact named **Artifact0000**.

Task 1b: Adding Code for the compute operation

In this task you add code that is the body of the `compute` operation that you created in Lesson 4: Creating a Class Diagram. To add code manually for the `compute` operation, follow these steps:

1. In the Model browser, double-click the **Calculate** diagram.
2. In the drawing area, right-click **Fib**.
3. Select **Go to Source**.
4. Remove the following lines:

```
int result;
int compute( int n){}
```

5. Replace them with the following code:

```
int result;
int compute( int n ){
   if(n == 0)
      result = 1;
   else if(n == 1)
      result = 2;
else
   result = n + compute(n - 1);
return result;
}
public static void main(String[] argv)
{
Fib myFib = new Fib();
myFib.compute( new Integer(argv
[0]).intValue() );
         System.out.println( "The result is " +
myFib.result );
   }
```

6. On the main menu bar, select **File** > **Save All**.

Task 1c: Viewing Model Updates

In this task, you view the updates that were made to the model when you added the code in the previous task. When you save your work, Tau detects the updated source files in your project and updates your model.

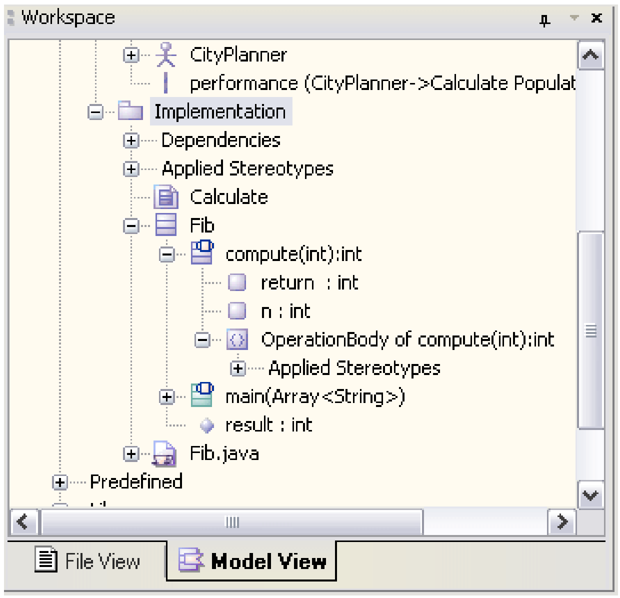
To view the updates you made in the previous task, follow these steps.

1. In the Model view, expand **Model**.

2. Right-click **Artifact0000**. This file is a build artifact that contains the stereotypes needed to build your model.

3. Select **Build (Java)** > **Update**.

4. In the **Implementation** package directory, expand **Fib**. Notice that Tau has added the `operationBody` element to the `compute(int):int` operation, as shown in the following figure. The `operationBody` is the code you added in the previous task.



Task 1d: Compiling Java Code

In this task, you compile the source code. To compile your code, follow these steps.

1. In the model browser, select **Fib**.

2. From the Java menu, select **Compile**.

3. Verify that a successful javac compilation command displays in the **Build** tab of the Output window.

Note

The output displayed in the **Build** tab may vary depending on your operating system.

Task 1e: Running your Application

The application runs an algorithm that computes Fibonacci numbers based on the code you added in the previous task. To run the application, follow these steps.

1. Open a command window and navigate to your project directory.
2. In the project directory, enter the following command:

```
java Implementation/Fib "n"
```

Where `[n]` is an integer representing a number of years in the future.

3. Press **Enter**. The result is total number of seniors expected to be residing in the city in the number of years you specify. For example, if you entered 22, the total number of senior expected to be residing in the city in 22 years would be approximately 254.

Summary

In this lesson, you generated Java code, manually added code, and ran your application. You learned how to

- Export a Java package
- Edit source code
- Test your model

Conclusion

This tutorial has introduced you to UML modeling with Telelogic Tau in a Java environment. By completing the exercises in this tutorial, you have become familiar with the Tau product. You have learned how to:

- Create a project
- Use drawing tools and shortcut keys
- Draw diagrams
- Export a Java package
- Compile code
- Add external code to a model

Your knowledge of how to perform these tasks gives you a basic understanding of Telelogic Tau. You will enhance your skills and product knowledge as you continue to work on UML modeling projects using this product.

Technical Support and Documentation

Contacting IBM Rational Software Support

Support and information for Telelogic products is currently being transitioned from the Telelogic Support site to the IBM Rational Software Support site. During this transition phase, your product support location depends on your customer history.

Product support

- If you are a heritage customer, meaning you were a Telelogic customer prior to November 1, 2008, please visit the Tau Support Web site.
- Telelogic customers will be redirected automatically to the IBM Rational Software Support site after the product information has been migrated.
- If you are a new Rational customer, meaning you did not have Telelogic-licensed products prior to November 1, 2008, please visit the IBM Rational Software Support site.

Before you contact Support, gather the background information that you will need to describe your problem. When describing a problem to an IBM software support specialist, be as specific as possible and include all relevant background information so that the specialist can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, or messages that are related to the problem?
- Can you reproduce the problem? If so, what steps do you take to reproduce it?
- Is there a workaround for the problem? If so, be prepared to describe the workaround.

Other information

For Rational software product news, events, and other information, visit the IBM Rational Software Web site.

Tau Documentation

The Telelogic Tau documentation provides information on most of the topics covered in this tutorial. documentation is available from the following locations:

- From the Start menu, click **Programs > Telelogic > Telelogic Lifecycle Solutions Documentation > Telelogic Tau *version number* > Tau Help**.
- From the Help menu in the Tau interface.

The following table lists the Help topics that provide additional information on key concepts covered in this tutorial.

Help Topic	**Reference Information**
"UML and Java"	General information on UML modeling using Tau in a Java environment.
"Working with Diagrams"	Provides information on creating, saving, and printing diagrams, as well as other common diagram operations.
"UML Language Guide"	Provides a complete list of UML language constructs and model elements

Index