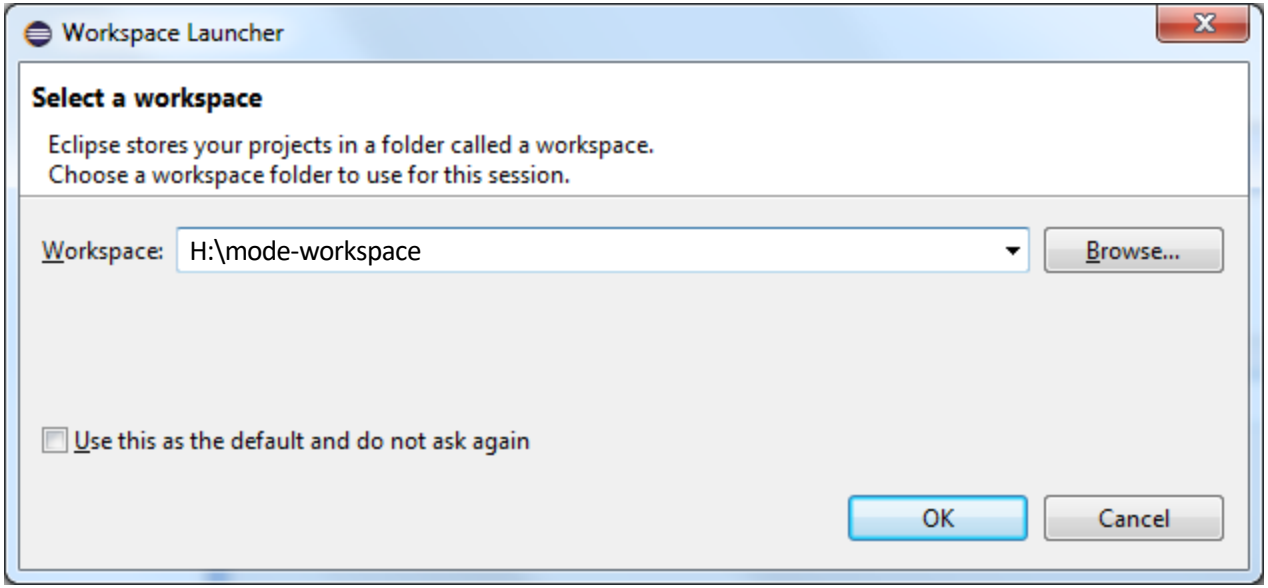
Fundamentals of Metamodelling with Ecore/Emfatic

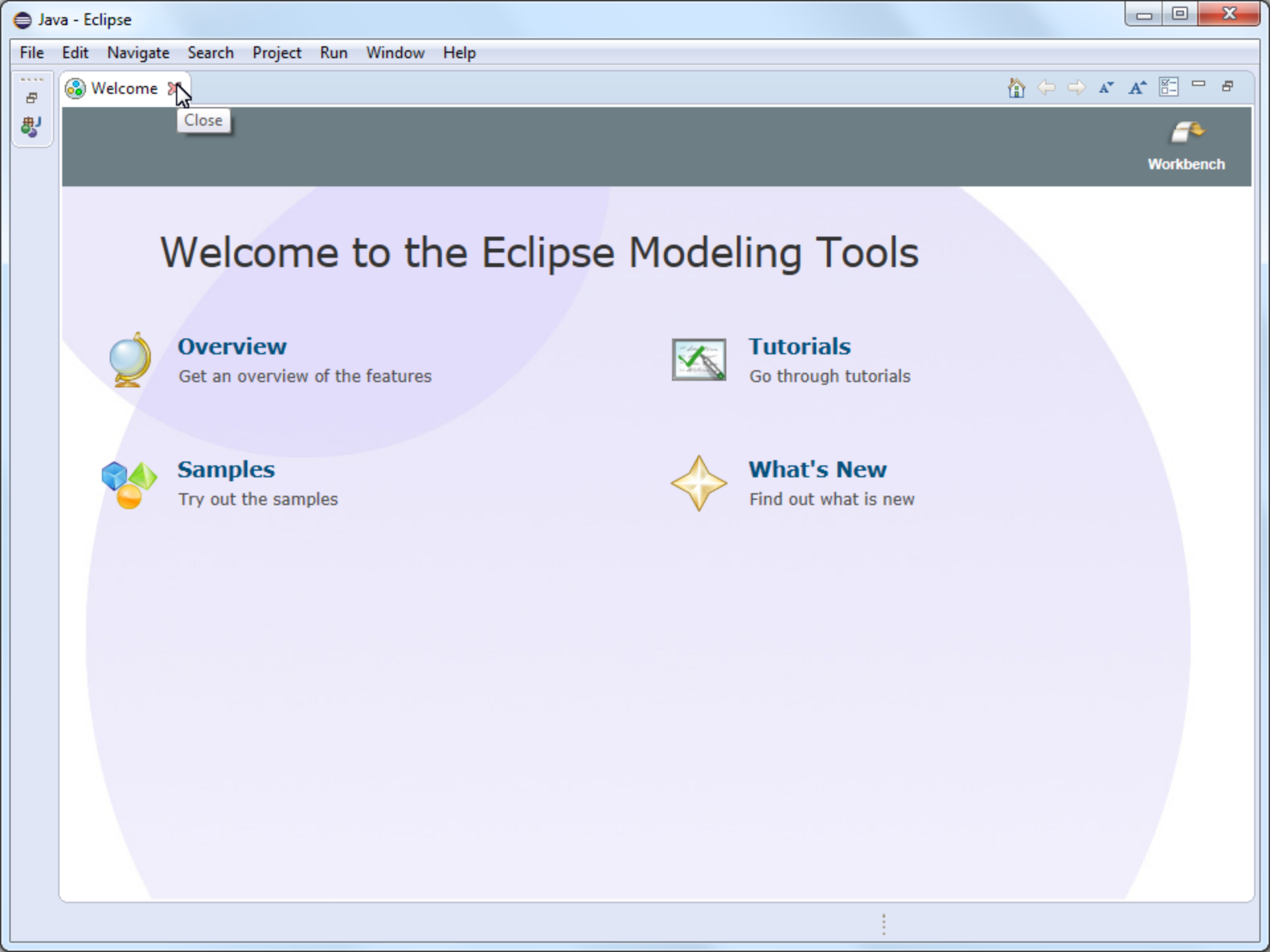
Homework Assignment 2

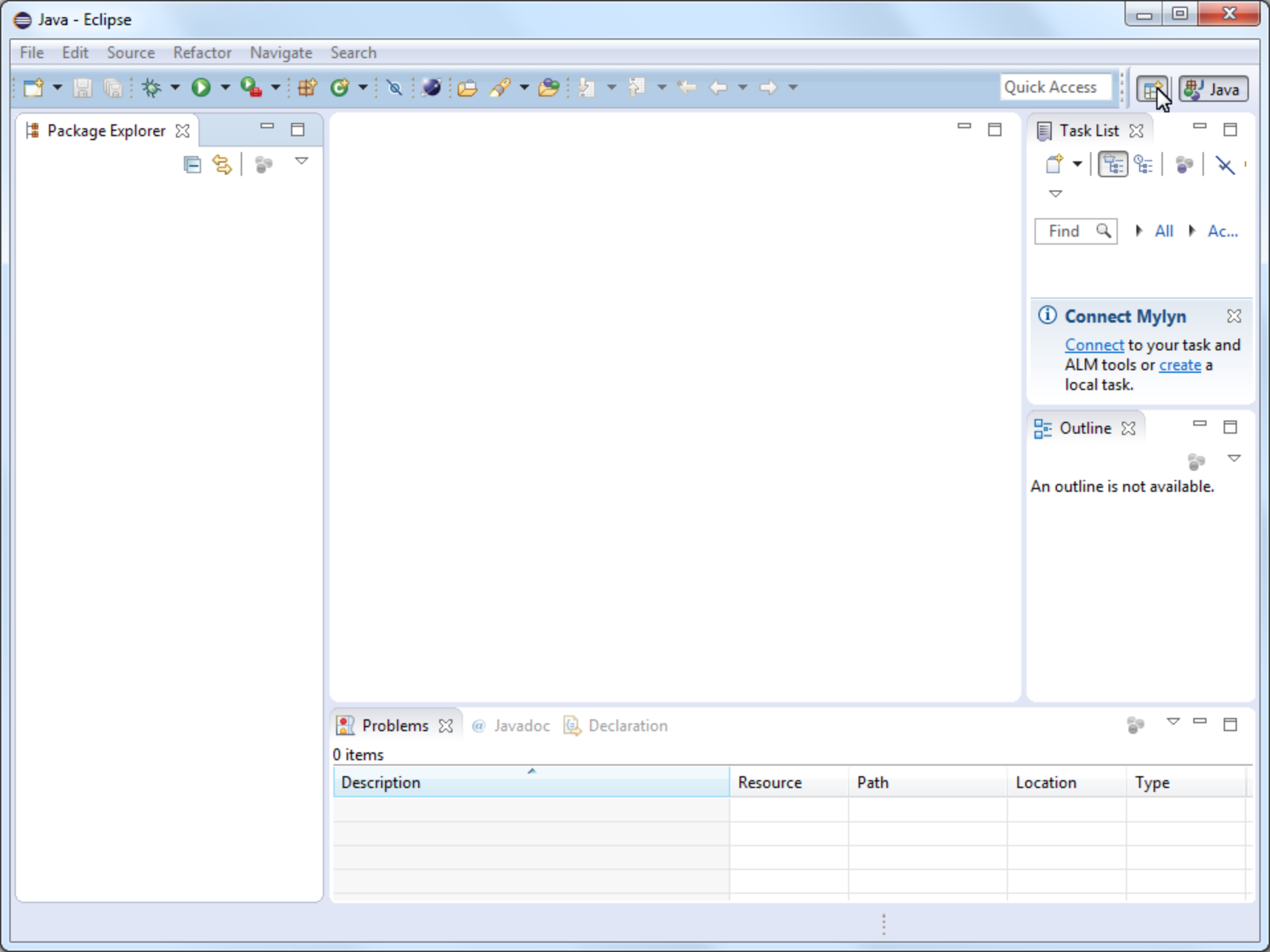
Due 27 February

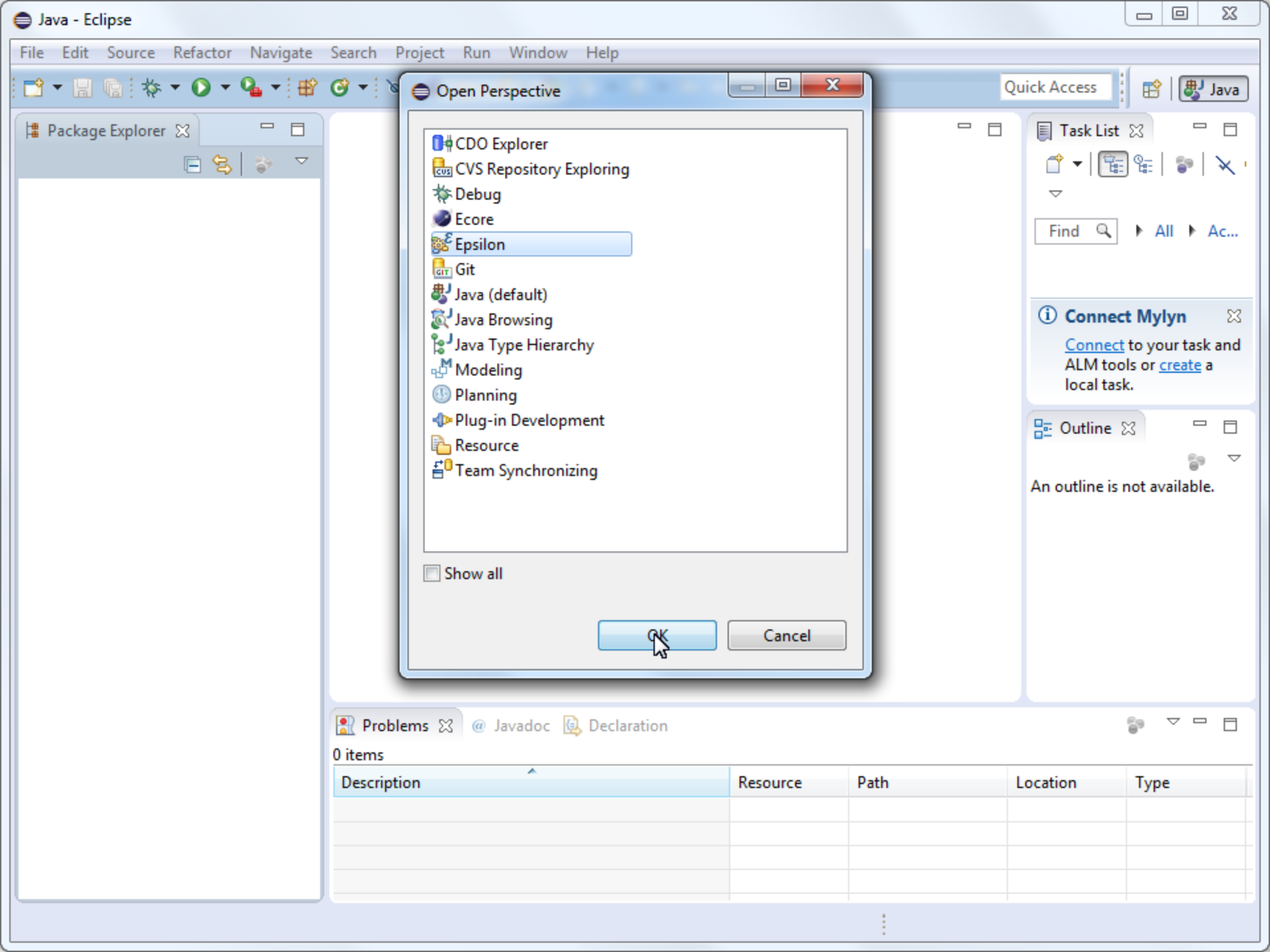
Introduction

- In this assignment you will develop metamodels (using Emfatic), and models that conform to them
  - Reminder: Emfatic is a textual syntax for EMF's metamodelling language (Ecore)
- You need to
  - Download and install an Eclipse Epsilon bundle from http://www.eclipse.org/epsilon/download/
  - And make sure to update your installation.


Workspace Launcher
Select a workspace
Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.
Workspace:
H:\mode-workspace
Browse...
Use this as the default and do not ask again
OK
Cancel


Java - Eclipse
File Edit Navigate Search Project Run Window Help
Welcome
Close
Workbench
Welcome to the Eclipse Modeling Tools
Overview
Get an overview of the features
Tutorials
Go through tutorials
Samples
Try out the samples
What's New
Find out what is new


Java - Eclipse
File
Edit
Source
Refactor
Navigate
Search
Quick Access
Java
Package Explorer
Task List
Find
All
Ac...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type


Java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
Java
Package Explorer
Open Perspective
CDO Explorer
CVS Repository Exploring
Debug
Ecore
Epsilon
Git
Java (default)
Java Browsing
Java Type Hierarchy
Modeling
Planning
Plug-in Development
Resource
Team Synchronizing
Show all
OK
Cancel
Task List
Find
All
Ac...
Connect Mylyn
Connect to your task and ALM tools or create a local task.
Outline
An outline is not available.
Problems
Javadoc
Declaration
0 items
Description
Resource
Path
Location
Type

Getting Started

- The tutorial in the URL below demonstrates
  - creating a library metamodel using Emfatic
  - creating a minimal model that conforms to the library metamodel
  - http://www.eclipse.org/epsilon/doc/articles/reflective-emf-tutorial/
- **Exercise:** Try to reproduce the steps of the tutorial above in your Eclipse installation

From Pen & Paper to Eclipse

- In the previous homework assignment you created three metamodels (DSLs), using pen and paper, for
  - Conference organisation
  - Software distribution
  - Research proposal design
- In this practical you will **implement these metamodels** in Eclipse using Emfatic and you will **create sample models** that conform to them

Metamodelling with Emfatic/Ecore

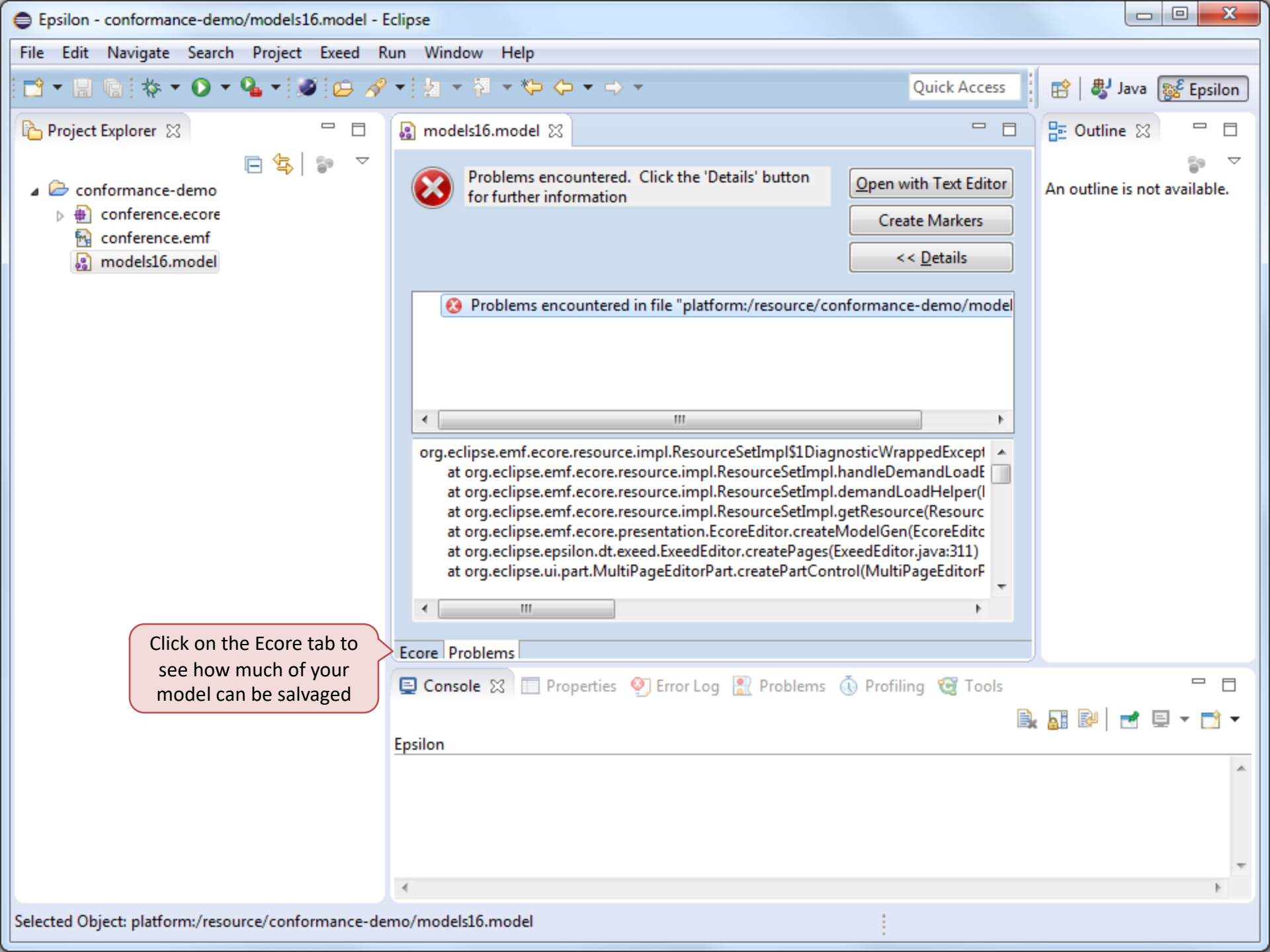
- **Exercise:** For each metamodel from Assignment 1
  - Create a new General -> Project in your Eclipse workspace
  - Create a new .emf file in that project and use it to specify your metamodel in Emfatic
    - Emfatic language reference: http://www.eclipse.org/epsilon/doc/articles/emfatic/
  - Create a model that exercises all the types/attributes/references in your metamodel
- All these steps were covered in the library tutorial
  - More tips in the following two slides

The Metamodelling Process

- Metamodelling is an iterative process
  - You start by constructing an initial version of your metamodel
  - You try to create a small model that conforms to it
    - … and typically you realise that your metamodel is incomplete/wrong
  - You update your metamodel accordingly
    - ... and go back to the previous step until you're happy with it

Breaking and Fixing Metamodel Conformance

- Modifying your metamodel may break conformance with existing models
  - e.g. when you rename/delete a class/reference/attribute
  - Adding classes/references/attributes is unlikely to introduce conformance problems
- If a model does not fully conform to its metamodel, EMF **cannot load it** (see next slide)
- In such a case you need to
  - Delete your non-conforming model and start again or
  - Right-click on your .model file, select *Open with -> Text Editor* and see if you can fix the XMI so that it conforms to the new metamodel
    - To open your model with the tree-based editor you need to close the text editor and right-click on it and select *Open with -> Exeed*


Epsilon - conformance-demo/models16.model - Eclipse
File Edit Navigate Search Project Exeed Run Window Help
Quick Access
Java
Epsilon
Project Explorer
conformance-demo
conference.ecore
conference.emf
models16.model
models16.model
Problems encountered. Click the 'Details' button for further information
Open with Text Editor
Create Markers
<< Details
Problems encountered in file "platform:/resource/conformance-demo/model
org.eclipse.emf.ecore.resource.impl.ResourceSetImpl$1DiagnosticWrappedExcept
at org.eclipse.emf.ecore.resource.impl.ResourceSetImpl.handleDemandLoadE
at org.eclipse.emf.ecore.resource.impl.ResourceSetImpl.demandLoadHelper(I
at org.eclipse.emf.ecore.resource.impl.ResourceSetImpl.getResource(Resourc
at org.eclipse.emf.ecore.presentation.EcoreEditor.createModelGen(EcoreEditc
at org.eclipse.epsilon.dt.exeed.ExeedEditor.createPages(ExeedEditor.java:311)
at org.eclipse.ui.part.MultiPageEditorPart.createPartControl(MultiPageEditorP
Ecore
Problems
Outline
An outline is not available.
Click on the Ecore tab to see how much of your model can be salvaged
Console
Properties
Error Log
Problems
Profiling
Tools
Epsilon
Selected Object: platform:/resource/conformance-demo/models16.model

Complex Constraints

- Models often need to satisfy constraints that cannot be expressed using Emfatic/Ecore
  - e.g. in the Research Proposal DSL, the end month of a task should always be greater than its start month
- Soon, you will learn how to specify and check such constraints using an executable language (EVL), but for now…
- **Exercise:** Write down as many constraints for each metamodel as you can think of in plain English

What to submit?

- Zip up your .emf files and email them to Richard.
- Embed any documentation (comments) you might like in your .emf files.

BONUS: EDITOR GENERATION

Editor Generation

- Generate dedicated tree-based editors for your metamodels as demonstrated in lecture
- Replace the default icons of your editor
  - Icons under <dsl>.edit/icons/full/obj16
  - You will need to find appropriate 16x16 icons
    - e.g. in http://iconarchive.com
- Customise the labels of model elements on the tree editor by modifying the getText(…) methods of <dsl>.edit/src/<dsl>.provider/<type>ItemProvider classes
  - Don't forget to set them to @generated NOT

BONUS: REFLECTIVE EDITOR CUSTOMIZATION

Reflective Editor Customisation

- You can also customise the appearance of the reflective tree-based editor using appropriate annotations
- Use the annotations in https://www.eclipse.org/epsilon/doc/articles/exeed-reference/ to customise the appearance of your model

BONUS: HUTN

HUTN

- HUTN is a generic JSON-like textual syntax for models that can be used as a replacement for XMI
- Follow the tutorial below, and then try to encode some of your models in HUTN
  - http://www.eclipse.org/epsilon/doc/articles/hutn-basic/