Engenharia de Domínio
Mestrado em Engenharia Informática
PL 10
Textual concrete syntaxes with EMFText
Exercise 5: EMFText

Alexandre Bragança, Isabel Azevedo
atb@isep.ipp.pt, ifp@isep.ipp.pt

Dep. de Engenharia Informática – ISEP

2017/2018

Acknowledgement

This lecture is mainly based on the contents of the EMFText guide, which is available in the EMFText web site at
`http://www.emftext.org/index.php/EMFText_Documentation`

After successful instalation of EMFText in Eclipse you can also access the same contents using the option "Help/Help Contents" and navigating to "EMFText User Guide"

There is also a simple tutorial available at `http://sites.google.com/site/luismiguelpedro/curriculum-vitae/technical-reports/emftextReport.pdf`.

Context

- **EMFText** is a tool that can be used to implement **textual concrete syntaxes for modelling languages**.
- Then, textual editors, tighty integrated with Eclipse, can be generated for those languages.

Overview

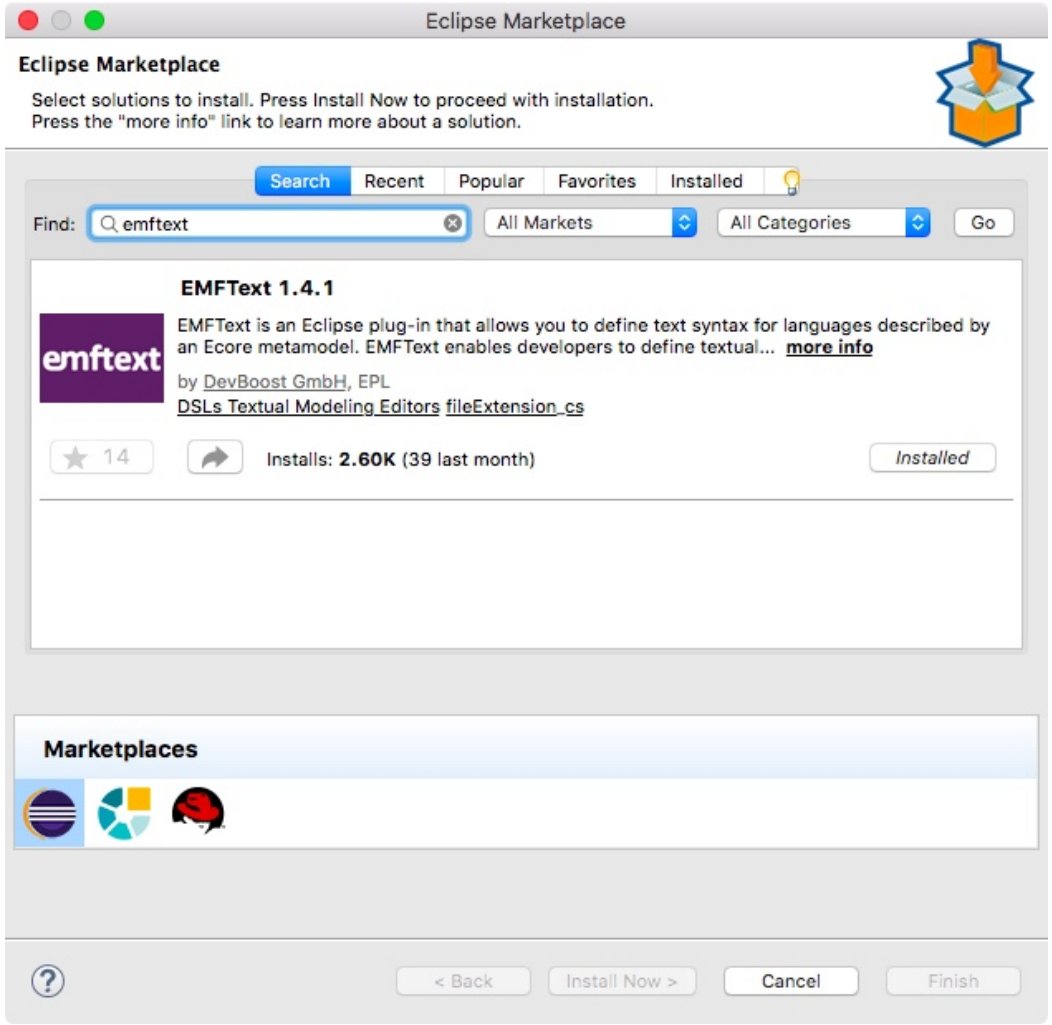
- **Tutorial**: We will see how to use EMFText to design a concrete textual language for our **mindmap** metamodel and then use the generated editor to create and edit instances in Eclipse.
- **Exercise 5**: You should accomplish a similar goal for the **requirements** metamodel for exercise 5.

Tutorial

- You will find in the repository `https://bitbucket.org/mei-isep/ex5emftext` the base project for this tutorial.
- This tutorial exemplifies how to develop a textual concrete syntax for the **mindmap** metamodel.

Setup EMFText in Eclipse

First we need to install EMFText in Eclipse. We can use the option "Help/Eclipse Marketplace..." in Eclipse and search for "emftext" as depicted in the Figure



Create a Project for Using EMFText

The essencial part in an EMFText project is **the specification of the concrete textual syntax for an ecore metamodel**. This is done in a file with the "**.cs**" extension.

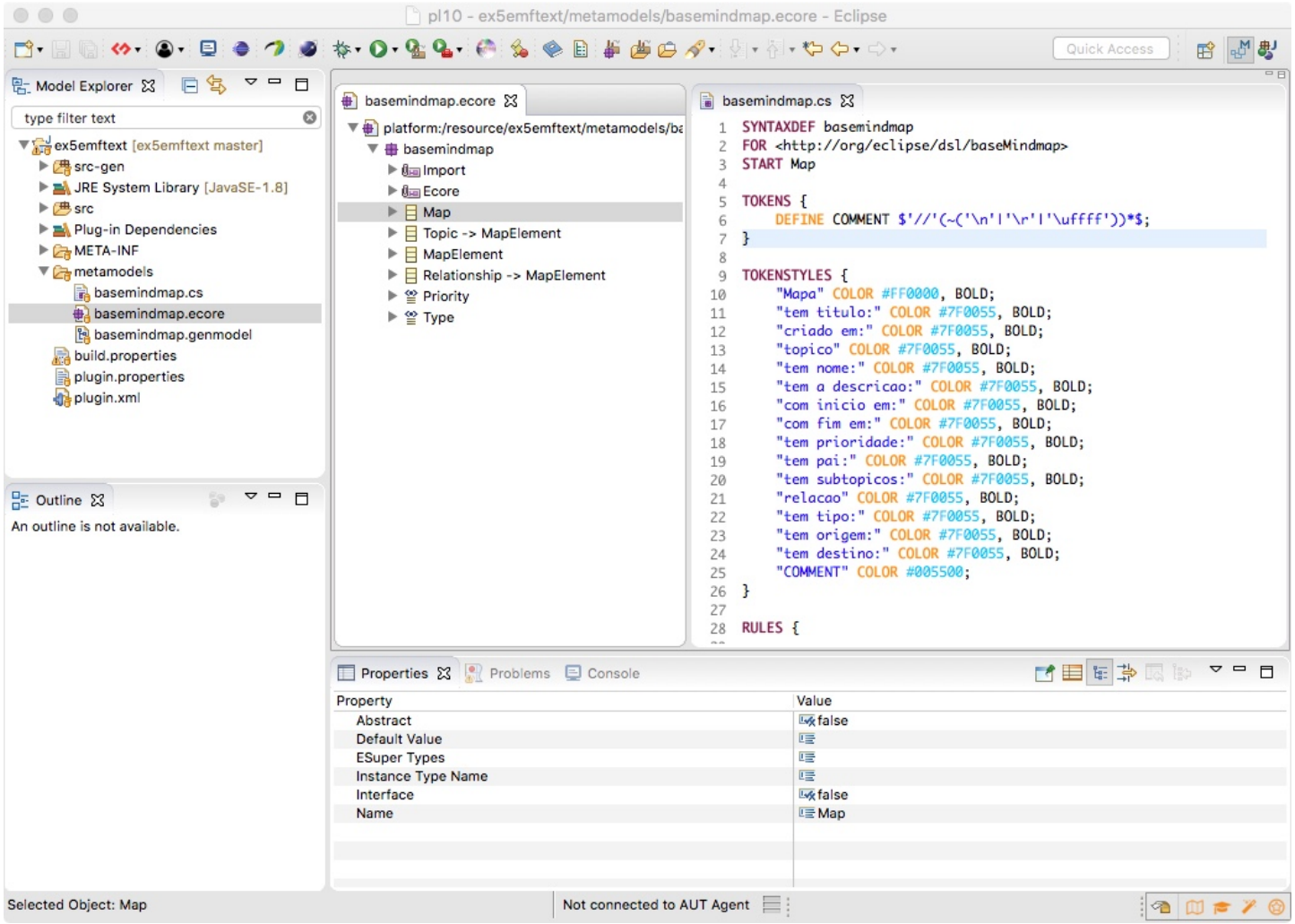
For that we can follow any of the following options:

1. **Create an EMFText Project** using "File/New/Other..." and entering "EMFText" and selecting the "EMFText Project" option.
   - A new window should be displayed to enter the specifics of the new project, such as is name. Several projects should be created to support the new sample DSL (for illustrative proposes).
2. **Create an Empty Java Project** using "File/New/Other..." and entering "java" and selecting the "Java Project" option.
   - A new window should be displayed to enter the specifics of the new java project. You should create an "empty" java project.
   - You should them create a new "metamodel" folder in the new project to add the metamodel files (.ecore and .genmodel) and the new concrete syntax file (.cs).

Note: You should download and use the prepared project for this tutorial that is available at `https://bitbucket.org/mei-isep/ex5emftext`. Therefore **you do not need to create a new project**, use the project you dowloaded from the previous url.

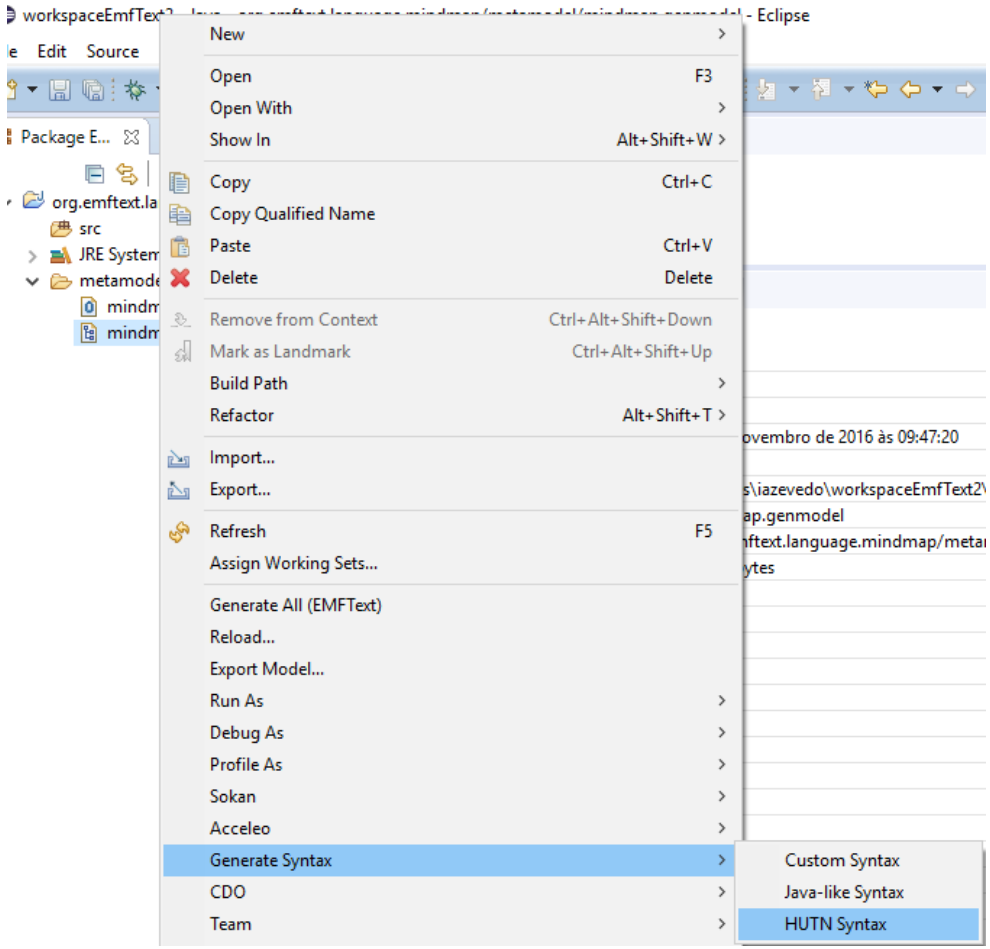
The ex5emftext Project

In this project you will find: the **basemindmap.ecore**, the **basemindmap.genmodel** and the **basemindamap.cs** for the concrete textual syntax



Automatically Generating Concrete Textual Syntaxes

In this project we already have a concrete syntax file for our basemindmap metamodel. However, if we did not have it, it is possible to automatically generate concrete textual syntaxes for existing metamodels by using the "Generate Syntax" option that is available for ecore files, as depicted in the Figure.



The cs file

Elements of a concrete syntax specification file :

- Header that can include file extension, meta model namespace URI, location and the start element(s), among others
- Options
- Token definitions
- Syntax rules

basemindmap.cs (1/2)

```
1  SYNTAXDEF basemindmap
2  FOR <http://org/eclipse/dsl/baseMindmap>
3  START Map
4
5  TOKENS {
6    DEFINE COMMENT $'//'(~('\n'|'\r'|'\uffff'))*$;
7  }
8
9  TOKENSTYLES {
10   "Mapa" COLOR #FF0000, BOLD;
11   "tem titulo:" COLOR #7F0055, BOLD;
12   "criado em:" COLOR #7F0055, BOLD;
13   "topico" COLOR #7F0055, BOLD;
14   "tem nome:" COLOR #7F0055, BOLD;
15   "tem a descricao:" COLOR #7F0055, BOLD;
16   "com inicio em:" COLOR #7F0055, BOLD;
17   "com fim em:" COLOR #7F0055, BOLD;
18   "tem prioridade:" COLOR #7F0055, BOLD;
19   "tem pai:" COLOR #7F0055, BOLD;
20   "tem subtopicos:" COLOR #7F0055, BOLD;
21   "relacao" COLOR #7F0055, BOLD;
22   "tem tipo:" COLOR #7F0055, BOLD;
23   "tem origem:" COLOR #7F0055, BOLD;
24   "tem destino:" COLOR #7F0055, BOLD;
25   "COMMENT" COLOR #005500;
26 }
```

Listing 1: basemindmap.cs

basemindmap.cs (2/2)

```
28  RULES {
29
30    @SuppressWarnings(minOccurenceMismatch, maxOccurenceMismatch)
31    Map ::=  "Mapa" "{" (
32      "tem titulo:" title['"',','"']
33      ("criado em:" created[])?
34      ("tem elemento:" elements : Topic, Relationship )*
35    ) "}";
36
37    @SuppressWarnings(minOccurenceMismatch, maxOccurenceMismatch)
38    Topic ::= "topico" "{"
39      "tem nome:" name['"',','"']
40      ("tem a descricao:" description['"',','"'])?
41      ("com inicio em:" start[])?
42      ("com fim em:" end[])?
43      ("tem prioridade:" priority[HIGH:"ALTA", MEDIUM:"MEDIA", LOW:"BAIXA"])?
44      ("tem pai:" parent[])?
45      ("tem subtopicos:" subtopics[])*
46    "}";
47
48    @SuppressWarnings(minOccurenceMismatch, maxOccurenceMismatch)
49    Relationship ::= "relacao" "{"
50      "tem nome:" name['"',','"']
51      "tem tipo:" type[DEPENDENCY:"DEPENDENCIA", INCLUDE:"INCLUSAO", EXTEND:"EXTENSAO"]
52      "tem origem:" source[]
53      "tem destino:" target[]
54    "}";
55  }
```

Listing 2: basemindmap.cs

Notes

Our project is ready for the generation of the concrete syntax supporting code

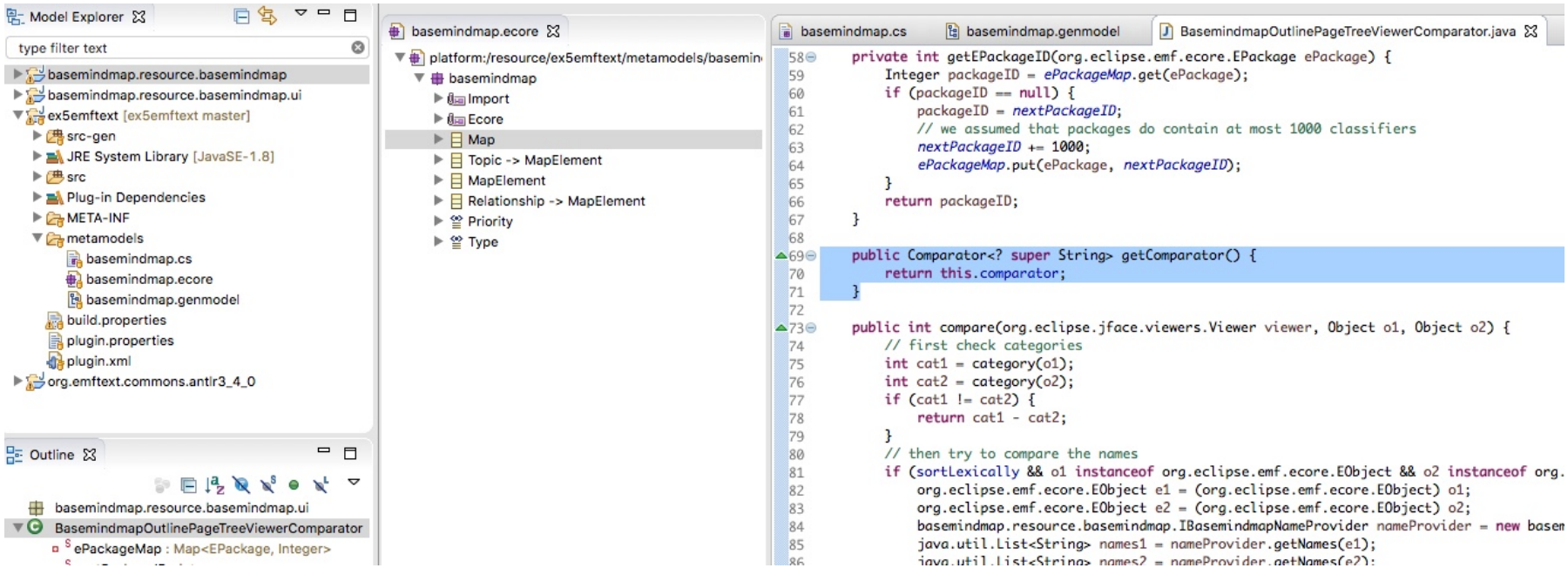
Before doing that you should make sure that:

1. EMFText is able to access the basemindmap metamodel. You should right click in the file and select "Register EPackages" or "Register Metamodel".
   - If the metamodel is not properly registered in Eclipse EMFText is not able to located it and can not "validate" the cs file.
2. Generate the model code for our metamodel. Open the basemindmap.genmodel file and right click the "Basemindmap" package. Select "Generate Model Code".

Generating Concrete Syntax Supporting Code

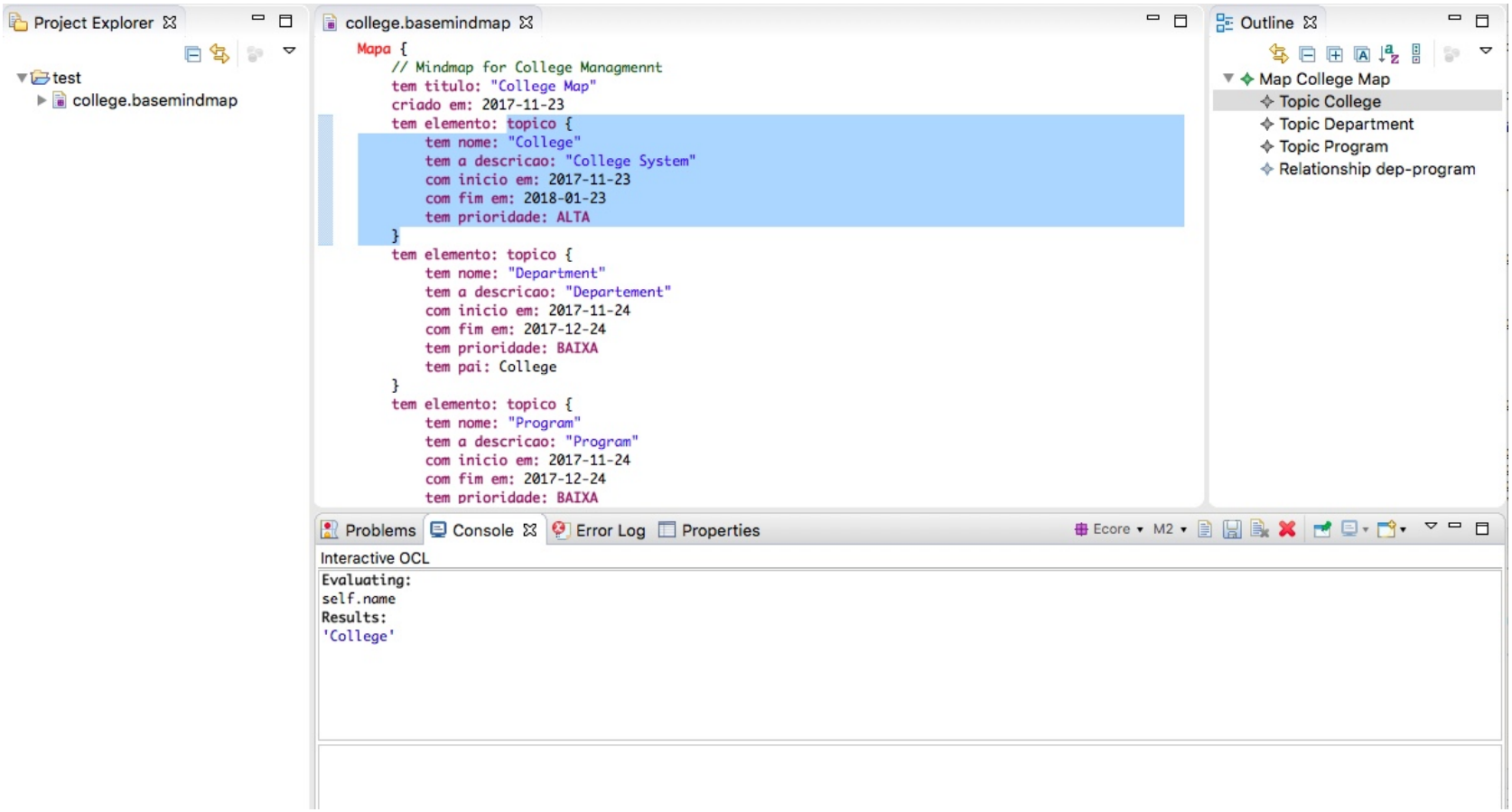
- Right click in the basemindmap.cs file and select "**Generate Text Resource**".

Note: Correct the error in a method of `...ui` package by applying the fix that eclipse suggests (**Change return type of 'getComparator(...)' to 'Comparator<? super String>'**)



Running Eclipse with the new Plugin (1/2)

- The generated code adds a new textual editor for files of type (i.e., extension) **basemindmap**.
- To activate this new code we need to setup a new Eclipse Run Configuration of type "Eclipse Application"to run a second Eclipse instance where this new code will be active.



Running Eclipse with the new Plugin (2/2)

- Eclipse now supports the edition of a new file type named "EMFText .basemindmap file".
  - Files with a ".basemindmap"extension should be automatically opened by the new editor.
- This file is persisted in a textual format.
- When opened in Eclipse these files can be used in the same way as other EMF instance files.
  - For instance, you can use the OCL console to query the model, as depicted in the previous slide.
  - We could also use it with other EMF related tools, like ATL or Acceleo.
- **How to Obtain an xmi version of the file.** If we want to have an xmi version of the file simple use "File/Save As..." and select ".xmi" as the extension. The result should be an xmi file with the same contents as the basemindmap file.
  - You should also be able to convert a model between different textual concrete syntaxes using the same approach (obviously they should share the same metamodel).

Exercise 5

Exercise 5: Goals

1. Using EMFText, add a **textual concrete syntax to the requirements metamodel**.
   - **Alternative A: Verbose Portuguese Syntax.** You should add a verbose Portuguese syntax for the requirements metamodel, similar to the example presented for mindmap.
   - **Alternative B: Compact English Syntax.** You should add a compact English syntax for the requirements metamodel. These syntax should require a minimum of keywords.
2. Using Acceleo and PlantUML[1], **generate Use Case diagrams for the functional requirements included in requirements models**.
   - Try to use in your Acceleo transformations "textual instances" as opposed to using "xmi" instances.

Review

- Your review should focus on how alternatives of goal 1 compare.

[1] See `http://plantuml.com`.