Software Language Engineering - Ecse 439, Group#9
Warehouse Management System
April 22, 2021
Yoan Poulmarc’k-260773983
Ege Odaci-260722818
Jamie Li-260725145

1)Work Distribution

After deciding our topic of choice for the project, we all worked together on designing the UML of our application in draw.io. Then Yoan worked on the Xtext specification. Jamie and Ege worked on the Acceleo at first then later Jamie shifted more into Acceleo as Ege focused on the report.

2)Xtext

2.1)Description of DSL

Our domain-specific language, Warehouse Management, is oriented at managing the warehouses of companies and their respective employees. Employees can either be a part of a team, or manage one as a manager. Though Warehouse Management does not simply track the employees current employment, but also their history in the company. Warehouse Management also manages an employee’s work and time, respectively in the form of tasks and availabilities.

Warehouse Management follows a very simple list-esque syntax. Simply name a company, followed by it’s warehouses (which contains employees) then it’s teams and finally the tasks and who’s working on them. Simply repeat the process for more companies. This syntax was designed with the thought of grouping based on dependency. For example an employee should only care about their own warehouse which is why employees are declared inside a warehouse, while teams can have employees and it’s manager from different warehouses which is why teams are declared at the same level as warehouses.

2.2)How to run language editor

Instructions:

- Go to org.xtext.example.wh project
- Right click src/org.xtext.example.wh/WH.xtext
- Click on ‘Run as’, then ‘Generate Xtext Artifacts’
- Right click project
- Click on ‘Run as’ then ‘Eclipse project’
- A new eclipse window should pop up
- Create a new Java project
- Copy example1.wh and example2.wh from /acceleo/model to the new project
- In order to generate the XMI for acceleo from the models, update the paths inside xtext/org.xtext.example.wh project/src/org.xtext.example.wh/Main.java to point to your current .wh models. Run that file as java application to generate their XMI version, or use the ones we have already generated in acceleo/model/

2.3)List of changed files

- xtext/org.xtext.example.wh project/src/org.xtext.example.wh/WH.xtext
- xtext/org.xtext.example.wh project/src/org.xtext.example.wh/Main.java (Used for generating the XMI version of our model
- acceleo/model/example1.wh (Simple model to test)
- acceleo/model/example2.wh (More complex model to test)

2.4)Metamodel figure

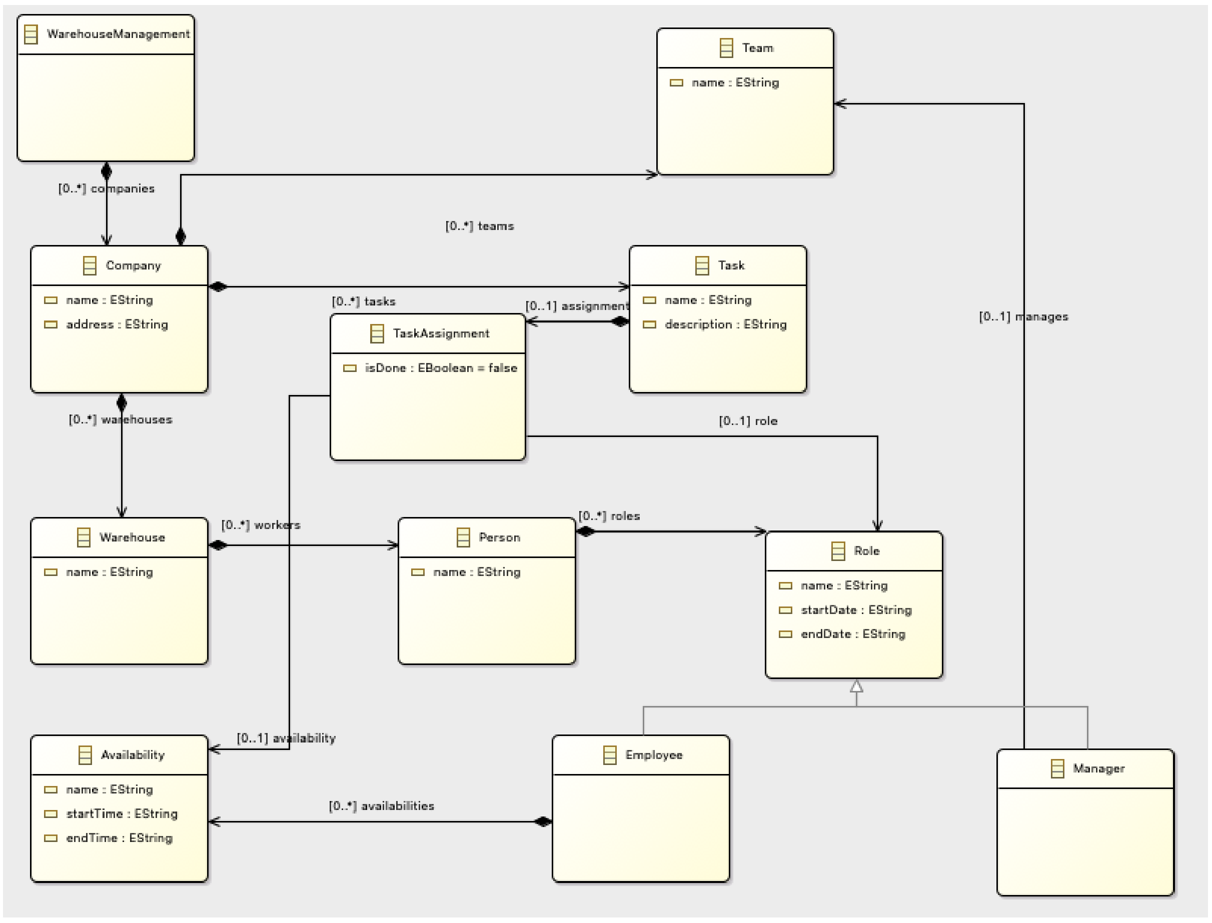


Figure-1

2.5)Discussion of key metamodel design decision

We used the User-Role pattern in order to be able to track not only employees' current position but also their history at the company. We also split up an employees time at the company further down with Availability which lets us track during what time period an employee worked on a task and what was their position at the time.

3)Acceleo

3.1)Description of target language

We used acceleo for our model to text transformation. Our target language of choice was java. We used an XMI file produced using Xtext as an input to acceleo to receive our desired transformation. Also, we have implemented a java class that consists of a with the java package generated by Xtext to implement the Java class.

3.2)Mapping from source to target

Our mapping done in the acceleo can be seen in the table below with their XMI and their Java file.

XMI	Java

`<wH:WarehouseManagement xmi:version="2.0"`
`...`
`</wH:WarehouseManagement>`
	`WHFactory factory = WHFactory.eINSTANCE;`
	`WarehouseManagement aWhm = factory.createWarehouseManagement();`

`<companies name="CoolCompanyInc" address="3000 Rue Coolio, QC H8T 3P8">`
`...`
`</companies>`
	`Company CoolCompanyInc = factory.createCompany();`
	`CoolCompanyInc.setName("CoolCompanyInc");`
	`CoolCompanyInc.setAddress("3000 Rue Coolio, QC H8T 3P8");`
	`aWhm.getCompanies().add(CoolCompanyInc);`

`<warehouses name="Warehouse10020">`
`...`
`</warehouses>`
	`Warehouse Warehouse10020 = factory.createWarehouse();`
	`Warehouse10020.setName("Warehouse10020");`
	`CoolCompanyInc.getWarehouses().add(Warehouse10020);`

`<workers name="p1">`
`...`
`</workers>`
	`Person p1 = factory.createPerson();`
	`Warehouse10020.getWorkers().add(p1);`
	`p1.setName("p1");`

`<roles xsi:type="wH:Manager" name="m1" startDate="13/11/1999" endDate="31/01/2022" manages="//@companies.0/@teams.0"/>`	`Manager m1 = factory.createManager();`
	`p1.getRoles().add(m1);`

`<availabilities name="av2" startTime="31/01/2022" endTime="23/02/2020"/>`	`Availability av2 = factory.createAvailability();`
	`av2.setId("av2") ;`
	`av2.setStartTime("31/01/2022");`
	`av2.setEndTime("23/02/2020") ;`

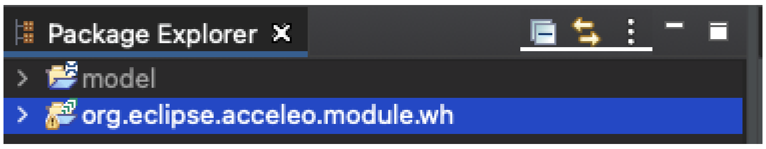
`<teams name="TeamB1"/>`	`Team TeamB1 = factory.createTeam();`
	`CoolCompanyInc.getTeams().add(TeamB1);`
	`TeamB1.setName("TeamB1");`

`<tasks name="t1" description="this is a random task"/>`	`Task t1 = factory.createTask();`
	`CoolCompanyInc.getTasks().add(t1);`
	`t1.setId("t1");`
	`t1.setDescription("this is a random task");`

3.3)How to run transformation

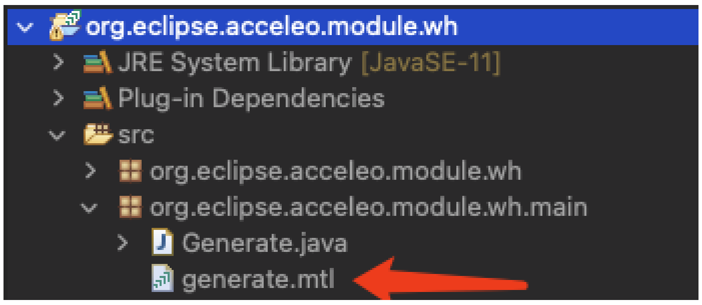
Instructions:

- in the Xtext project, right click on the `` `org.xtext.example.wh` `` and `` `run as Eclipse Application` ``
- Once we have the **second** instance of eclipse opened, import two projects in the \`acceleo\` folder, **\`model\`** and **\`org.eclipse.acceleo.module.wh\`** separately as \`existing project\`. Then we will have the following project structure



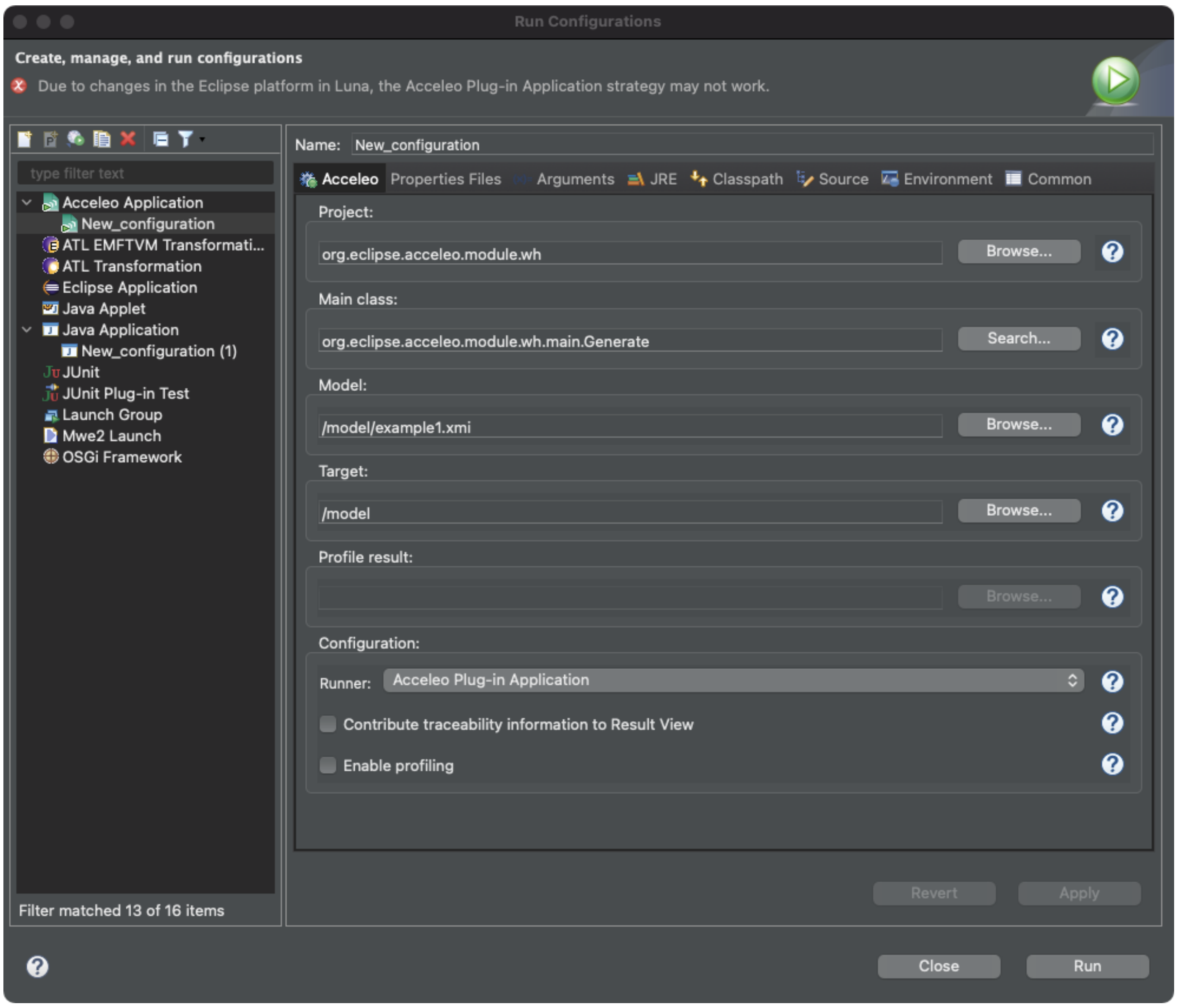
Package Explorer
model
org.eclipse.acceleo.module.wh

- Go to folder
    - **org.eclipse.acceleo.module.wh/src/org/eclipse/acceleo/module/wh/main/generate.mtl**



org.eclipse.acceleo.module.wh
JRE System Library [JavaSE-11]
Plug-in Dependencies
src
org.eclipse.acceleo.module.wh
org.eclipse.acceleo.module.wh.main
Generate.java
generate.mtl

- Right click on **generate.mtl**, select \`run configuration\` and do the following setup


Run Configurations

Create, manage, and run configurations

Due to changes in the Eclipse platform in Luna, the Acceleo Plug-in Application strategy may not work.

Name: New_configuration

Project: org.eclipse.acceleo.module.wh

Main class: org.eclipse.acceleo.module.wh.main.Generate

Model: /model/example1.xmi

Target: /model

Profile result:

Configuration:

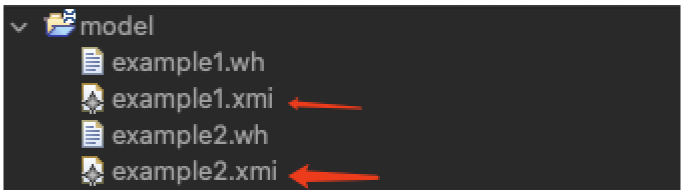
Runner: Acceleo Plug-in Application

Contribute traceability information to Result View

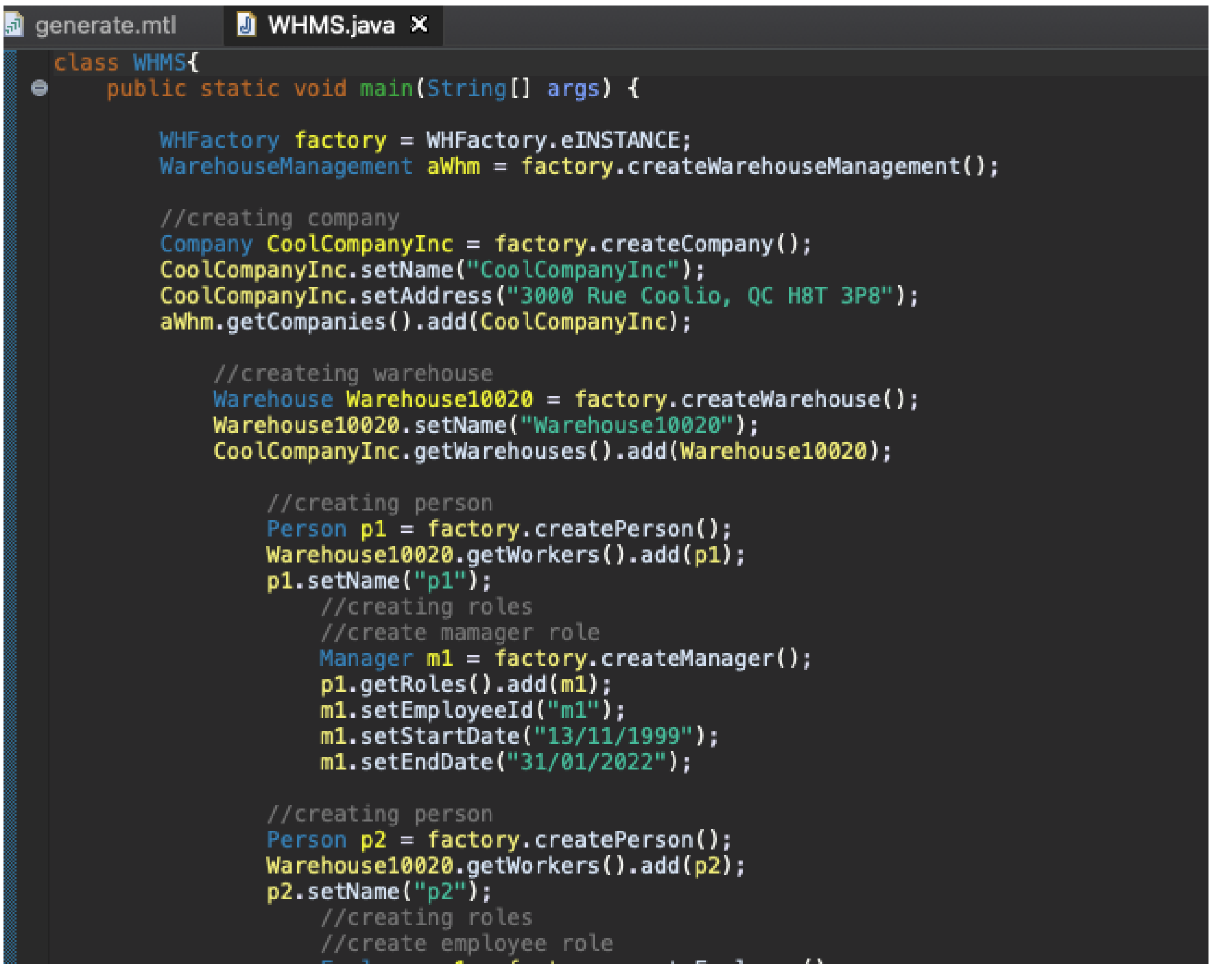
Enable profiling

Filter matched 13 of 16 items

- For model, you can try two xmi files(example1.xmi, example2.xmi) in the `model` project



- For the target, feel free to pick your desired output folder
- **change the Runner to `Acceleo plug-in Application`**
- click run, you should see a generated **WHMS.java** file in your target folder
  (There is one WHMS.java in the `model` folder, which is generated by example1.xmi)

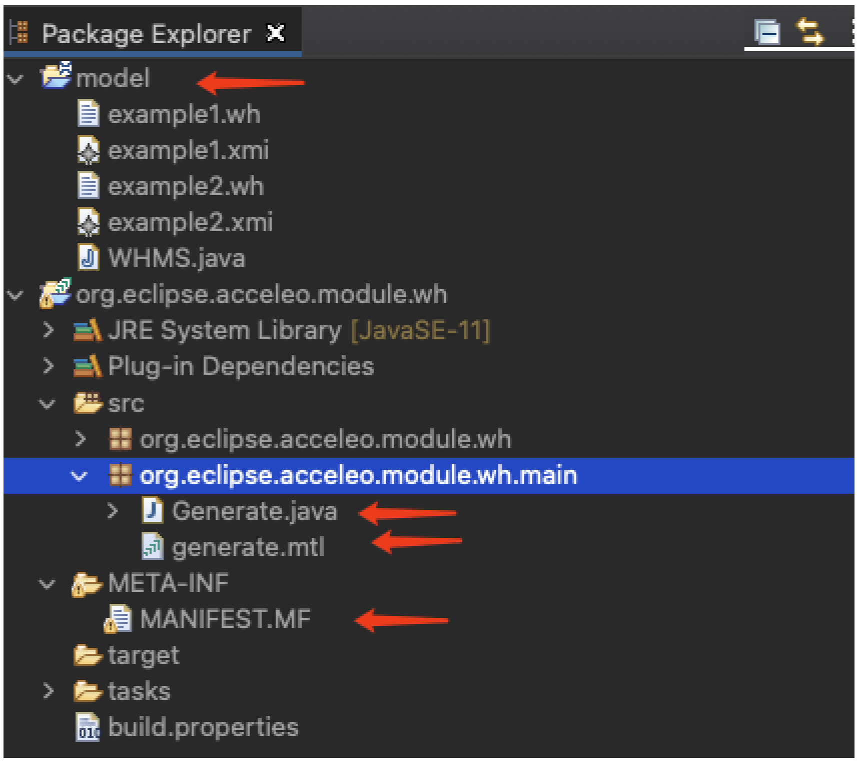


```
generate.mtl    WHMS.java
```

```java
class WHMS{
    public static void main(String[] args) {

        WHFactory factory = WHFactory.eINSTANCE;
        WarehouseManagement aWhm = factory.createWarehouseManagement();

        //creating company
        Company CoolCompanyInc = factory.createCompany();
        CoolCompanyInc.setName("CoolCompanyInc");
        CoolCompanyInc.setAddress("3000 Rue Coolio, QC H8T 3P8");
        aWhm.getCompanies().add(CoolCompanyInc);

            //createing warehouse
            Warehouse Warehouse10020 = factory.createWarehouse();
            Warehouse10020.setName("Warehouse10020");
            CoolCompanyInc.getWarehouses().add(Warehouse10020);

                //creating person
                Person p1 = factory.createPerson();
                Warehouse10020.getWorkers().add(p1);
                p1.setName("p1");
                    //creating roles
                    //create mamager role
                    Manager m1 = factory.createManager();
                    p1.getRoles().add(m1);
                    m1.setEmployeeId("m1");
                    m1.setStartDate("13/11/1999");
                    m1.setEndDate("31/01/2022");

                //creating person
                Person p2 = factory.createPerson();
                Warehouse10020.getWorkers().add(p2);
                p2.setName("p2");
                    //creating roles
                    //create employee role
```

Figure-2(generated java code by acceleo)

3.4)List of changed files



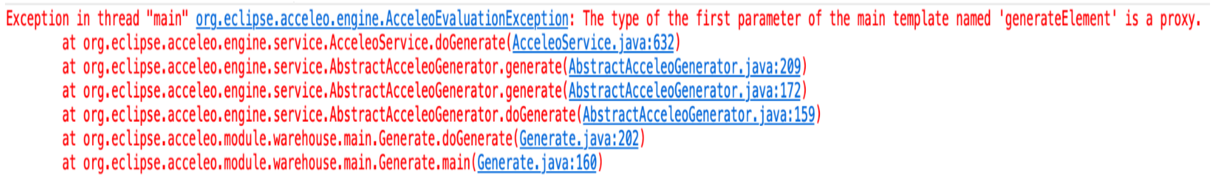
```
Package Explorer
model
    example1.wh
    example1.xmi
    example2.wh
    example2.xmi
    WHMS.java
org.eclipse.acceleo.module.wh
    JRE System Library [JavaSE-11]
    Plug-in Dependencies
    src
        org.eclipse.acceleo.module.wh
        org.eclipse.acceleo.module.wh.main
            Generate.java
            generate.mtl
    META-INF
        MANIFEST.MF
    target
    tasks
    build.properties
```

- \`model\` is where we store the input xmi files and output java file
- There is one line added in **Generate.java:registerPackages()** to register our package
- **META-INF** is edited to include the /bin path
- **\`generate.mtl\`** is where Model to text transformation code is written

4)Discussion

4.1)What did not go well?

At the beginning we have had some issues regarding the setup of the project. Acceleo setup was very much challenging. Although two of our team members focused on the Acceleo each of us experienced different types of errors. Our error example can be seen in the figure below in figure



```
Exception in thread "main" org.eclipse.acceleo.engine.AcceleoEvaluationException: The type of the first parameter of the main template named 'generateElement' is a proxy.
	at org.eclipse.acceleo.engine.service.AcceleoService.doGenerate(AcceleoService.java:632)
	at org.eclipse.acceleo.engine.service.AbstractAcceleoGenerator.generate(AbstractAcceleoGenerator.java:209)
	at org.eclipse.acceleo.engine.service.AbstractAcceleoGenerator.generate(AbstractAcceleoGenerator.java:172)
	at org.eclipse.acceleo.engine.service.AbstractAcceleoGenerator.doGenerate(AbstractAcceleoGenerator.java:159)
	at org.eclipse.acceleo.module.warehouse.main.Generate.doGenerate(Generate.java:202)
	at org.eclipse.acceleo.module.warehouse.main.Generate.main(Generate.java:160)
```

Figure-3

After investing in a significant amount of time, we have finally fixed our setup issue by launching a second instance of an eclipse and doing the acceleo part in that eclipse.. Another issue was that acceleo gave some errors due to having some issues related to xtext transformation and they were hard to understand what the real issue was.

About the Xtext and Acceleo documentation was not very detailed thus we had to do many trial and error in order to figure things out.Another issue we had was that our xtext created an ecore file although we needed an XMI file for the acceleo. We have followed the stackoverflow link provided to us however it wasn't that easy to figure out. After many times of trying it, we finally were able to transform our ecore file into an XMI file.

4.2)What went well?

We knew how our UML diagram would look like at the very beginning. It was very easy to draw the diagram. Xtext was intuitive and not particularly challenging. Sadly anything past the basics became a lot harder as documentation online was very sparse and hard to find. Regarding the acceleo part, we knew what we were supposed to do, except for the setup part which I mentioned in the previous section. We have had a fun time transforming our model into java using acceleo as it felt very natural. Acceleo transformations required a lot of coordination with our modeling thus we needed a lot of zoom meetings in order to coordinate things and help each other understand the overall project.