COP5339 - OOP

COP5339 - Object Oriented Programming

Written by:

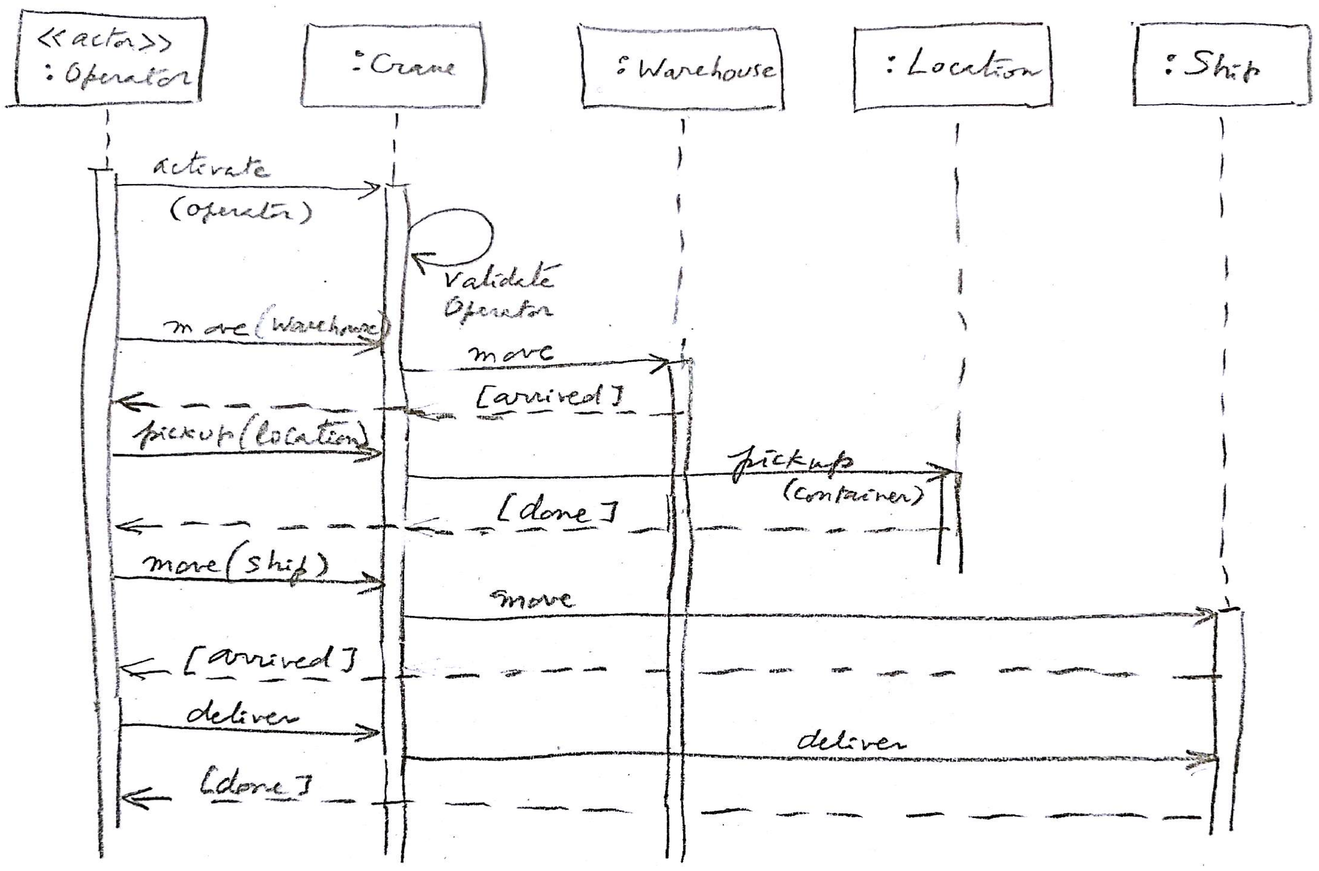
Christopher Foley
Z15092976

Academic Year: 2016-2017

Table of Contents

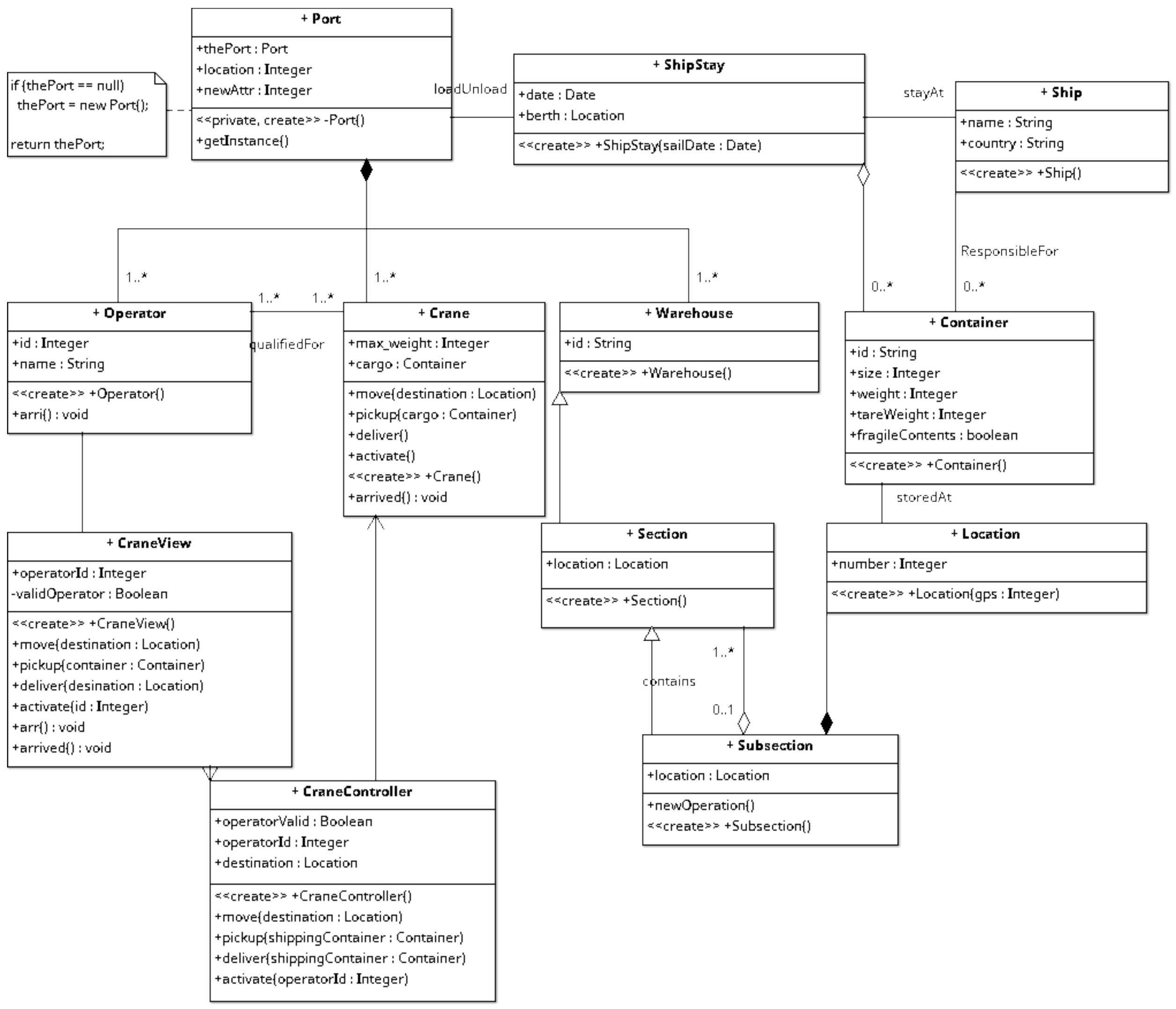
1. Assignment

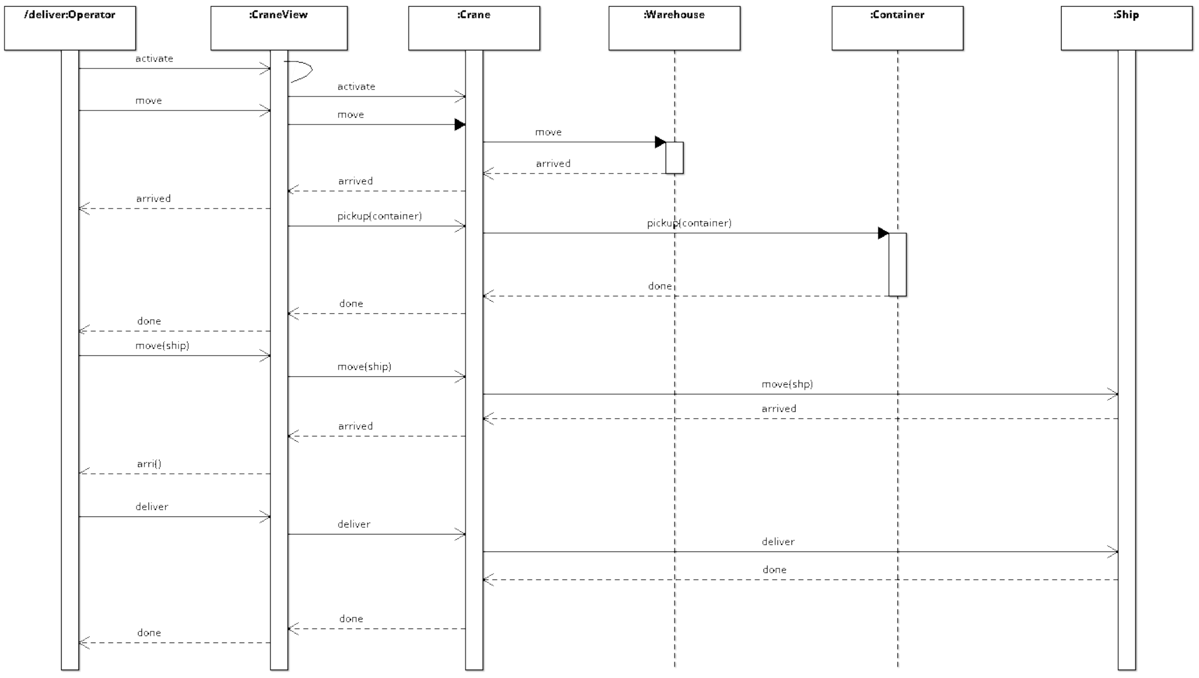
- Continue with the class model of Assignment 2 (without the patterns).
- We need to add security constraints such that crane operators must be authenticated and can only operate the cranes for which they are authorized. Write the constraints in OCL and show in the UML model how to enforce these constraints using views.
- Add safety constraints in OCL such that cranes cannot carry containers that weigh more than their limits and cannot accelerate beyond their limits.
- Write a Java or C++ program to load a container in a ship. Use the attached sequence diagram as a guide.



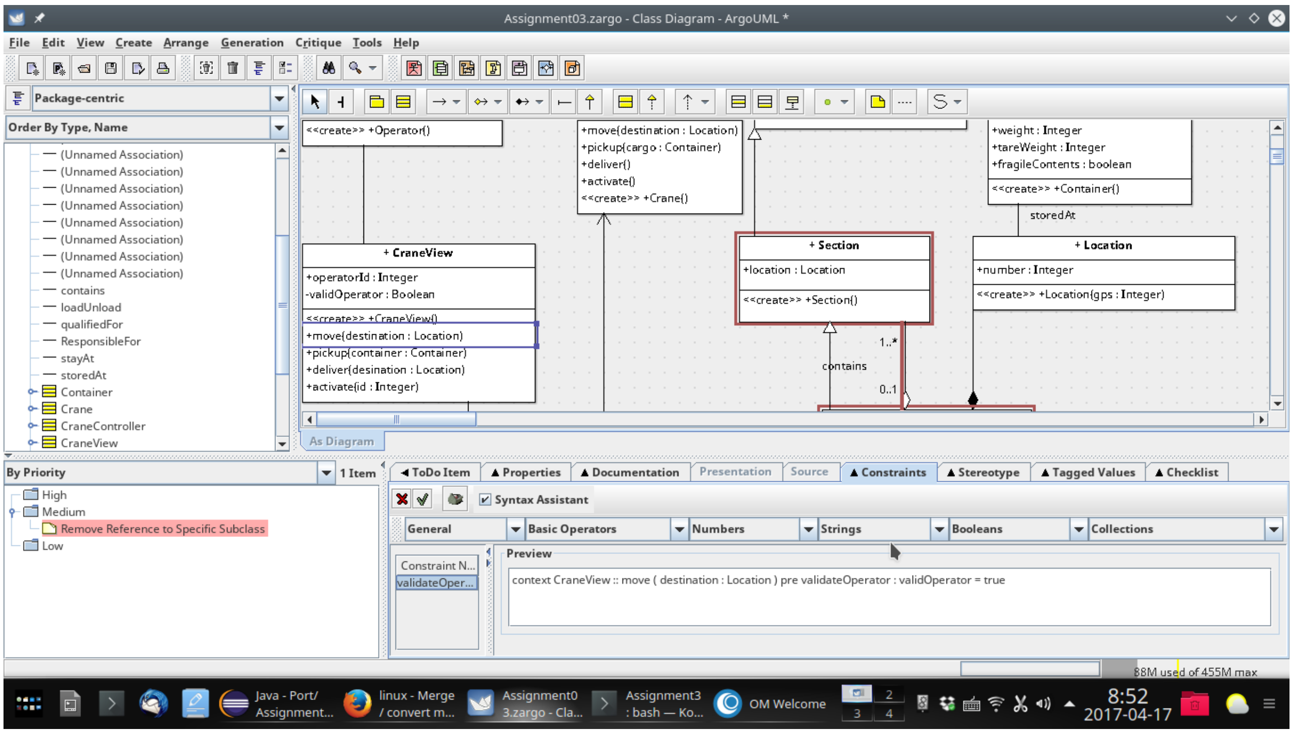
2. Problems/Explanations

Development was done with ARGO UML. The following Class and Sequence diagrams were created with ARGO.


+ Port
+thePort : Port
+location : Integer
+newAttr : Integer
<<private, create>> -Port()
+getInstance()
if (thePort == null)
thePort = new Port();
return thePort;
loadUnload
+ ShipStay
+date : Date
+berth : Location
<<create>> +ShipStay(sailDate : Date)
stayAt
+ Ship
+name : String
+country : String
<<create>> +Ship()
ResponsibleFor
1..*
1..*
1..*
1..*
1..*
0..*
0..*
+ Operator
+id : Integer
+name : String
<<create>> +Operator()
+arri() : void
qualifiedFor
+ Crane
+max_weight : Integer
+cargo : Container
+move(destination : Location)
+pickup(cargo : Container)
+deliver()
+activate()
<<create>> +Crane()
+arrived() : void
+ Warehouse
+id : String
<<create>> +Warehouse()
+ Container
+id : String
+size : Integer
+weight : Integer
+tareWeight : Integer
+fragileContents : boolean
<<create>> +Container()
storedAt
+ CraneView
+operatorId : Integer
-validOperator : Boolean
<<create>> +CraneView()
+move(destination : Location)
+pickup(container : Container)
+deliver(desination : Location)
+activate(id : Integer)
+arr() : void
+arrived() : void
+ Section
+location : Location
<<create>> +Section()
+ Location
+number : Integer
<<create>> +Location(gps : Integer)
1..*
contains
0..1
+ Subsection
+location : Location
+newOperation()
<<create>> +Subsection()
+ CraneController
+operatorValid : Boolean
+operatorId : Integer
+destination : Location
<<create>> +CraneController()
+move(destination : Location)
+pickup(shippingContainer : Container)
+deliver(shippingContainer : Container)
+activate(operatorId : Integer)



Input of OCL was via an OCL Editor



```
context CraneView :: move ( destination : Location ) pre validateOperator : validOperator = true
```

Of note is that the syntax follows OCL conventions:

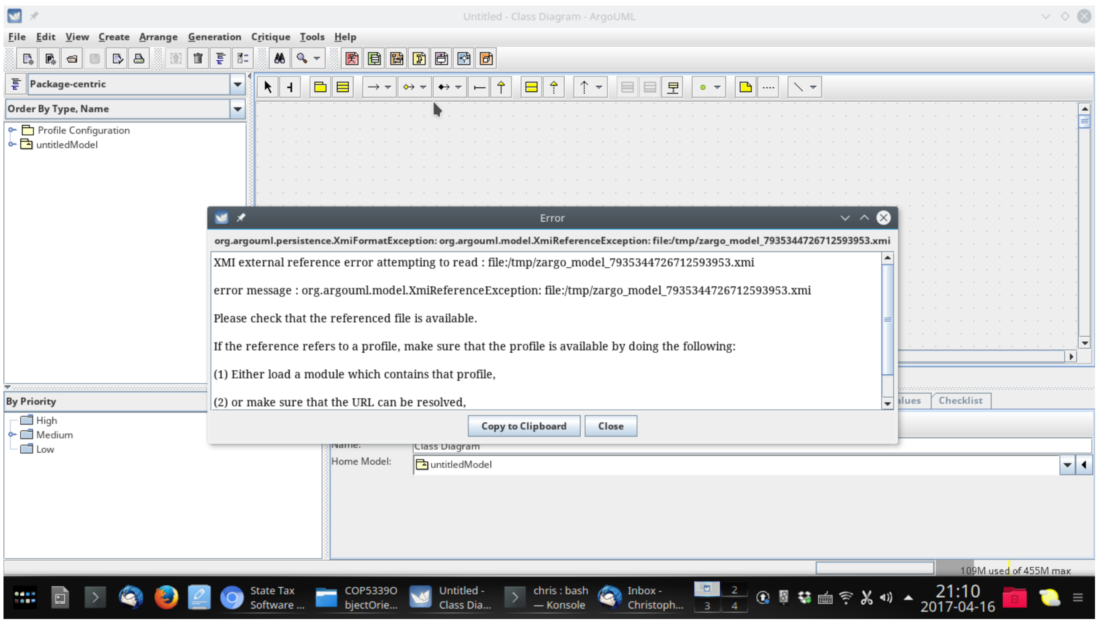
context CraneView :: move (destination : Location) pre validateOperator : validOperator = true

This is then implemented in the code as (blank lines deleted and highlight added for emphasis):

```
import java.util.Vector;
public class CraneView {
  public Integer operatorId;
  private Boolean validOperator;
    public Vector  myOperator;
    public Vector  myCraneController;
  public CraneView() {
  }
    /**
   *
   * @precondition validateOperator: validOperator = true
   */
  public void move(Location destination) {
  }
  public void pickup(Container container) {
  }
```

2.1. Problems

During development, during the final check on April 16, a NULL Pointer exception was encountered which corrupted the profile for the project **and** its automatic backup, with no graphic notice to the user. The error appeared on the Console but Argo gracefully recovered with no appreciable problem until the saved files were reopened. This was evidenced by the following:



```
org.argouml.persistence.XmiFormatException: org.argouml.model.XmiReferenceException: file:/tmp/zargo_model_7935344726712593953.xmi

XMI external reference error attempting to read : file:/tmp/zargo_model_7935344726712593953.xmi

error message : org.argouml.model.XmiReferenceException: file:/tmp/zargo_model_7935344726712593953.xmi

Please check that the referenced file is available.

If the reference refers to a profile, make sure that the profile is available by doing the following:

(1) Either load a module which contains that profile,

(2) or make sure that the URL can be resolved,
```

Prior to the first faiilure constraints had been created in classes as evidenced by the following grep output (after the error the previously generated source was copied to a saved location):

```
[chris@euclid Assignment3Save]$ ls

Container.java        CraneView.java     Operator.java  Ship.java
Warehouse.java

CraneController.java  gzipOfAssignment3  Port.java      ShipStay.java

Crane.java            Location.java      Section.java   Subsection.java

[chris@euclid Assignment3Save]$ grep pre *.java

CraneView.java:   * @precondition validOperator: validOperator = true

CraneView.java:   * @precondition validOperator: validOperator = true

CraneView.java:   * @precondition weightLimit: crane . max_weight >= container .
weight

CraneView.java:   * @precondition cargoPresent: crane . cargoForTransport <> 0

CraneView.java:   * @precondition validOperator: validOperator = true
```

It can be seen that the only missing was the speed scaling when a fragile container was loaded. The operator validation was at the CraneView module and the speed was to be added at the CraneController. The CargoPresent and weightLimit were to be added to both CraneView and CraneController.

I have not decompiled or run the resulting code to determine the java compiler's implementation of the @precondition directive.

3. Source Generated

The following files were generated, no blank lines removed.

3.1. Container.java

```
import java.util.Vector;


public class Container {


  public String id;


  public Integer size;


  public Integer weight;


  public Integer tareWeight;


  public boolean fragileContents;


    public Ship ResponsibleFor;
    public Vector  myShipStay;
    public Location storedAt;


  public Container() {
  }


}
```

3.2. CraneView.java

```
import java.util.Vector;

public class CraneView {

  public Integer operatorId;

  private Boolean validOperator;

    public Vector  myOperator;
    public Vector  myCraneController;

  public CraneView() {
  }

    /**
   *
   * @precondition validateOperator: validOperator = true
   */
  public void move(Location destination) {
  }

  public void pickup(Container container) {
  }

  public void deliver(Location desination) {
  }
```

```
 public void activate(Integer id) {
  }

  public void arrived() {
  }

}
```

3.3. Port.java

```
import java.util.Vector;

/*
 * getInstance()
 */
public class Port {

  public Port thePort;

  public Integer location;

  public Integer newAttr;

    public Vector  loadUnload;
    /**
   *
   * @element-type Operator
   */
  public Vector  myOperator;
```

```
    /**
   *
   * @element-type Crane
   */
  public Vector  myCrane;
    /**
   *
   * @element-type Warehouse
   */
  public Vector  myWarehouse;

  private Port() {
  }

  public void getInstance() {
  }

}
```

3.4. ShipStay.java

```
import java.util.Vector;
import java.util.Date;

public class ShipStay {

  public Date date;

  public Location berth;
```

```
    public Vector  loadUnload;
    public Vector  stayAt;
    /**
   *
   * @element-type Container
   */
  public Vector  myContainer;

  public ShipStay(Date sailDate) {
  }

}
```

3.5. CraneController.java

```
import java.util.Vector;

public class CraneController {

  public Boolean operatorValid;

  public Integer operatorId;

  public Location destination;

      public Vector  myCrane;

  public CraneController() {
  }

  public void move(Location destination) {
  }

  public void pickup(Container shippingContainer) {
  }

  public void deliver(Container shippingContainer) {
  }

  public void activate(Integer operatorId) {
  }

}
```

3.6. Location.java

```
import java.util.Vector;

public class Location extends Warehouse {

  public Integer number;

    public Container storedAt;
    public Vector  mySubsection;

  public Location(Integer gps) {
  }
}
```

3.7. Section.java

```
public class Section extends Warehouse {

  public Location location;

    public Subsection contains;

  public Section() {
  }

}
```

3.8. Subsection.java

```
import java.util.Vector;

public class Subsection extends Section {

  public Location location;

    public Location myLocation;
    /**
   *
   * @element-type Section
   */
  public Vector  contains;
  public void newOperation() {
  }

  public Subsection() {
  }

}
```

3.9. Crane.java

```
import java.util.Vector;

public class Crane {

  public Integer max_weight;

  public Container cargo;

    /**
   *
   * @element-type Operator
   */
  public Vector  qualifiedFor;
    public Vector  myPort;

  public void move(Location destination) {
  }

  public void pickup(Container cargo) {
  }

  public void deliver() {
  }

  public void activate() {
  }

  public Crane() {
  }

  public void arrived() {
  }

}
```

3.10. Operator.java

```
import java.util.Vector;

public class Operator {

  public Integer id;

  public String name;

    /**
   *
   * @element-type Crane
   */
  public Vector  qualifiedFor;
    public Vector  myPort;
    public Vector  myCraneView;

  public Operator() {
  }

  public void arri() {
  }

}
```

3.11. Ship.java

```
import java.util.Vector;

public class Ship {

  public String name;

  public String country;

    public Vector  stayAt;
    /**
   *
   * @element-type Container
   */
  public Vector  ResponsibleFor;
  public Ship() {
  }

}
```

3.12. Warehouse.java

```
import java.util.Vector;

public class Warehouse {

  public String id;

    public Vector  myPort;

  public Warehouse() {
  }

}
```