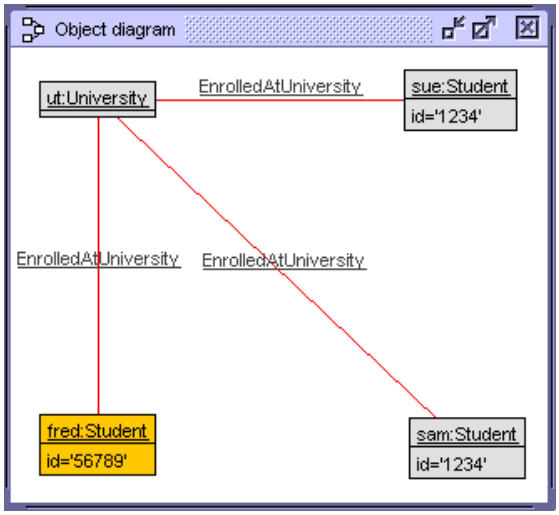
Homework IV

Gregory Williams
GW4975
EE 382C Requirements Engineering

12/03/2015

4.1.1

The invariant is violated by `Fred` who has an `id` with length equal to `5`. The invariant `StudentIdMustBeLength4` works within the `University` context; it loads the set of all of the `students` that belong to `EnrolledAtUniversity` (i.e. are enrolled at the University) and checks each student's `id` to see if the length is equal to `4`.



4.1.2

```
-- OCL constraints
constraints
context University
    -- A student's id number must be exactly
    -- four characters long
    inv StudentIdMustBeLength4:
        self.students->forAll( s | s.id.size() = 4 )

    -- A student's id number must be unique
    inv StudentIdMustBeUnique:
        self.students->forAll( s1, s2 | s1.id = s2.id implies s1 = s2 )
```

4.1.3

```
checking invariant (2) `University::StudentIdMustBeUnique': FAILED.
-> false : Boolean
Results of subexpressions:
    University.allInstances : Set(University) = Set{@ut}
    self : University = @ut
    self.students : Set(Student) = Set{@fred,@sam,@sue}
    s1 : Student = @fred
    s1.id : String = '56789'
    s2 : Student = @fred
    s2.id : String = '56789'
    (s1.id = s2.id) : Boolean = true
    s1 : Student = @fred
    s2 : Student = @fred
    (s1 = s2) : Boolean = true
    ((s1.id = s2.id) implies (s1 = s2)) : Boolean = true
    s1 : Student = @fred
    s1.id : String = '56789'
    s2 : Student = @sam
    s2.id : String = '1234'
    (s1.id = s2.id) : Boolean = false
    ((s1.id = s2.id) implies (s1 = s2)) : Boolean = true
    s1 : Student = @fred
    s1.id : String = '56789'
    s2 : Student = @sue
    s2.id : String = '1234'
    (s1.id = s2.id) : Boolean = false
    ((s1.id = s2.id) implies (s1 = s2)) : Boolean = true
    s1 : Student = @sam
    s1.id : String = '1234'
    s2 : Student = @fred
    s2.id : String = '56789'
    (s1.id = s2.id) : Boolean = false
    ((s1.id = s2.id) implies (s1 = s2)) : Boolean = true
    s1 : Student = @sam
    s1.id : String = '1234'
    s2 : Student = @sam
    s2.id : String = '1234'
    (s1.id = s2.id) : Boolean = true
    s1 : Student = @sam
    s2 : Student = @sam
    (s1 = s2) : Boolean = true
    ((s1.id = s2.id) implies (s1 = s2)) : Boolean = true
    s1 : Student = @sam
    s1.id : String = '1234'
    s2 : Student = @sue
    s2.id : String = '1234'
    (s1.id = s2.id) : Boolean = true
    s1 : Student = @sam
    s2 : Student = @sue
    (s1 = s2) : Boolean = false
    ((s1.id = s2.id) implies (s1 = s2)) : Boolean = false
    self.students->forAll(s1, s2 : Student | ((s1.id = s2.id) implies (s1 = s2))) : Boolean = false
    University.allInstances->forAll(self : University | self.students->forAll(s1, s2 : Student |
     ((s1.id = s2.id) implies (s1 = s2)))) : Boolean = false
```

4.1.4

```
-- OCL constraints
constraints

context University
    -- A student may be a GraduateStudent
    -- or an UndergraduateStudent
    -- but not both
    inv StudentIsGradOrUndergradNotBoth:
        self.undergraduates->intersection(self.graduates)->isEmpty()
```

4.1.5

```
checking invariant (1) `University::StudentIsGradOrUndergradNotBoth': FAILED.
-> false : Boolean
Results of subexpressions:
    University.allInstances : Set(University) = Set{@ut}
    self : University = @ut
    self.undergraduates : Set(Student) = Set{@sam}
    self : University = @ut
    self.graduates : Set(Student) = Set{@sam}
    self.undergraduates->intersection(self.graduates) : Set(Student) = Set{@sam}
    self.undergraduates->intersection(self.graduates)->isEmpty : Boolean = false
    University.allInstances->forAll(self : University |
     self.undergraduates->intersection(self.graduates)->isEmpty) : Boolean = false
```

4.1.6

```
-- OCL constraints
constraints

context University
    -- A student may not exceed the maxApprovedSemesterHours
    -- All Courses offered are assumed to have 3 credit hours
    inv StudentIsGradOrUndergradNotBoth:
        self.students->forAll(s | s.takingCourses->size() * 3 <= s.maxApprovedSemesterHours)
```

4.1.7

```
checking invariant (1) `University::StudentIsGradOrUndergradNotBoth': FAILED.
-> false : Boolean
Results of subexpressions:
    University.allInstances : Set(University) = Set{@ut}
    self : University = @ut
    self.students : Set(Student) = Set{@sam}
    s : Student = @sam
    s.takingCourses : Set(Course) = Set{@BUS311,@CS306,@E306,@EE302,@EE323,@EE338,@EE379K}
    s.takingCourses->size : Integer = 7
    3 : Integer = 3
    (s.takingCourses->size * 3) : Integer = 21
    s : Student = @sam
    s.maxApprovedSemesterHours : Integer = 18
    ((s.takingCourses->size * 3) <= s.maxApprovedSemesterHours) : Boolean = false
```

```
    self.students->forAll(s : Student | ((s.takingCourses->size * 3) <= s.maxApprovedSemesterHours)) :
     Boolean = false
    University.allInstances->forAll(self : University | self.students->forAll(s : Student |
     ((s.takingCourses->size * 3) <= s.maxApprovedSemesterHours))) : Boolean = false
```

4.2.1

```
-- OCL constraints
constraints

context Student
    inv studentEnrolledInUniversity: self.isEnrolledAt.students->includes(self)

context Student :: drop(c : Course)
    pre studentIsRegistered: self.takingCourses->includes(c)
    pre studentHasMoreThanOneClass: self.takingCourses->size() > 1
    post studentIsNotRegistered: not self.takingCourses->includes(c)
    post studentDidNotDropOtherCoursesRegistered: self.takingCourses->including(c) =
     self.takingCourses@pre
    post droppedCourseNotFull: c.isFull = false
    post studentStillEnrolledInUniversity: self.isEnrolledAt = self.isEnrolledAt@pre
    post onlyThisStudentWasRemoved: c.studentsEnrolled->including(self) = c.studentsEnrolled@pre
```

4.2.2

```
!create ut : University
!create sam : Student
!create sue : Student
!insert (sam,ut) into EnrolledAtUniversity
!insert (sue,ut) into EnrolledAtUniversity
!create EE302 : Course
!create CS306 : Course
!create BUS311 : Course
!create EE411 : Course
!create EE379K : Course
!create E306 : Course
!create EE338 : Course
!create EE323 : Course
!insert (sam,EE302) into TakingCourse
!insert (sam,CS306) into TakingCourse
!insert (sam,BUS311) into TakingCourse
!insert (sam,EE323) into TakingCourse
!insert (sam,EE379K) into TakingCourse
!insert (sam,E306) into TakingCourse
!insert (sam,EE338) into TakingCourse
!insert (sue,EE302) into TakingCourse
!insert (sue,CS306) into TakingCourse
!insert (sue,BUS311) into TakingCourse
!insert (sue,EE323) into TakingCourse
!set EE302.isFull := true
!openter sam drop(EE302)
!delete (sam,EE302) from TakingCourse
!set EE302.isFull := false
!opexit
```

4.2.3

```
CR2.cmd> !openter sam drop(EE302)
precondition `studentIsRegistered' is true
precondition `studentHasMoreThanOneClass' is true
CR2.cmd> !delete (sam,EE302) from TakingCourse
CR2.cmd> !set EE302.isFull := false
CR2.cmd> !opexit
postcondition `studentIsNotRegistered' is true
postcondition `studentDidNotDropOtherCoursesRegistered' is true
postcondition `droppedCourseNotFull' is true
postcondition `studentEnrolledInUniversity' is true
postcondition `onlyThisStudentWasRemoved' is true
```