**Name: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Roll#: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Start Time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ End Time: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Maximum Time Allowed: 45 min**

**Questions- Case Study 1- AspectOCL**

1. Considering **AspectOCL Constraint# 1**, perform the following tasks.
   1. Change the constraints by replacing iterator **“select”** with iterator **“one”** and correspondingly removing size () =1. **”**. **(Make changes directly on constraints sheet)**
   2. Add uniqueness condition within the “**select”** iterator for one more attribute of each class in let clause (for example branch, branchType,..) using **“and”** operator. List of the mapping attributes corresponding to their context that is to be added is given below in table 1. The new constraint will become,

***result****= T.allInstances()->* ***select****(t|t.name=self.name* ***and*** *b. A =self. A)->****size****()=1*

For example, the let expression is updated as (Similarly do it for others)

**let T->A : Branch -> BranchCode,**

**BranchType -> TypeCode, ………**

**(Write constraint in the space provided on the constraints sheet)**

**Table 1 Context and Constraints to be Added**

|  |  |
| --- | --- |
| **Context (T)** | **Mapping (A)** |
| Branch | BranchCode |
| BranchType | TypeCode |
| PerformanceIndicator | PerfInstrument |
| Country | CountryCode |
| CarModel | ModelNum |
| CarGroup | GroupLabel |
| RentalDuration | DurationType |
| ServiceDepot | DepotNumber |
| Discount | DiscountType |

1. Considering **AspectOCL Constraint# 2** defined on functions **pickUpBranch()** and **dropOffBranch()** (defined in let expressions)**,** add a similar constraint for another function **maintenanceBranch()**  by adding one more expression in mapping part such that variable **T** will represent MakeRental::maintenanceID():Branch and variable **A** will represent MakeRental ::MaintenanceID. **(You have to define mapping yourself like in question 1, i-e T->A. Write constraint in the space provided on the constraints sheet)**
2. For **AspectOCL Constraint# 3**, append following constraints shown in table 2 using **“and”** operator in the **“select”** iterator clause. For example, the new select clause becomes,

*…….(select(b|b.id=self.id and t.A.size()= 5)->size()=1)……*

Update the mapping part as example shown below for the variable values in table 2,

*let T-> A: {EU\_RentPerson -> barcode, ……..}*

**(Write constraint in the space provided on the constraints sheet)**

**Table 2 Variable Values**

|  |  |
| --- | --- |
| **Context (T)** | **Mapping (A)** |
| Eu\_RentPerson | barcode |
| PendantOrder | carOrder |

1. For **AspectOCL Constraint# 4** append following constraint at the end of the constraint using **“and”** operator.

Constraint in OCL to be added*: (self.CheckAvailabilty(self.A ->isNotEmpty ())*

**(Make changes directly on constraints sheet)**

1. For **AspectOCL constraint# 5**, perform the following tasks.
   1. Change the constraints by replacing iterator **“one”** with iterator **“select”**. **(Make changes directly on constraints sheet)**
   2. Combine the two clauses specified using **“one”** iterator into a single clause as:

*.… result= T.allInstances()->one(t|t.first->isEmpty()) and t.second->isEmpty()) …..*

**(Write constraint in the space provided on the constraints sheet)**

* 1. Change the constraint by replacing the function **“isEmpty()”** with function **“size()=0”.** **(Make changes directly on constraints sheet)**

1. For **AspectOCL constraint# 6**, reform the constraint by deleting following clause from it.

*self. perf= B*

For this you will have to update the **mapping** part by deleting the attributes representing B. For example, by eliminating *ExistingRentalDuration::durationLimit* and *ExistingPerformanceIndicator:: performanceLevel*. **(Make changes directly on constraints sheet)**

1. For **AspectOCL constraint# 7,** perform following tasks.
   1. Update the let expression (in the **post** condition) by replacing the “**select**” iterator with “**collect**” iterator such that the new condition becomes;

…….*“collect(c|c.A)”….*

*………………………………..*

For this you will have to update the **mapping** part by deleting the attributes representing B. For example, by deleting **ExistingCar::regNumber** , **ExistingCarGroup ::carGroup** (similarly for other mappings). **(Make changes directly on constraints sheet)**

* 1. For the constraints defined on context **“ExistingCarGroup::carG():CarGroup”** and **“ExistingCarModel::carM():CarModel” ( see from mapping part)**, update the iterator condition in the let expression (in the **post** condition, not mapping) by changing the equality sign “**=**” into greater than sign “**>**”. For this purpose,
     1. Update the mapping by removing the complete expression starting with context **ExistingCarGroup** and **ExistingCarModel.**
     2. Then write one separate aspect for these two contexts **“ExistingCarGroup::carG():CarGroup”**and**“ExistingCarModel::carM():CarModel”** similarly as shown in **AspectOCL constraint# 7**. **(Write constraint in the space provided on the constraints sheet)**

1. For **AspectOCL constraint# 8,** update the **post** condition of the operations by appending following clause at the end of constraints using an **“or”** operator.

*“cgdp.rentalduration.lastmodification= now()”*

**(Make changes directly on constraints sheet)**

1. For **AspectOCL constraint# 9**, perform following tasks. **(Make changes directly on constraints sheet for both parts)**
   1. Replace the clause *oclIsTypeOf(MoveCars)* with *oclIsKindOf(S)*. Update the mapping by adding additional variable S such that T-> {A,B,S}. will represent RequestTransfer and DoTransfer.
   2. For function **“intersection” in both constraints**, *oclIsKindOf(BeingTransferredCar)* with *oclIsKindOf(OwnCar).*