**OCL**

**context Task**

**inv:**

**--Status must be either "Done" or "Pending"**

**self.status = Status::Done or self.status = Status::Pending**

**-- Task must have a parent TodoList**

**self.parentTodo->notEmpty**

**-- Task IDs should be unique**

**self.allInstances()->isUnique(taskId)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**context User**

**inv:**

**-- User must have a unique username**

**self.getAllUsers()->isUnique(username)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**context Folder**

**inv:**

**--Every Folder must have exactly one owner**

**self.owner->size() = 1**

**-- A Folder may have a parent Folder and multiple subFolders. Root folders do not have a parent.**

**self.parentFolder->isEmpty() implies self.subFolders->isEmpty() =**

**true**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**context TodoList**

**inv:**

**-- TodoList must have an onwer.**

**self.owner->notEmpty()**

**--TodoList can have zero or more Tasks.**

**self.tasks->size() >= 0**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--OCL constraint that the one who can delete the task is the one who can write to-do list**

**context Task**

**inv:**

**self.list.owner.DeleteTask(self) implies**

**self.list.owner.canWriteToDoList()**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--OCL constraint that the to-do list must have a parent**

**context ToDoList**

**inv:**

**self.parent <> null**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--OCL constraint that the one who update to-do list is either the**

**owner of Todo or have edit permission**

**context ToDoList**

**inv:**

**self.owner = self.updater or self.editPermission.exists(p | p.user =**

**self.updater)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**-- OCL constraint that the one who share to-do list must be the**

**owner of to-do list**

**context ToDoList**

**inv:**

**self.sharedWith->isEmpty() or self.sharedWith->forAll(u | u =**

**self.owner)**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--OCL constraint that ensures the person deleting a to-do list is the**

**owner of that to-do list**

**context ToDoList**

**inv: deleter = owner**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--OCL constraint that who can read a to-do list should be either the**

**owner of that to-do list or that to-do list has public view**

**context ToDoList**

**inv:**

**self.readPermission.exists(p | p.user = self.owner) or self.isPublicView**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--OCL constraint that to create a to-do list you must be either owner**

**of the parent folder of that to-do list you want to create or you have**

**permission to edit the parent folder of that to-do list**

**context ToDoList**

**inv:**

**self.parentFolder <> null implies (self.parentFolder.owner =**

**self.creator or self.parentFolder.editPermission.exists(p | p.user =**

**self.creator))**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Folder must have parent , Folder name is not Root**

**context Folder**

**inv:**

**self.parent <> null and self.name <> 'Root'**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Todo must have parent , Folder name is not Root**

**context Todo**

**inv:**

**self.parent <> null and self.name <> 'Root'**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--update Folder => owner of Todo || have Edit permission , Folder name is not Root**

**context Folder**

**inv:**

**self.update(owner) or self.hasEditPermission()**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Share Folder => Owner of todo, Folder name is not Root**

**context Folder**

**inv:**

**self.share(owner) and self.name <> 'Root'**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Delete => owner of Todo, Folder name is not Root, Todo inside Folder must be deleted with it**

**context Folder**

**inv:**

**self.delete(owner) and self.name <> 'Root' and self.todos->forAll(todo | todo.delete())**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Read Folder => owner || Todo has ( Public || User ) View**

**context Folder**

**inv:**

**self.read(owner) or self.todos->forAll(todo | todo.publicView() or todo.userView())**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Create Folder => Owner of parentFolder || Can Edit parentFolder**

**context Folder**

**inv:**

**self.create(owner) or self.parentFolder.hasEditPermission()**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--update user, where user != modifier, user => role == Admin**

**context User**

**inv:**

**self.update(user, modifier) implies user.role = 'Admin'**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Delete user**

**context User**

**inv:**

**self.delete()**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**--Get all user, user => role == Admin**

**context User**

**inv:**

**self.getAll() implies self.role = 'Admin'**