Formal Requirement Specification

For Online Real Estate System

Version 1.0

Submitted in partial fulfillment of the requirements of the degree of MSE

Mosaad Aloomery
CIS 895 – MSE Project
Kansas State University
model ORES

--
-- ENUMERATIONS
--

denum State {Selled, NotSelled}
denum Air {Yes, No}
denum Furn {Yes, No, Half}

--
-- CLASSES
--

class User
attributes
    userName:String
    password:String
operations
    verifyLogin(userName:String,password:String):Boolean =
        User.allInstances->exists(u:User | u.userName = userName and u.password = password)
logout()
end

class Manager < User
operations

CreateEmployee(UserName:String, password:String):Boolean
    --Employee userName should not be already used
    pre:Employee.allInstances->forall(e1:Employee | e1.userName <> userName)
    --record of existing employees are not changed only new employee is added
    post:Employee.allInstances.userName =
        Employee.allInstances.userName@pre->including(userName)

    post:Employee.allInstances.password =
        Employee.allInstances.password@pre->including(password)

    MaintainLocation()
end
class Employee < User
operations
CreateNewRealEstate()
end
class Customer
attributes
fullName:String
email:String
phone:Integer
address:String
operations
Search()
Request()
end
class Property
attributes
Price:Integer
BPrice:Integer
City:String
Status:State
ID:Integer
Quarter:String
Region:String
Street:String
operations
getProperty(ID:Integer): Boolean =
Property.allInstances->exists(p:Property | p.ID = ID)
updateProperty(Price:Integer, Bprice:Integer, City:String, Status:
State, ID: Integer, Quarter:String, Region:String, Street:String):
Boolean
-- pre: the user in logedin
--the property should exists
pre: Property.allInstances.ID->includes(ID)
post: Property.allInstances.ID = Property.allInstances.ID@pre
post: Property.allInstances.Price = Property.allInstances->
>select(p:Property | p.ID <> ID).Price@pre->including(Price)
post: Property.allInstances.BPrice = Property.allInstances->select(p: Property | p.ID <> ID).BPrice@pre->including(BPrice)
post: Property.allInstances.City = Property.allInstances->select(p: Property | p.ID <> ID).City@pre->including(City)
post: Property.allInstances.Status = Property.allInstances->select(p: Property | p.ID <> ID).Status@pre->including(Status)
post: Property.allInstances.Quarter = Property.allInstances->select(p: Property | p.ID <> ID).Quarter@pre->including(Quarter)
post: Property.allInstances.Region = Property.allInstances->select(p: Property | p.ID <> ID).Region@pre->including(Region)
post: Property.allInstances.Street = Property.allInstances->select(p: Property | p.ID <> ID).Street@pre->including(Street)

FindProperty(ID: Integer) : Set(Property) =
if Property.allInstances->exists(p: Property | p.ID = ID) then
    Property.allInstances->select(p: Property | p.ID = ID)
else
    oclEmpty(Set(Property))
endif
end

class House < Property
attributes
Site: String
RoomNo: Integer
end

class Land < Property
attributes
size: Integer
end

class Apartment < Property
attributes
airCondition: Air
bathNo: Integer
furniture: Furn
BedRoomNo: Integer
end

--
-- ASSOCIATIONS
--
association PropertyManager between Manager[1] role managedby Property[0..*] role manage
end

association propertyEmployee between Employee[*] role empl Property[*] role prop
end

association propertyCustomer between Customer[*] role ownedby Property[*] role own
end

--
-- C O N S T R A I N T S
--

constraints
--
-- Employee
--
--
-- employee userName should be unique
--

context Employee
inv usernameUnique:
Employee.allInstances->forAll(e1,e2 | e1<>e2 implies e1.userName <> e2.userName)

--
-- Property
--
--
-- Property ID should be unique
--

context Property
inv uniqueID:
Property.allInstances->forAll(p1,p2 | p1<>p2 implies p1.ID <> p2.ID)

--
-- property price should be positive
--
context Property
inv propertyPrice:
self.Price > 0

--
-- Apartment should have at least one bedroom
--

context a:Apartmetn
inv AptBedNo:
a.BedRoomNo > 0