CODICE ALLOY

```
open util/integer
abstract sig  Bool {}
sig True extends Bool{}
sig False extends Bool{}


sig ThirdParty {
	queriesToIndividuals: set QueryToSingleUser,
	queriesToGroup: set QueryToGroupOfUsers,
}


sig QueryToGroupOfUsers{
	mittent: one ThirdParty,
	receivers: one Class,
	dataDemanded: set Data,
	--warnings: set ImpossibleToAnonimyzeDataWarning,
	accepted: one Bool
}

--a query by a ThirdParty in order to access some data
sig QueryToSingleUser {
	mittent:one ThirdParty,
	dataDemanded: set Data,
	receiver: one User,
	accepted: one Bool,
	--warnings: set UnaccessibleDataWarning
}
sig Class {
	members: set User,
	classType: one ClassType
}


abstract sig ClassType{}
one sig Over70People extends ClassType{}
--one sig ResidentInMilanPeople extends ClassType{}


sig User {
	personalData: set Data,
	preferences: one D4HPreferences,
	notifications: set UserNotification,
	receivedQueries: set QueryToSingleUser
}


abstract sig Data {
	owner: one User,
	dataType: one DataType,
```

```
	accessible: one Bool
}


sig DataType{}
sig HealthData extends DataType {
	healthStatus: one HealthStatus
}
sig LocationData extends DataType {
	locations: one  Location
}


sig Location {
	-- latitude
	coordX: one Int,
	 -- longitude
	coordY: one Int
}

sig  HealthStatus{}

--notification reaches a User, when a ThirdParty asks for his data
sig UserNotification {
	notified: one User,
	queryReceived: one QueryToSingleUser,
	confirmQuery: one Bool
}


sig D4HPreferences {
	user: one User,
	--data defined by the User as accessible by third parties
	accessibleData: set Data,
	--data defined by the User as not accessible by any third party
	unaccessibleData: set Data,
	--third parties which the User defines as allowed to access some data
	allowedThirdParties: set ThirdParty,
	--third parties which the User defines as not allowed to access some data
	notAllowedThirdParties: set ThirdParty
}


fact DataUserConnection {
	all  u: User | all  d: Data | ( d in u.personalData implies  d.owner = u) and
	(d.owner = u implies  d in u.personalData)
}


fact NoTwoUsersToTheSameData {
	all d1: Data | no disj u1,u2:User | d1.owner= u1 and d1.owner= u2
}
```

```
fact UserPreferencesConnection {
	all u1: User | all p1: D4HPreferences | ( u1.preferences= p1 implies p1.user = u1) and
	(p1.user= u1 implies u1.preferences= p1)
}


fact NoTwoUserToSamePreferences {
	all p1: D4HPreferences | no disj u1, u2: User | u1.preferences=p1 and u2.preferences=p1
}


fact NoTwoPreferencesToSameUser {
	all u1: User | no disj p1, p2: D4HPreferences | p1.user= u1 and p2.user=u1
}


fact DataAccessibleD4HPreferencesConnection {
	all u1:User | all p1: D4HPreferences | all d1: Data | (p1= u1.preferences and d1 in
p1.accessibleData) implies
	(d1 in u1.personalData and d1.accessible=True)
}


fact DataUnaccessibleD4HPreferencesConnection {
	all u1:User | all p1: D4HPreferences | all d1: Data | (p1= u1.preferences and d1 in
p1.unaccessibleData) implies
	(d1 in u1.personalData and d1.accessible=False)
}

--each data defined in a D4HPreferences setting can not be owned by two different users,
--because each set of preferences refers to only one user and only to his data
fact DataInPreferencesReferToOnlyOneUser{
	all p1: D4HPreferences| all d1: Data| no disj u1,u2: User |
	(d1 in p1.accessibleData and d1 in u1.personalData and d1 in u2.personalData)
	or
	(d1 in p1.unaccessibleData and d1 in u1.personalData and d1 in u2.personalData)
}


fact QueryToSingleUserUserConnection {
	all u1: User | all q1:QueryToSingleUser | (q1.receiver = u1 implies q1 in u1.receivedQueries)
and
	(q1 in u1.receivedQueries implies q1.receiver=u1)
}


fact UnacceptedQueryToSingleUser {
	all q1: QueryToSingleUser | q1.accepted = False iff
	(q1.dataDemanded in q1.receiver.preferences.unaccessibleData or
	q1.mittent in q1.receiver.preferences.notAllowedThirdParties or
```

```
	(some n1: UserNotification | n1.notified= q1.receiver and n1.queryReceived= q1 and
n1.confirmQuery = False))
}


fact AcceptedQueryToSingleUser{
	all q1: QueryToSingleUser | q1.accepted = True iff
	(q1.dataDemanded in q1.receiver.preferences.accessibleData and
	(some n1: UserNotification | n1.notified= q1.receiver and n1.queryReceived= q1 and
n1.confirmQuery = True))
}

fact AcceptedQueryToGroupOfUsers {
	all q1: QueryToGroupOfUsers| q1.accepted=True iff (#q1.receivers.members>1000)
}


fact UnacceptedQueryToGroupOfUsers {
	all q1: QueryToGroupOfUsers| q1.accepted=False iff (#q1.receivers.members<=1000)
}


fact DataAccessibleAnNotAccessibleAtTheSameTime {
	all p1: D4HPreferences |no d1: Data |  ((d1 in p1.accessibleData) and (d1 in
p1.unaccessibleData))
}


fact QueryToSingleUserGenerateUserNotification {
	all q1: QueryToSingleUser | all u1:User | u1= q1.receiver implies
	 (one n1: UserNotification| n1.notified = u1 and n1.queryReceived = q1 and
q1.accepted=n1.confirmQuery)
}


fact NotificationUserConnection {
	all u1: User | all n1: UserNotification | (n1.notified = u1 implies n1 in u1.notifications) and
	(n1 in u1.notifications implies n1.notified=u1)
}


--if a third party makes a query to a single user,the query is not about data concerning another user
fact DataDemandedInQueryToSingleUserAreOwnedByTheSameUserWhoReceivesQuery{
	no q1: QueryToSingleUser | q1.dataDemanded.owner != q1.receiver
}


--if a third party makes a query to a class of users,the query is about data concerning only the data
--about users who are members of this class
fact DataDemandedInQueryToGroupOfUsersAreOwnedByGroupMembers{
	all q1: QueryToGroupOfUsers | no d1:Data | (d1 in q1.dataDemanded) and (d1.owner !in
q1.receivers.members)
```

```
}


pred show () {
	one d1:Data | d1.dataType = HealthData and (one d2:Data | d2.dataType = LocationData)
	one d1:Data | d1.accessible = True and (one d2:Data | d2.accessible= False)
	one t1: ThirdParty | #t1.queriesToIndividuals=1 and #t1.queriesToGroup=0
	one c1: Class | one u1: User | one q1: QueryToSingleUser| c1.members=u1 and
q1.receiver!=u1
}
--run show for 1 but 2 User, 2 Data, 2 DataType, 2 D4HPreferences
run show for 2 but 1 QueryToGroupOfUsers, 1 QueryToSingleUser, 1 Class, 1 ClassType, 1
Location,1 HealthStatus, 1 UserNotification
```