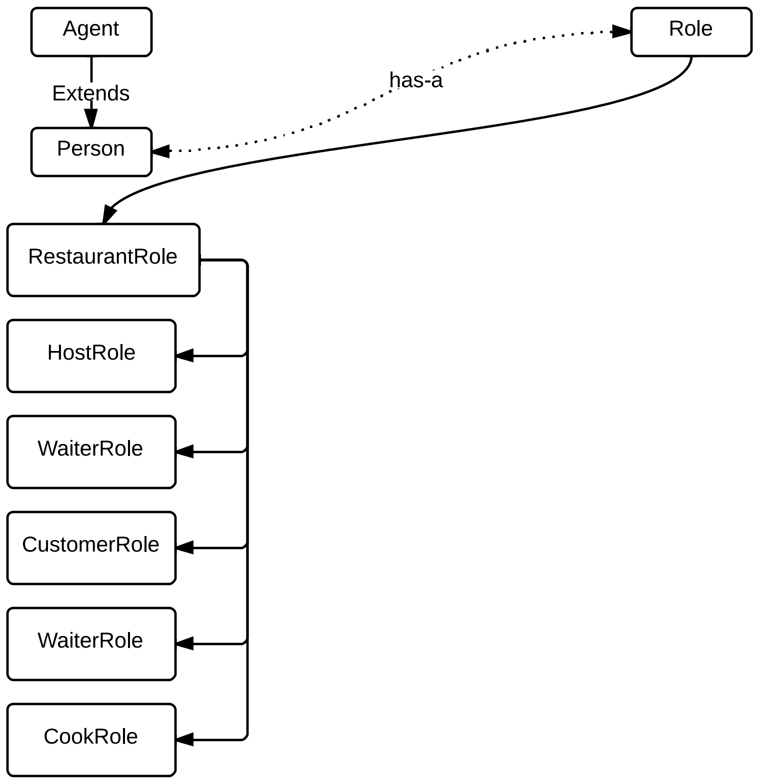

Agent
Extends
Person
has-a
Role
RestaurantRole
HostRole
WaiterRole
CustomerRole
WaiterRole
CookRole

Person Super Class

Data

```
int id;
enum HouseType;
int address;
enum addressType;
Job job;
int curBuilding;
int destination;
enum VehicleType;

enum Intent { withdraw, deposit, sleep, work}

Gui gui;
Semaphore animation;

List<Role> roles;
int hungerLevel;
double money;

Inventory inventory;
enum Moraility {good, crook}
enum PersonState {walking, inside, sleeping, idle}

Inventory inventory;
```

Scheduler

```
//Role pick and execute and action.
if there exists an active Role r in roles
   then boolean ret = r.pickandExectueAnAction()


Otherwise
if PersonState is sleeping
   then goToSleep();

// What to do as a person.
   // If low on money then go to bank.
   // If hungry then go to restaurant.
   // Depending on time of day, go to work or go
home.

if destination is not -1
   then goTo(destination);

return ret;
```

Messages

```
void msgCreateRole(Role r){
   if there does not exists r in roles
      then roles.add(r);
}

void msgSleep(){
   sleeping = true;
}

void msgEnterBuilding(Building b){
   destination = b.id;
}

void msgEnteredBuilding(Building b){
   curBuilding = b.id;
}
```

Utility

```
void goTo(Building b)
{
   b.msgEnterBuilding(this);
   //call gui animation
}

void stateChanged();

void acquire(){
   animation.acquire();
}
void release(){
   animation.release();
}

void goToSleep(){
   destination = house.id;
}
```

Super Class Role

Data

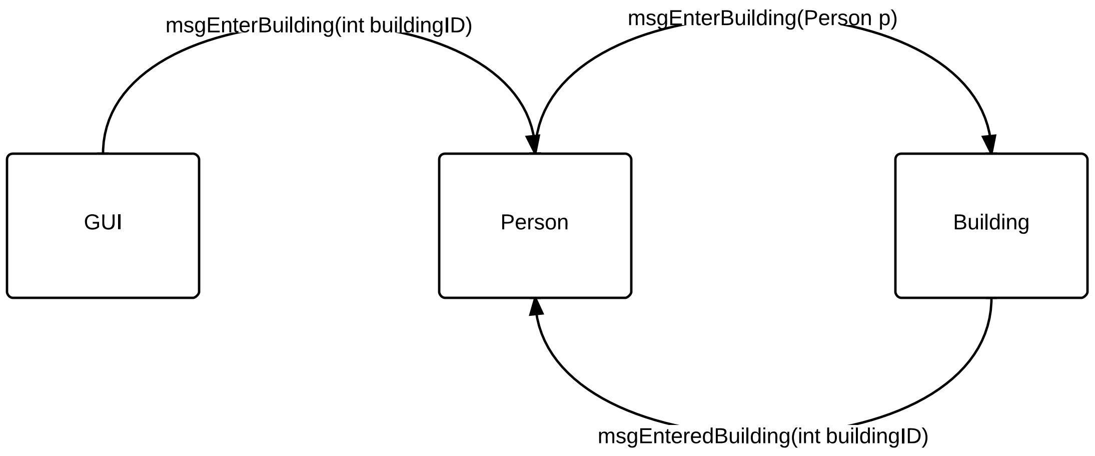
```
Person myPerson;

boolean active;
```

Scheduler

```
abstract pickAndExecuteAction;
```

Messages

```
void stateChanged(){
   myPerson.stateChanged();
}
```


msgEnterBuilding(int buildingID)
msgEnterBuilding(Person p)
GUI
Person
Building
msgEnteredBuilding(int buildingID)