Class relationship graph:

```
                          ┌─────────────┐
                          │  POOFight   │
                          └─────────────┘
                        ↗                 ↖
            ┌─────────────┐          ┌─────────────┐
            │  POOArena   │          │   POOPet    │
            └─────────────┘          └─────────────┘
                  ↑                        ↑
            ┌─────────────┐          ┌─────────────┐
            │ Arena1*1*2  │          │ PetBase*1*2 │
            └─────────────┘          └─────────────┘
```

| Skill_menu*1 | PetList | Status_control | POOAction |

ImageJFrame   ImageButton   LayoutManager   Monsters   POOSkill

TransparentIcon   POOCoordinate   SkillList

CoordinateXY   Skills
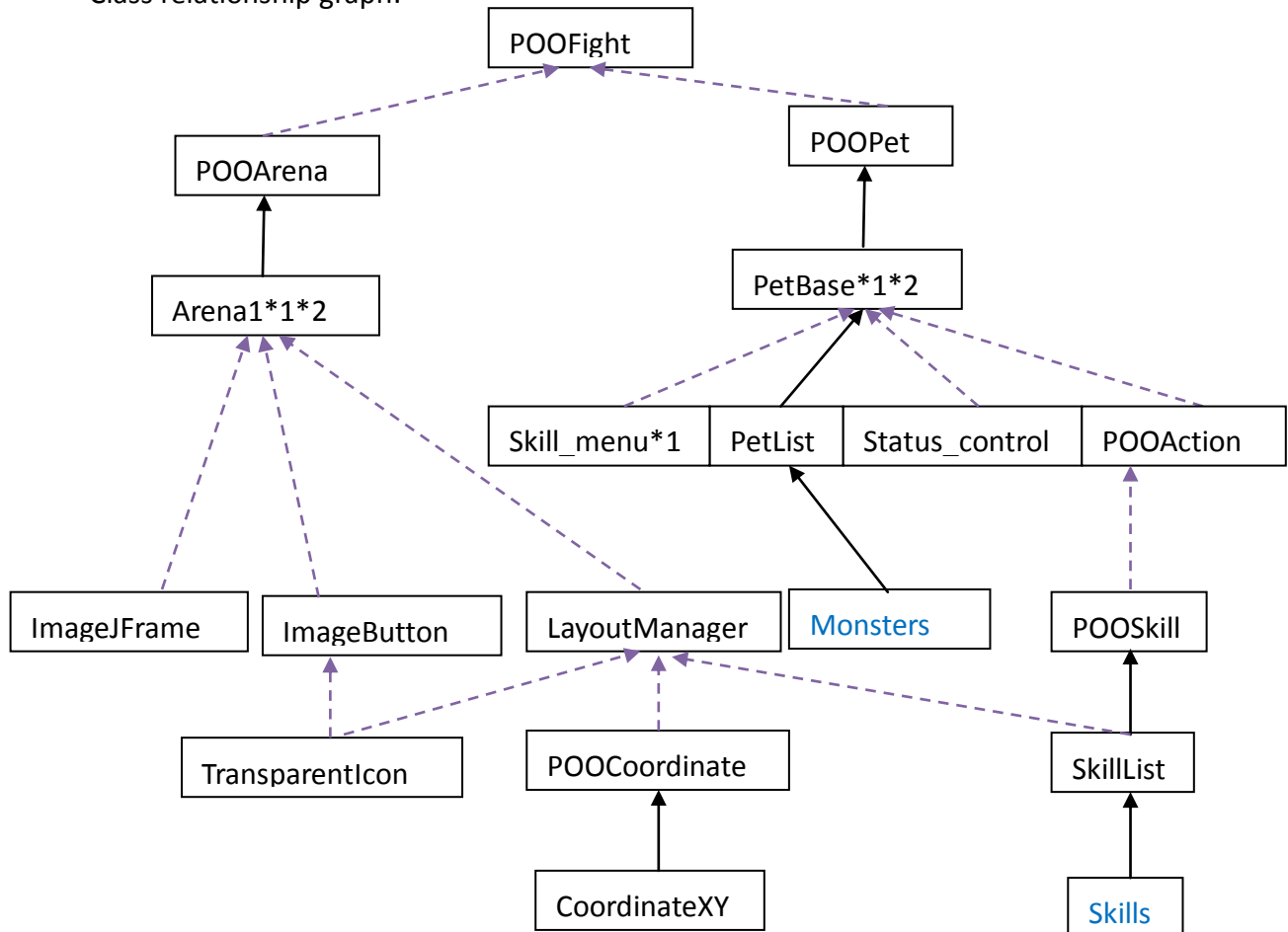
*1: implements MonseListener

*2: implement KeyListener

↑ : inherit

┆↑ : has

Advantage of my design in terms of software engineering:

1.

All basic methods are implement in Base class such as Arena1, PetBase, SkillList

2.

Add new monsters or skills only need to inherit PetBase, SkillList and override some methods

Disadvantage of my design in terms of software engineering:

1.

Any changes in Base class will influence subclass of them, so it is hard to

changes the variables, methods in Base class.

Advantage of my design in terms of interestingness of the game:

1.

Use GUI interface, all things needs for Graphics are implement in Base classes. Add new class only need to override the image path, so the image will change to whatever you want.

2.

LayoutManger control the ground of the arena and has a linkedlist for skills, so it can handle skills that influence floor, and makes the game more interesting.

Disadvantage of my design in terms of interestingness of the game:

1.

Pet hierarchy structure, when implement AI, we needs to use retrieve methods or let variable be protected, let makes some problems in data encapsulation if we use protected for variables.

Bonus:

1.

GUI interface

2.

Implement key, mouse listener for user friendly interface

3.

AI support

4.

Beside normal Attack action, we can support floor Attack. When monster enter some place, they will get hurt or regain health.

5.

Remove background of the monster image

6.

Variance attack type of skills. Contain normal attack skill(TinyAttack), reduce AGI skill(Mucus), Floor skill(Trap, Mud), change coordinate skill(Silking), range skill(Mud).

How to play:

1.

When start, chose monsters you like to control(others will be AI control)

2.

Current act monsters will be orange border, left click the monster, and use arrow key to control its moving. Use esc key to undo moving.

3.

Right click to call skill menu. Left click to select a skill to use.

4.

When there is only one monster at the arena => game over

Skill:

1.

TinyAttack => cause 2 damage, range 1

2.

Mud splash=> cause 2 damage, range 2 splash range 1, reduce 1 agi when hit, make floor be mud: cause extra 1 move point when leaving it, continued time 1

3.

Mucus=> cause 1 damage, range 3, reduce 3 agi when hit

4.

Sliking=> reduce 2 agi, 2 mp, range 5, pull target to you.

5.

Trap=> place a trap on the floor, everyone will not see it except the owner, when other encounter it: cause 2 damage, terminate its round. After triggered: cause extra 3 move point when leaving it, continue time 2.