物件導向程式設計 HW4

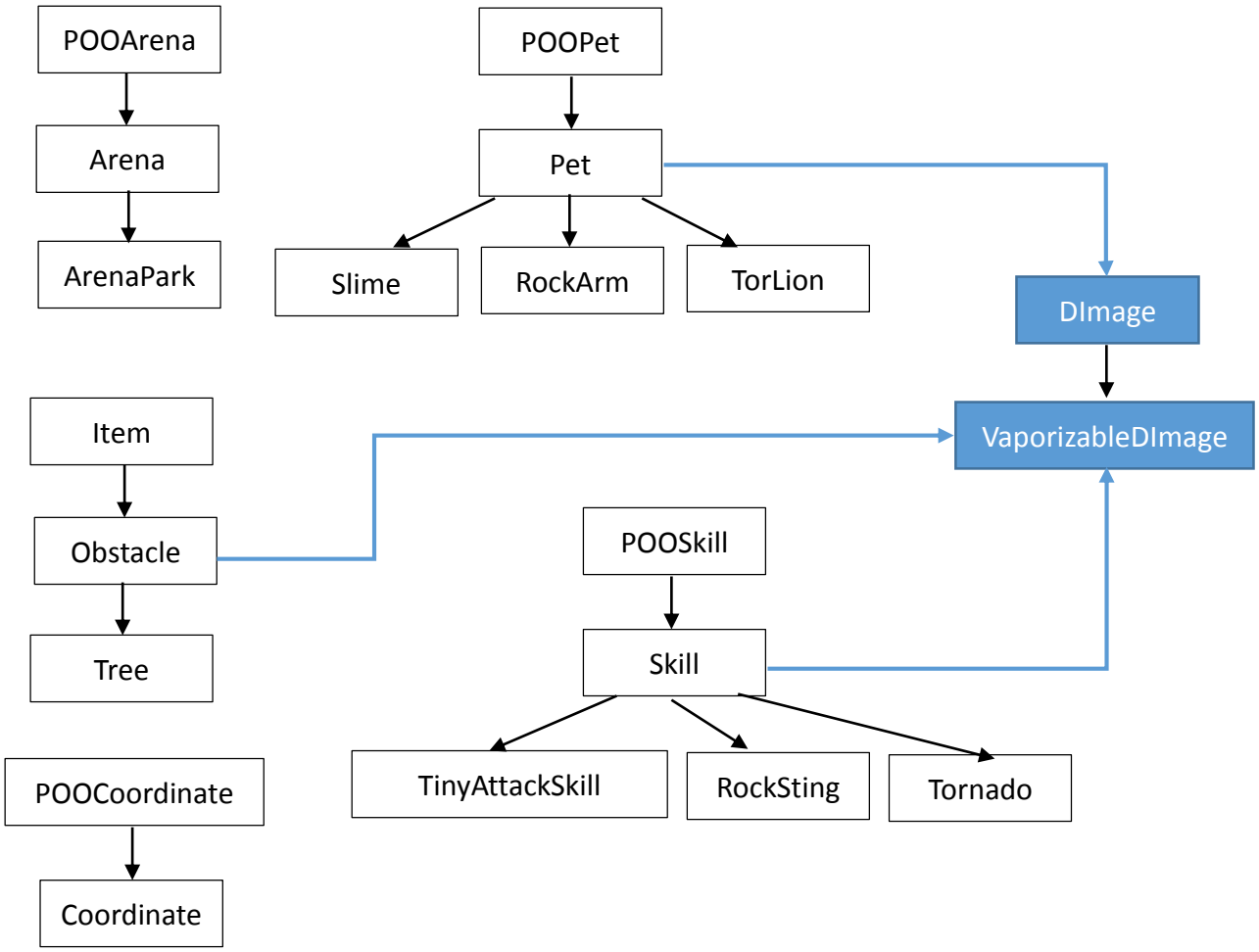
B99902008 資工三 李朋臻

1. Relations between classes

   (1) **Inheritance** and **implementation**

   Black arrow means “extends”, and blue arrow means “implements.”

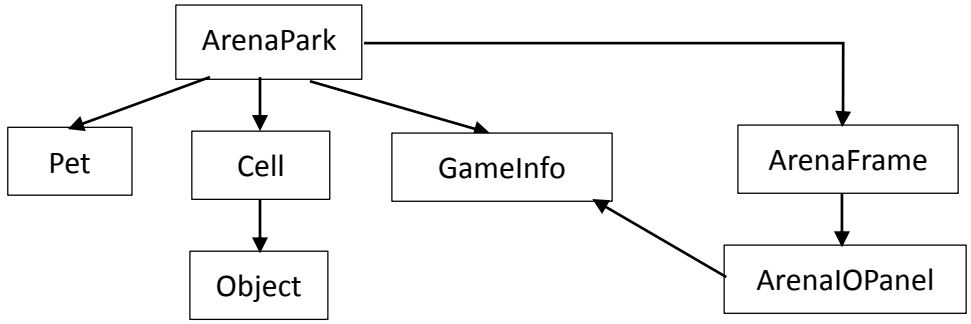
   Black box is a “Class”, and blue box is an “Interface.”



Each given class with prefix “POO” is inherited by a more suitable one for my own game except for POOAction and POOFight. As you see in the above graph, I implement these classes with interface DImage or VaporizableDImage.

In fact, class POOAction and method act are not really used in the game despite that I do make a concrete method in Pet. They are not compatible with Class Pet because of my AI and Player Control design. For each time step in ArenaPark, it will call method oneTimeStep() for every pet, and oneTimeStep() will call Strategy() if it’s controlled by AI or call Player() otherwise. Method strategy is where I put AI.

(2) Part of “Has A” relations



This is a simple sketch of “Has A” relations. Class ArenaPark which has a class GameInfo is the controller of the game. Class ArenaFrame and class ArenaIOPanel, as their names, are Frame and Panel to show on screen. Class GameInfo is the required game information shared by ArenaPark and ArenaIOPanel. Class Cell store information of each grid in the game map, such as type of Object and list of skills. Besides, Cell has a class Object which can be any class. In fact, Object usually is a Pet or an Obstacle.

2. Advantages in terms of software engineering

The most important advantage is that this game is easy to expand because of the structure of inheritance and the use of abstract methods. Three examples are shown as follows.

(1) To add new kinds of Pets

First, extend class Pet and implement all needed abstract methods.

Second, draw images.

Finally, test for bugs.

(2) To add new kinds of Arenas

First, extend class Arena and implement methods. If we have to draw images while implementing the methods, call methods of ArenaFrame.

Second, draw images.

Finally, test for bugs.

(3) To add new kinds of Skills

First, extend class Arena and implement methods.

Second, draw images and add this skill in skill list of the indicated kind of pet.

Finally, test for bugs.

As you see from the above example, the use of abstract methods will remind developers what they have to implement. Besides, separating IO and control of the game will make things easier.

3. Disadvantages in terms of software engineering

   I have mentioned that abstract methods and classes are helpful while developing. However, abstract static methods cannot appear in Java. So, developers may forget to implement static methods. Moreover, variable encapsulation is a problem. The first developer to implement abstract class Pet, Arena, and Skill should implement a lot of variable interfaces for private variables, which is a huge task. And accessing private variables through such interfaces is a painful thing.

4. Advantages in terms of interestingness of the game

   This design make the game have more possibility and easier to maintain.

   (1) There can be layers of outputs on screen.

   Since IO and control of the game are separated, the interface provided can easily make the following effect.

   background -> background units -> units -> foreground units -> Fog of War

   (2) There can be a lot of “things” in a single grid

   There can be many dead bodies, types of grass, skills and a pet or an obstacle in a single grid since a grid contains ArrayList instead of a single one.

   (3) Having a lot of kinds of skills is possible

   Skills can be put on ground, on pet. Skills can affect a single grid or more. Skills can move on the arena or even follow a pet.

5. Disadvantages in terms of interestingness of the game

   Since the IO and control of the game is separated, at least one between ArenaFrame and ArenaIOPanel has to rewrite if there is a significant change.

   (1) only a single player on ArenaPark

   (2) different display style

   This is because that original ArenaIOPanel provides only a fixed style of displaying game information.

6. Bonus and Game Introduction

   (1) Introduction

   There are three kinds of monsters live in a terrible park. The monsters are Slimes, Rock Arm and TorLion. The monsters will hunt each other. A successful kill will reward with HP and MP. The monsters will be in two different status, “angry” and “normal.” The monsters will become angry if there is a prey in their sight.

   Being angry means that the monster will have a HP gain and MP gain about

twice the original amount and a little gain on AGI. that is, they become extremely dangerous. The time stay on angry and time to restore angry differ among monsters. When monsters calm down or become normal, their HP and MP will reduce to original value and then add a little growth.

	Slime	Rock Arm	TorLion
HP	3	7	16
MP	3	7	10
AGI	12	15	20
Anger	11	30	35
Sight	3	6	8
Skill z	Tiny Attack Skill	Rock Sting	Tiny Attack Skill
Skill x	None	None	Tornado
special	gain sight range with higher HP and gain AGI on every kill	gain sight range on every kill	gain sight range with lower HP

(2) Introduction about Monsters and AI behavior

Slime, a low form of life, is the weakest monster in the park. Slime will dash to enemy on seeing them without any strategy. Slime will not attack Slime and will always try to be angry to grow HP as fast as it can

Rock Arm, a brutal monster, will smash anything in sight until HP or MP is low.

TorLion, an extremely dangerous and cunning hunter, will hunt closest prey in sight until HP or MP is low. Even if its HP or MP is low, it will follow its prey until it gain back HP or MP.

(3) Game option

The game is very fair because what AI can see is the same as what player can. But for shamed player, press F1 can remove fog of war and press F2 to give control to AI.

Change the first type of pet in run.sh to change monster. Default is TorLion.

(4) Control: This game only use keyboard

Arrow keys to move.

Press “z”, “x” to use skill of certain pet.

Press “Space” to be angry or restart if you win or lose.

Press “Enter” to restart if you win or lose.

Press “F1” (cheat) to change between having fog of war of not.

Press “F2” (cheat) to change between control by AI or player.

Press “F9” to restart the game.

Press “Esc” to exit the game.