Game API Design Specification

Team Sleepy Day

(URL: https://pacman-team-sleepyday.herokuapp.com)

Part One. Description of interfaces/abstract classes

Abstract classes:

GameObj: This abstract class is the superclass of all entity objects of the game.

Fields	
Modifier and Type	Field and Description
Optional<String>	color the color of the object in the game
Optional<Integer>	size the size of the object in the game
Optional<String>	src source image of the object in the game
String	type the type of the object in the game
int	score player's score during the game
Methods	
Modifier and Type	Method and Description
String	getType() Return the type of object
int	getScore() Return player's score

Optional<String>	getColor() Return object's color
void	setColor() Set object's color
void	setSize() Set object's size
void	setScore() Set player's score
Optional<Integer>	getSize() Get game object's size
Optional<String>	getSrc() Get game object's source

EdibleObj: This class is the superclass of all game objects that can be eaten (fruits, energizer, small dots)

Methods	
Modifier and Type	Method and Description
void	setEaten() Add score from eaten objects

MovingObj: This class is the superclass of all game objects that can move in the game. (ghosts, Pacman)

Fields	
Modifier and Type	Field and Description

int	speed speed of the moving object in the game
Direction	direction direction of the moving object in the game
Point	Location Location of the moving object in the game
boolean	Invincible If the player is using invincible skill
boolean	willTurn If the player is going to turn
Direction	newDirection The new direction the player is going to turn
Constructors	
MovingObj (int speed, Point location, AUpdateStrategy updateStrategy) int speed speed of moving object Point location location of moving object AUpdateStrategy updateStrategy pacman's strategy Create a moving object with specific strategy	
Modifier and Type	Method and Description
void	update() Update moving object location

void	reset() Return to its original position
boolean	detectCollision() detect collisions between this moving object and walls in the maze
Point	setLocation(Point location) set location of one moving object
void	getLocation() get location of one moving object
void	setSpeed(int speed) set speed of one moving object
void	setDirection(Direction d) set direction of one moving object
Direction	getDirection() return direction of one moving object
boolean	isInvincible() return if user is using invincible skill
void	setInvicible(boolean invincible) ser if user is using invincible skill
void	setWillTurn(boolean willTurn) set if user is going to turn

boolean	isWillTurn() return if user is going to turn
void	setNewDirection(Direction newDirection) set newDirection of moving object
Direction	getNewDirection() return new direction of moving object
AUpdateStrategy	getUpdateStrategy() return strategy of moving object
void	setUpdateStrategy() set strategy of moving object
boolean	innerCollision() detect collisions between moving objects

AUpdateStrategy This class is the super class of all concrete moving strategies class of ghosts in the game

Fields	
Modifier and Type	Field and Description
String	type Strategy type name

Method	
Modifier and Type	Method and Description

String	getType() get ghost's type
void	updateState(MovingObj self, AllEnv allEnv) update game characters state Parameter: MovingObj - The moving object AllEnv – The game environment

Interface:

ISkillStrategy This interface defines skills of Pacman in the game

Field	
Modifier and Type	Field and Description
String	name skill name
int	cd cool-down (cd) before another skill used
int	lasting lasting time of the skill
final int	maxCd max cool-down time
final int	maxLasting total time of one skill can last for

Player	caster player who has the skill
final int	cost cost of one skill
Constructors	
ISkillStrategy(int maxCd, int maxLasting, int cost) int maxCd max time for cooling down int maxLasting max time for skills last for int cost cost for applying the skill Create a skill for Pacman	
Methods	
Modifier and Type	Field and Description
void	cancel() cancel the specific skill
void	update() update it in each period to calculate the lasting and cd information
String	getName() return skill name
void	cast(Player player, AllEnv allenv) Parameter: Player – Pacman created by user Allenv – environment loaded in the game
int	getCd() returns skill's cd
void	setCd(int cd) set skill's cd

Player	getCaster() get caster player
void	setLasting(int lasting) set lasting of skill
void	setCaster(Player caster) set player as caster
int	getLasting() return skill's lasting time
int	getMaxCd() return skill's Max Cd
int	getMaxLasting() return skill's max lasting name
int	getCost() return skill's cost

IGameCmd This interface is implemented by all concrete commands, such as UpdateCmd.

Method	
Modifier and Type	Method and Description
void	Execute(MovingObj src) Executes the command on certain game object Parameter: MovingObj – The game moving object

Part Two. Use cases:

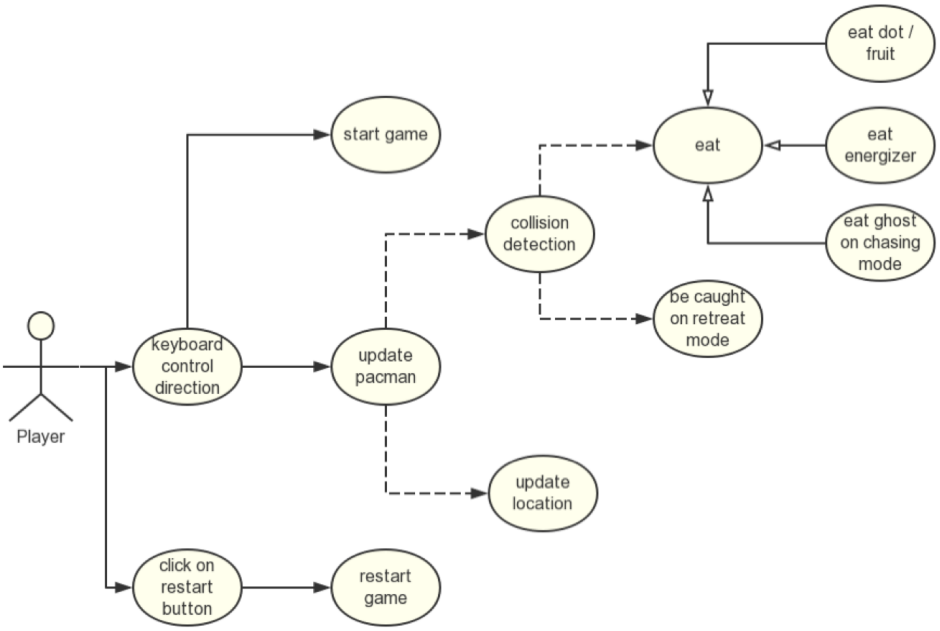
Pacman movement:

1. Users press any arrow key to start the game, and Pacman has three lives initially.
2. Users can control Pacman's direction in the maze by arrow key.
3. Pacman can move in the maze when there is at least one life left.
4. Pacman can eat small dots and earn 10 points.
5. Fruit will be generated at a random location of the map. Pacman can eat fruits and earn 100 points.
6. Pacman can eat energizer and earn 50 points. Energizer can set all the ghosts to be in retreat mode for a while.
7. Pacman can eat ghost in retreat mode and earn 200 points. After a ghost in retreat mode being eaten, it will become two eyes and go back to its initial position. Points earned by eating the same ghost will increase.
8. When Pacman collides with a moving ghost in chase mode, Pacman will be eaten, and Pacman will lose one life.
9. Pacman will be chased by different types of ghosts with different strategies when moving in the maze.
10. Pacman will keep moving after one arrow key being pressed by the user, and will stop if Pacman collides with a wall.
11. When all small dots are eaten, users will win and enter the next round.
12. When users run out of all remaining lives, the game will be over.
13. Users can use acceleration skills to move faster than usual.
14. Users can use invincible skills to avoid being eaten by ghosts.
15. Users can use teleport skills to move to a random location of the map.

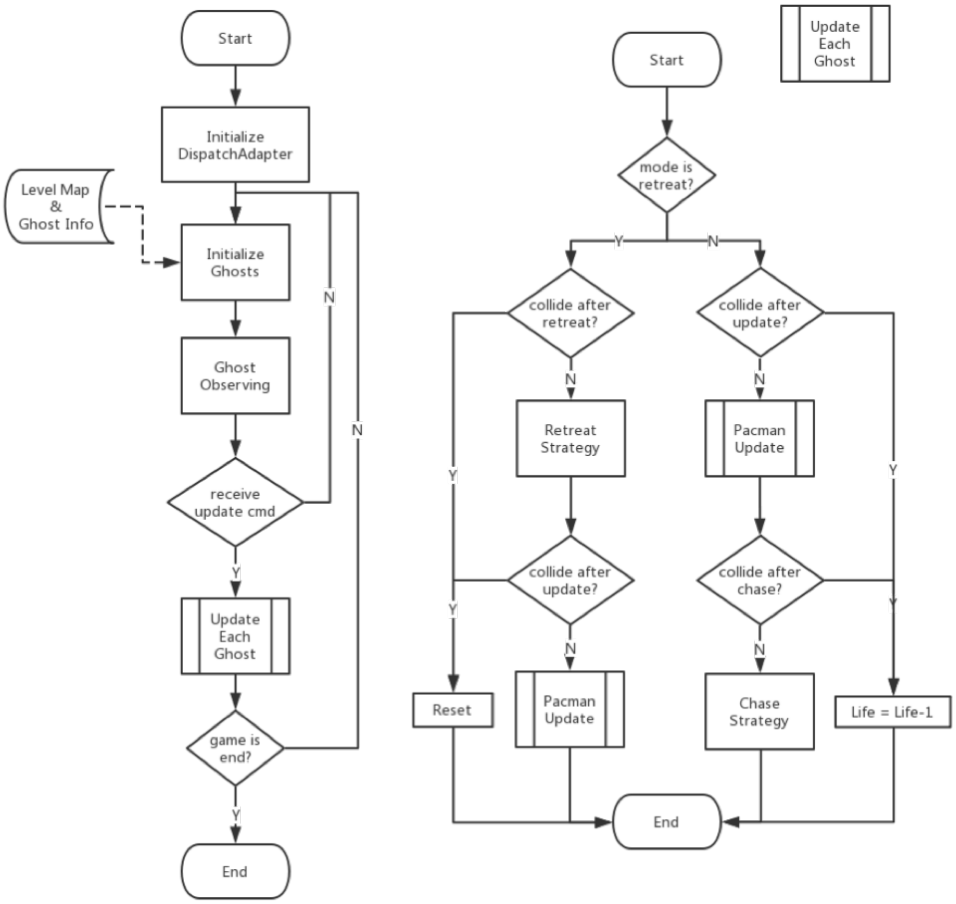
Ghost movement:

1. Ghost will be initially located in the middle of the map when users start the game.
2. There are four different types of ghosts moving in the map and chasing Pacman controlled by users.
3. Red ghost's speed will increase as users' score increases.
4. Yellow ghosts go around the blocks in the maze in a counterclockwise direction.
5. Blue ghosts go around blocks in a clockwise direction.
6. Pink ghosts move randomly in the maze and will only chase Pacman when very close to Pacman.
7. Ghosts will be in the retreat mode for five seconds when Pacman eats energizer.
8. Ghosts always find a way to approach Pacman in chase mode.
9. Ghosts move far away from Pacman in retreat mode.
10. Ghosts will appear in the middle of the map and begin to move after eaten by Pacman.

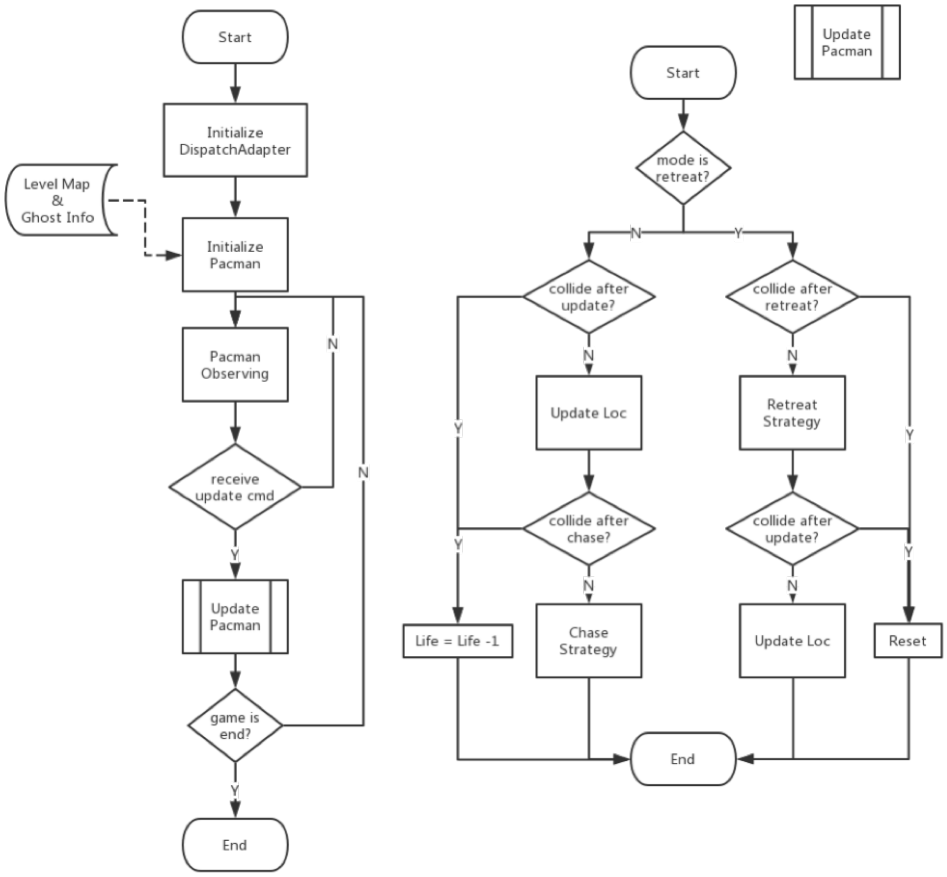
.



Use Case Diagram



Ghost movement flowchart



Pacman movement flowchart

Part Three. User extensible

1. More players and ghosts could be easily added to our game: in our model, since PropertyChangeSupport has a list of PropertyChangeListeners, adding any number of these listeners (players and ghosts) is free, where Player and Ghost are both subclasses of MovingObj that implements PropertyChangeListener.
2. Because the strategy pattern is used in our design, behaviors of any moving objects can be changed by replacing its moving strategy that implements IUpdateStrategy.
3. Extra module of magic skills is conveniently added into our project as well, since we design one ISkillStrategy for every Pacman, and Pacman's skills can be changed easily by concrete class that implements ISkillStrategy.
4. We also deployed Command pattern in our design, more commands except for updating information of Pacman or ghost could be added.
5. As long as the game level being increased, different mazes can be loaded. Since we use a 2-dimension matrix to represent the map and let every unit represent

multiple actual pixels, all we need to do is just passing a new matrix of map to the front-end and let the front-end draw it accordingly.