Pac-Man Quicksilver Ver

Bill Huang, Yang Zhou, Zhen Gao,

Jeff Zhou, Jianing Lou, Jerry Zhang, Zhengtong Liu

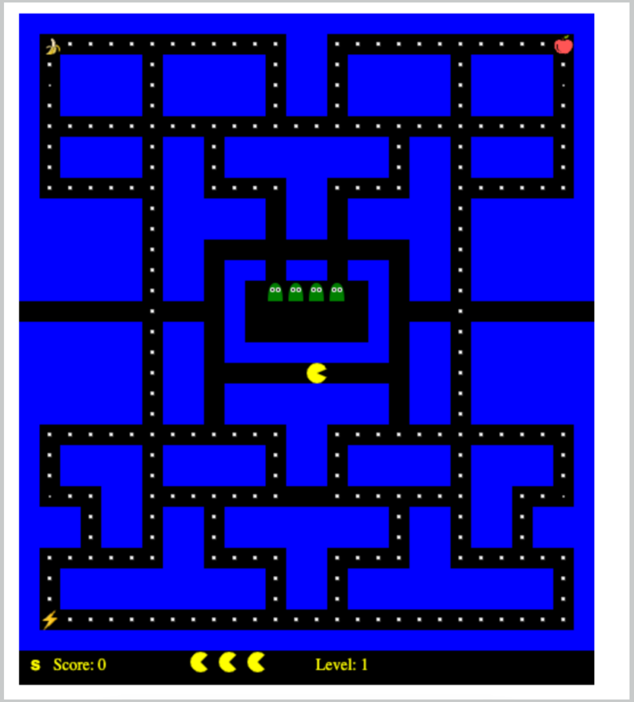
Agenda

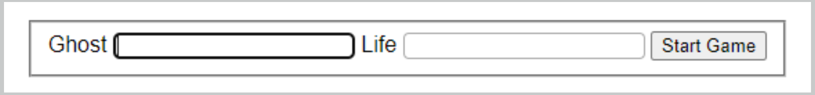
- App design
- Design patterns
- UI demo
- Q & A

App Design

Design features

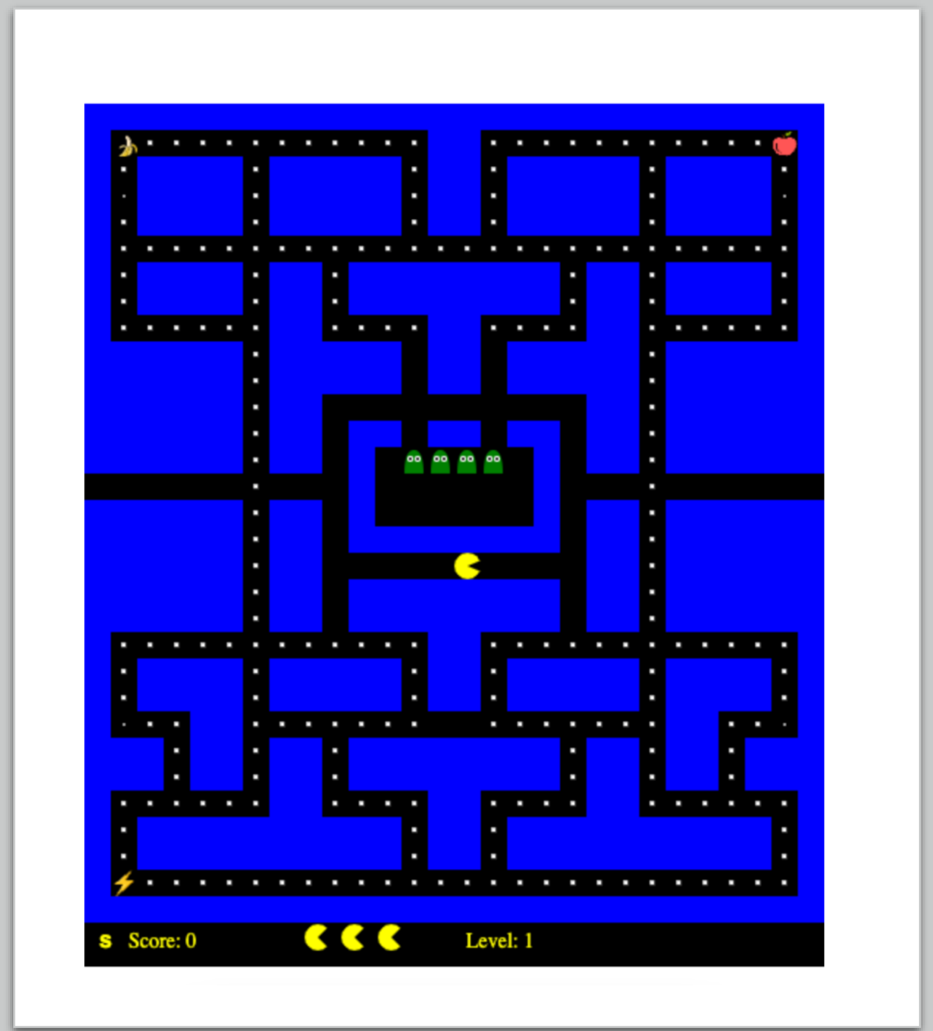
- Superpower of Quicksilver
- Update by tick
- No respawn, even if you lose a life
- Extensibility





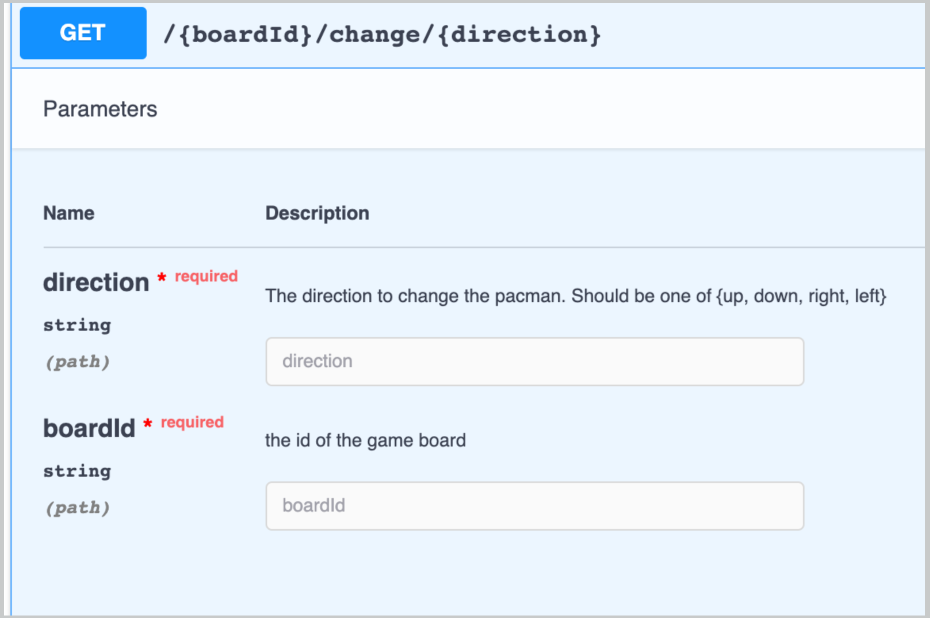
Endpoints

- **GET {boardId}/board**
  - Gets the board data
  - Board data passed in 2D array
  - Front End render the 2D array into game board.



Endpoints

- **GET {boardId}/change/direction**
  - Respond to user input(keyDown)
  - Change the direction of the pacman

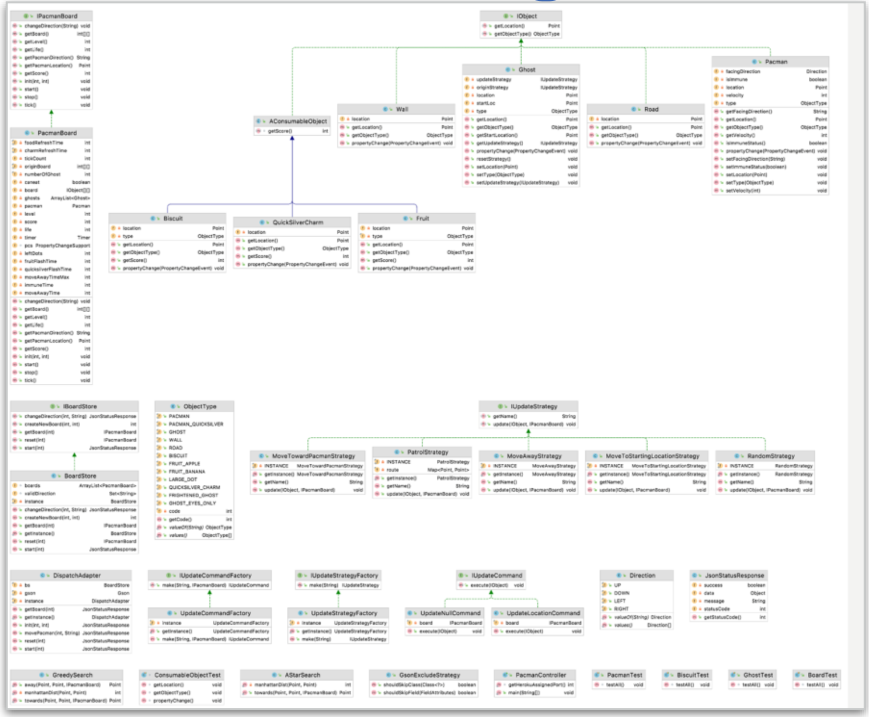


GET `/{boardId}/change/{direction}`

Parameters

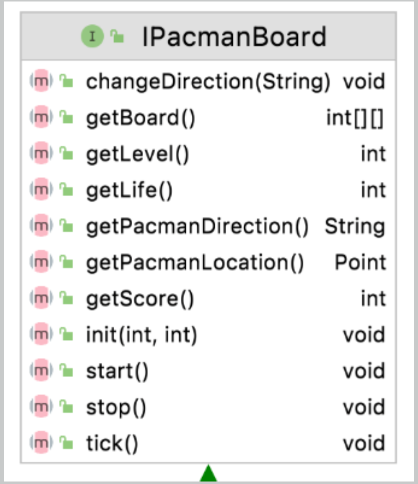
Name	Description
direction * required string (path)	The direction to change the pacman. Should be one of {up, down, right, left} direction
boardId * required string (path)	the id of the game board boardId

UML Diagram

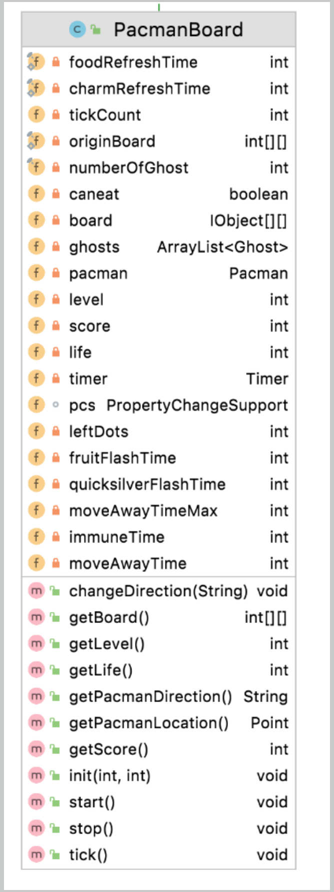

IPacmanBoard
PacmanBoard
IObject
AConsumableObject
Wall
Ghost
Road
Pacman
Biscuit
QuickSilverCharm
Fruit
IBoardStore
BoardStore
ObjectType
IUpdateStrategy
MoveTowardPacmanStrategy
PatrolStrategy
MoveAwayStrategy
MoveToStartingLocationStrategy
RandomStrategy
DispatchAdapter
IUpdateCommandFactory
UpdateCommandFactory
IUpdateStrategyFactory
UpdateStrategyFactory
IUpdateCommand
UpdateNullCommand
UpdateLocationCommand
Direction
JsonStatusResponse
GreedySearch
ConsumableObjectTest
AStarSearch
GsonExcludeStrategy
PacmanController
PacmanTest
BiscuitTest
GhostTest
BoardTest

Board

- Stores all the elements in one game
- Responsible for maintaining the states for the elements
- Responsible for maintaing the game logic inside tick()
- In tick, every movable objects moves towards specified direction and then do collision detection, etc.



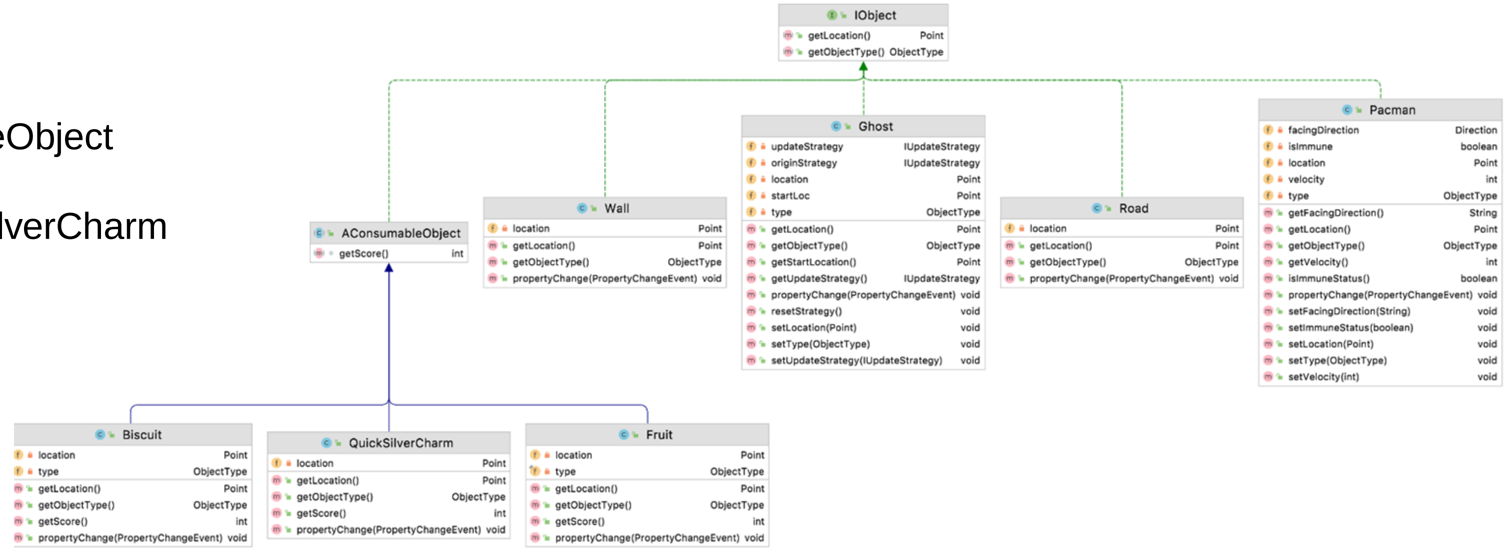
IPacmanBoard	
changeDirection(String)	void
getBoard()	int[][]
getLevel()	int
getLife()	int
getPacmanDirection()	String
getPacmanLocation()	Point
getScore()	int
init(int, int)	void
start()	void
stop()	void
tick()	void



PacmanBoard	
foodRefreshTime	int
charmRefreshTime	int
tickCount	int
originBoard	int[][]
numberOfGhost	int
caneat	boolean
board	IObject[][]
ghosts	ArrayList<Ghost>
pacman	Pacman
level	int
score	int
life	int
timer	Timer
pcs	PropertyChangeSupport
leftDots	int
fruitFlashTime	int
quicksilverFlashTime	int
moveAwayTimeMax	int
immuneTime	int
moveAwayTime	int
changeDirection(String)	void
getBoard()	int[][]
getLevel()	int
getLife()	int
getPacmanDirection()	String
getPacmanLocation()	Point
getScore()	int
init(int, int)	void
start()	void
stop()	void
tick()	void

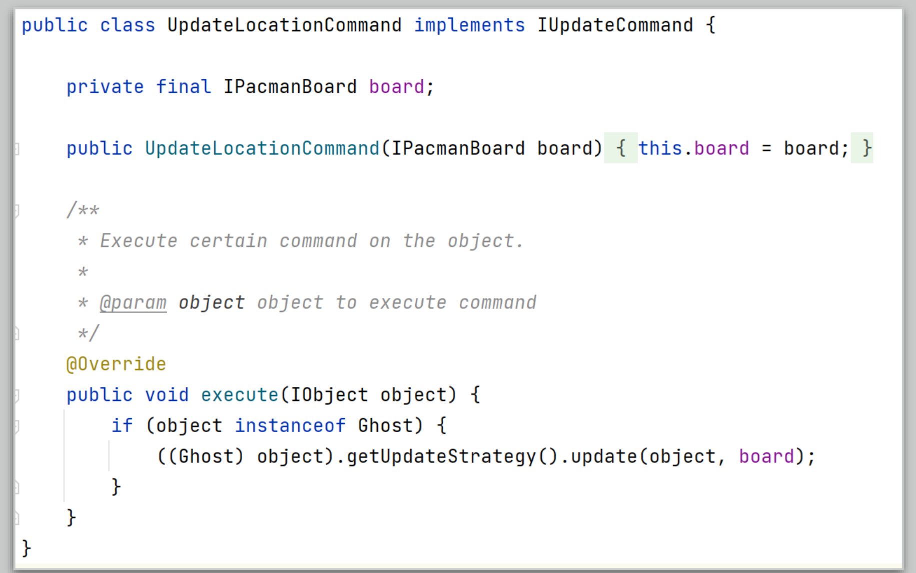
Objects

- IObject
- AConsumableObject
  - Biscuit
  - QuickSilverCharm
  - Fruit
- Wall
- Ghost
- Road
- Pacman


IObject
getLocation() Point
getObjectType() ObjectType
AConsumableObject
getScore() int
Wall
location Point
getLocation() Point
getObjectType() ObjectType
propertyChange(PropertyChangeEvent) void
Ghost
updateStrategy IUpdateStrategy
originStrategy IUpdateStrategy
location Point
startLoc Point
type ObjectType
getLocation() Point
getObjectType() ObjectType
getStartLocation() Point
getUpdateStrategy() IUpdateStrategy
propertyChange(PropertyChangeEvent) void
resetStrategy() void
setLocation(Point) void
setType(ObjectType) void
setUpdateStrategy(IUpdateStrategy) void
Road
location Point
getLocation() Point
getObjectType() ObjectType
propertyChange(PropertyChangeEvent) void
Pacman
facingDirection Direction
isImmune boolean
location Point
velocity int
type ObjectType
getFacingDirection() String
getLocation() Point
getObjectType() ObjectType
getVelocity() int
isImmuneStatus() boolean
propertyChange(PropertyChangeEvent) void
setFacingDirection(String) void
setImmuneStatus(boolean) void
setLocation(Point) void
setType(ObjectType) void
setVelocity(int) void
Biscuit
location Point
type ObjectType
getLocation() Point
getObjectType() ObjectType
getScore() int
propertyChange(PropertyChangeEvent) void
QuickSilverCharm
location Point
getLocation() Point
getObjectType() ObjectType
getScore() int
propertyChange(PropertyChangeEvent) void
Fruit
location Point
type ObjectType
getLocation() Point
getObjectType() ObjectType
getScore() int
propertyChange(PropertyChangeEvent) void

Commands

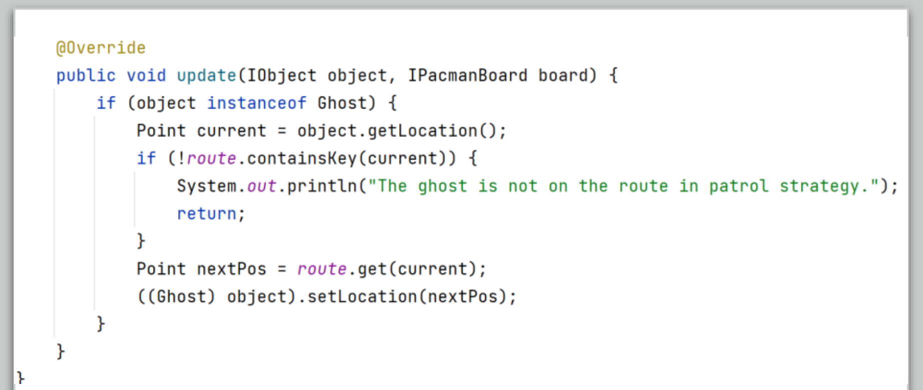
- interface IUpdateCommand
- Class UpdateLocationCommand
- Class UpdateNullCommand
- Interface IUpdateCommandFactory
- Class UpdateCommandFactory



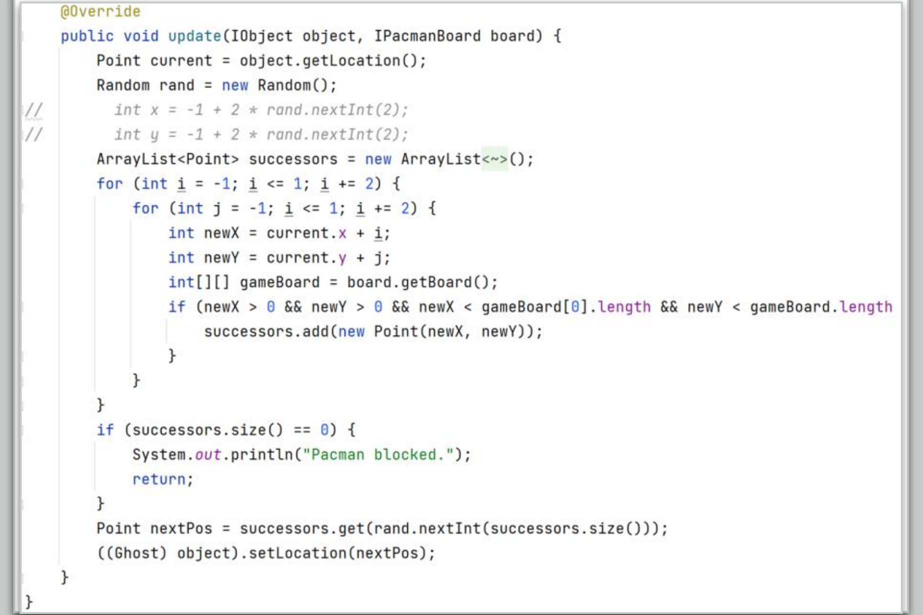
```java
public class UpdateLocationCommand implements IUpdateCommand {

    private final IPacmanBoard board;

    public UpdateLocationCommand(IPacmanBoard board) { this.board = board; }

    /**
     * Execute certain command on the object.
     *
     * @param object object to execute command
     */
    @Override
    public void execute(IObject object) {
        if (object instanceof Ghost) {
            ((Ghost) object).getUpdateStrategy().update(object, board);
        }
    }
}
```

Strategies

- interface IUpdateStrategy
  - Class MoveAwayStrategy
  - Class MoveToStartingLocationStrategy
  - Class MoveTowardPacmanStrategy
  - Class PatrolStrategy
  - Class RandomStrategy
- interface IUpdateStrategyFactory
- class AStarSearch
- class GreedySearch
- class UpdateStrategyFactory



```java
    @Override
    public void update(IObject object, IPacmanBoard board) {
        if (object instanceof Ghost) {
            Point current = object.getLocation();
            if (!route.containsKey(current)) {
                System.out.println("The ghost is not on the route in patrol strategy.");
                return;
            }
            Point nextPos = route.get(current);
            ((Ghost) object).setLocation(nextPos);
        }
    }
}
```



```java
    @Override
    public void update(IObject object, IPacmanBoard board) {
        Point current = object.getLocation();
        Random rand = new Random();
//        int x = -1 + 2 * rand.nextInt(2);
//        int y = -1 + 2 * rand.nextInt(2);
        ArrayList<Point> successors = new ArrayList<~>();
        for (int i = -1; i <= 1; i += 2) {
            for (int j = -1; i <= 1; i += 2) {
                int newX = current.x + i;
                int newY = current.y + j;
                int[][] gameBoard = board.getBoard();
                if (newX > 0 && newY > 0 && newX < gameBoard[0].length && newY < gameBoard.length
                    successors.add(new Point(newX, newY));
                }
            }
        }
        if (successors.size() == 0) {
            System.out.println("Pacman blocked.");
            return;
        }
        Point nextPos = successors.get(rand.nextInt(successors.size()));
        ((Ghost) object).setLocation(nextPos);
    }
}
```

Design Patterns

1. **Strategy Design Pattern**

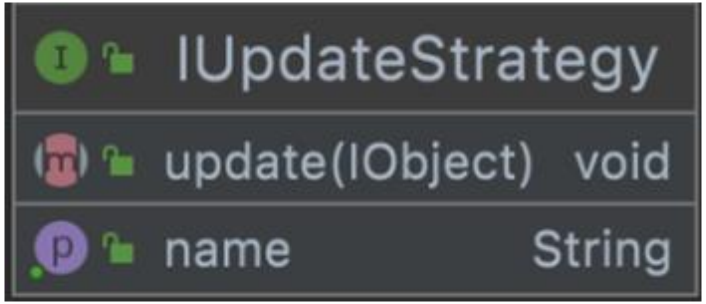
2. **Command Design Pattern**

3. **Observer Design Pattern**

4. **Factory Design Pattern**

5. **Singleton Design Pattern**

1. Strategy Design Pattern

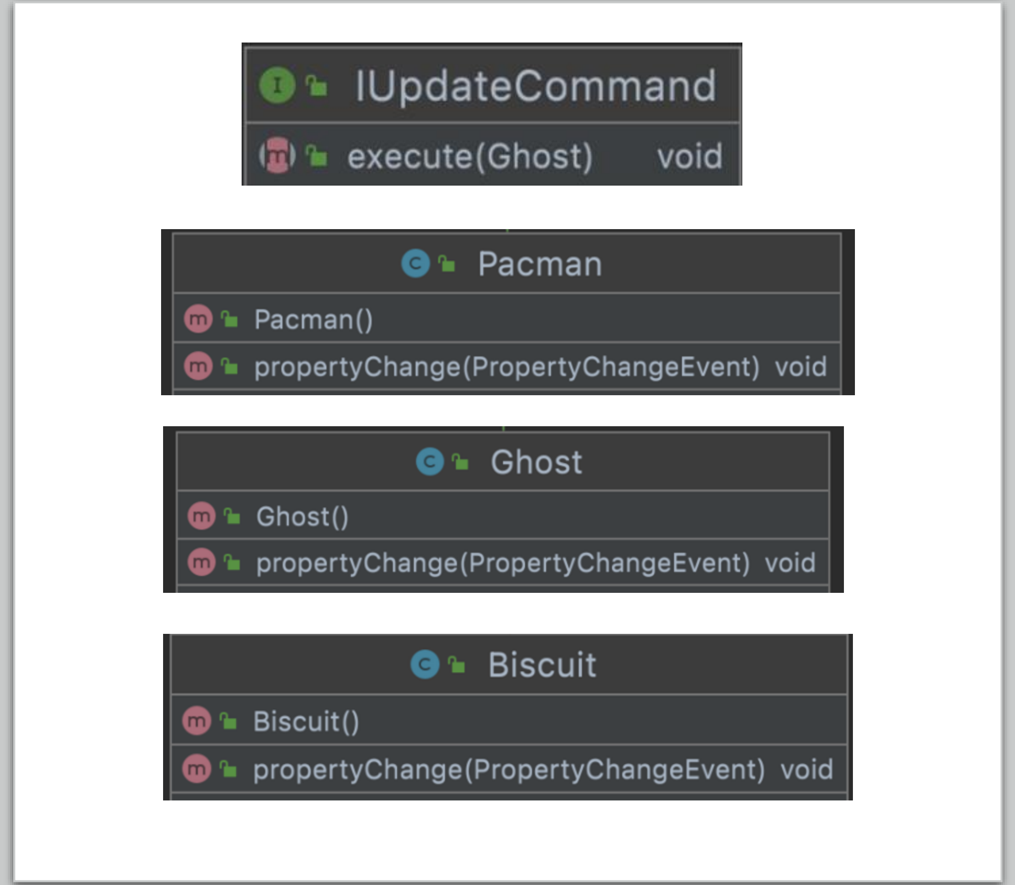

IUpdateStrategy

update(IObject) void

name String

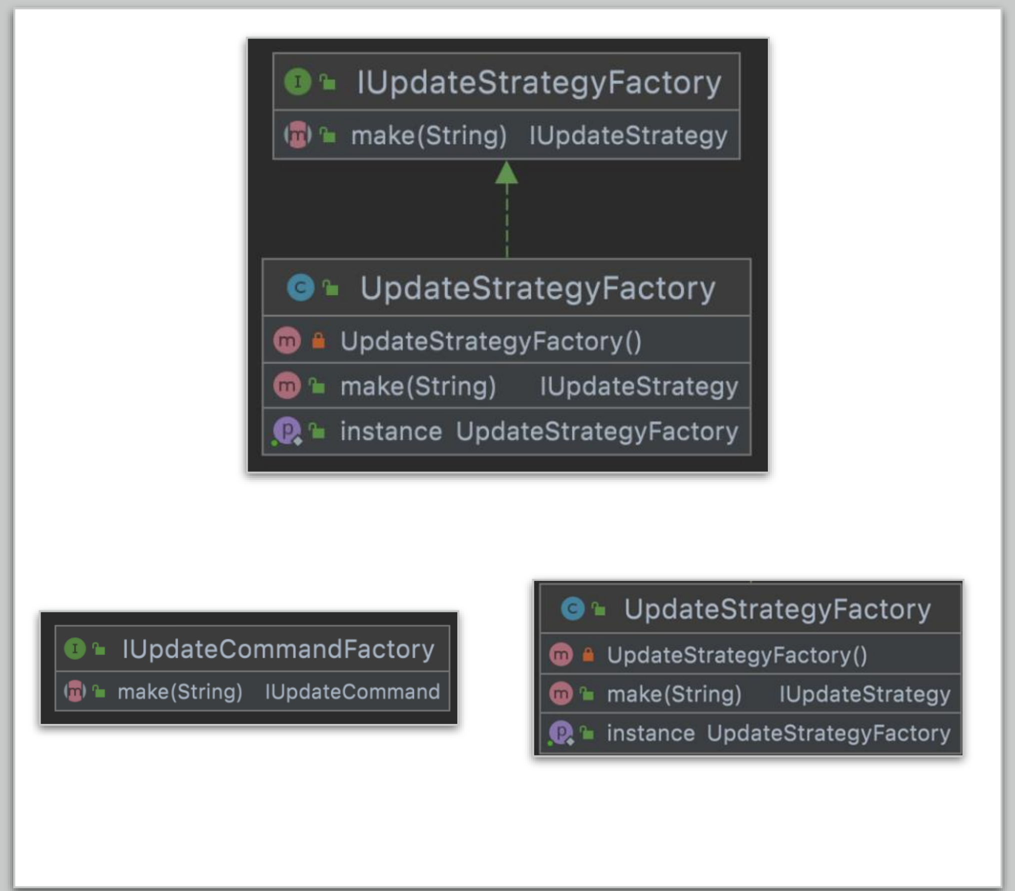
2. Command Design Pattern

3. Observer Design Pattern


IUpdateCommand
execute(Ghost) void
Pacman
Pacman()
propertyChange(PropertyChangeEvent) void
Ghost
Ghost()
propertyChange(PropertyChangeEvent) void
Biscuit
Biscuit()
propertyChange(PropertyChangeEvent) void

4. Factory Design Pattern

5. Singleton Design Pattern


IUpdateStrategyFactory
make(String) IUpdateStrategy
UpdateStrategyFactory
UpdateStrategyFactory()
make(String) IUpdateStrategy
instance UpdateStrategyFactory
IUpdateCommandFactory
make(String) IUpdateCommand
UpdateStrategyFactory
UpdateStrategyFactory()
make(String) IUpdateStrategy
instance UpdateStrategyFactory

Demo

Q&A

Thanks for listening！