



RICE UNIVERSITY  
Department of Computer Science

---

# API DOCUMENTATION FOR PACMAN

---

COMP 504 Group Project 2

**Professor:** Dr. Mack Joyner, Dr. Stephen Wong

**Team Name:** Good Team

**Team Member:**

**Team Lead:** Ying-Hsuan Chen

**Tech Lead:** Steven Li

**Documentation Lead:** Yan Xu

**Developer:** Zewen Xu; Serena Chen; Yuxi Liang; Jiacheng Sun

**Nov. 15, 2021**

## Overview

This project builds on the classic PacMan game. It keeps the features with "PacMac", "walls", "ghost", and "eating items", and it applies the original rules, such as "PacMac" will lose one life once it meets a "ghost". Different from the classic version, this project used the abstract class and interfaces to increase its extensibility. Therefore, a user can increase the difficulty level by adding more paths and more ghosts. For the project design, Java is used to support the backend, while the frontend uses JavaScript, HyperText Markup Language (HTML), and Cascading Style Sheets (CSS). The Demo link is <https://pacman-api-team-good-team.herokuapp.com/>. This Application Programming Interface (API) documentation introduces three parts: (1) use case; (2) game rules; (3) Unified Modeling Language (UML) diagram; (4) API endpoints.

## Table of Contents

<b>1. Use Case .....</b>	<b>1</b>
<b>2. Game Rules .....</b>	<b>1</b>
<b>3. UML Diagram and Class Explanation .....</b>	<b>3</b>
<b>2.1. Model.....</b>	<b>4</b>
2.1.1. AGameObj .....	4
2.1.2 Strategy.....	6
2.1.3 Command .....	7
2.1.4. GameBoard.....	7
<b>2.2. Adapter .....</b>	<b>8</b>
<b>3. API Endpoints.....</b>	<b>8</b>

## 1. Use Case

The use case is separated into three parts: (1) start a game, and (2) play the game - move Pac-Man; (3) end the game, Fig. 1. To begin with, a user starts the game by clicking the "start" button. Once it starts, the user can move the Pac-Man by using the arrow buttons on the keyboard. Finally, a user can end the game in three ways: (a) end the game after winning the current round, where the user will be pushed to the next higher difficulty level of the game automatically; (b) end the game after losing the current round, and then the user will be asked to join a new round at the current difficulty level; (c) end the game by close the webpage.

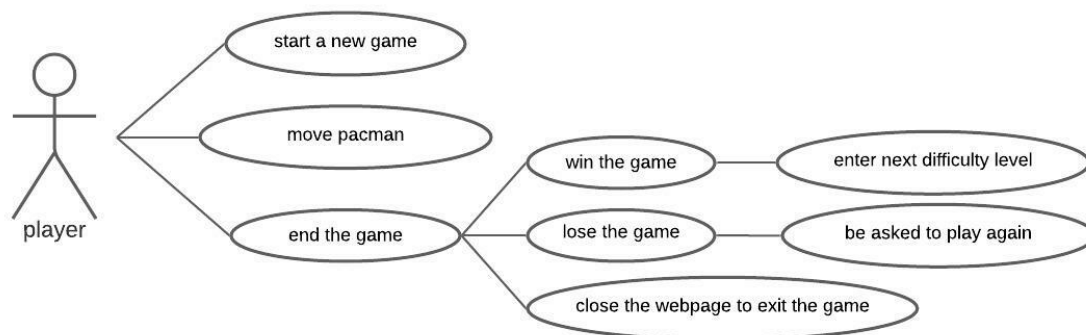


Fig. 1 Use case for the Chat APP

## 2. Game Rules

Like the classic Pac-Man game, this game contains two kinds of characters (Pac-Man and ghosts) and three eating items (big dots, small dots, and fruits). Table 1.1 - Table 2.3 are the detailed rules for each item.

Table 1.1 Moves instructions for Pac-Man

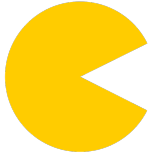




Pac-Man Image	Control button	Description
		Pac-Man moves to the right
		Pac-Man moves down
		Pac-Man moves to left
		Pac-Man moves up

Table 1.2 Interaction rules for Pac-Man

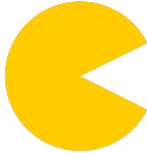

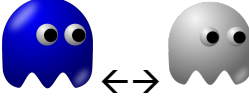

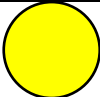

Pac-Man Image	Meeting Items		Description
	Image	name	
		normal ghost	Pac-Man will lose one life
		scared ghost	Pac-Man eats scared ghosts and earns points
		fruit	Pac-Man eats scared ghosts and earns points
		big dot	Pac-Man eats big dots and earn points
		small dot	Pac-Man eats small dots and earn points

Table 2.1 Movement of normal ghosts












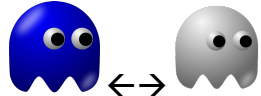

Ghost Image	Ghost Name	Description
	Random Ghost	Its moving directions are random
	Waiting Ghost	It first moves randomly and will start to follow Pac-Man after Pac-Man is approaching (gets close to it)
	Chasing Ghost	It follows Pac-Man
	Speeding Ghost	It moves randomly at a slow speed and then starts to move fast after eating a ghost

Table 2.2 Interaction rules for ghost

Image	Name	Change Events	Description
	normal ghosts	after Pac-Man  eats a big yellow dot 	be scared and will turn blue  , and then start to flash  ↔  for a while
		can not be eaten by Pac-Man 	Pac-Man loses one life after meeting a ghost
	scared ghosts	can be eaten by Pac-Man 	Pac-Man eats scaring ghosts and earns points

### 3. UML Diagram and Class Explanation

The APP follows the Model, View, Control (MVC) design pattern, using union, command, and strategy design patterns. The Unified Modeling Language (UML) diagram overview is in Fig.

2.

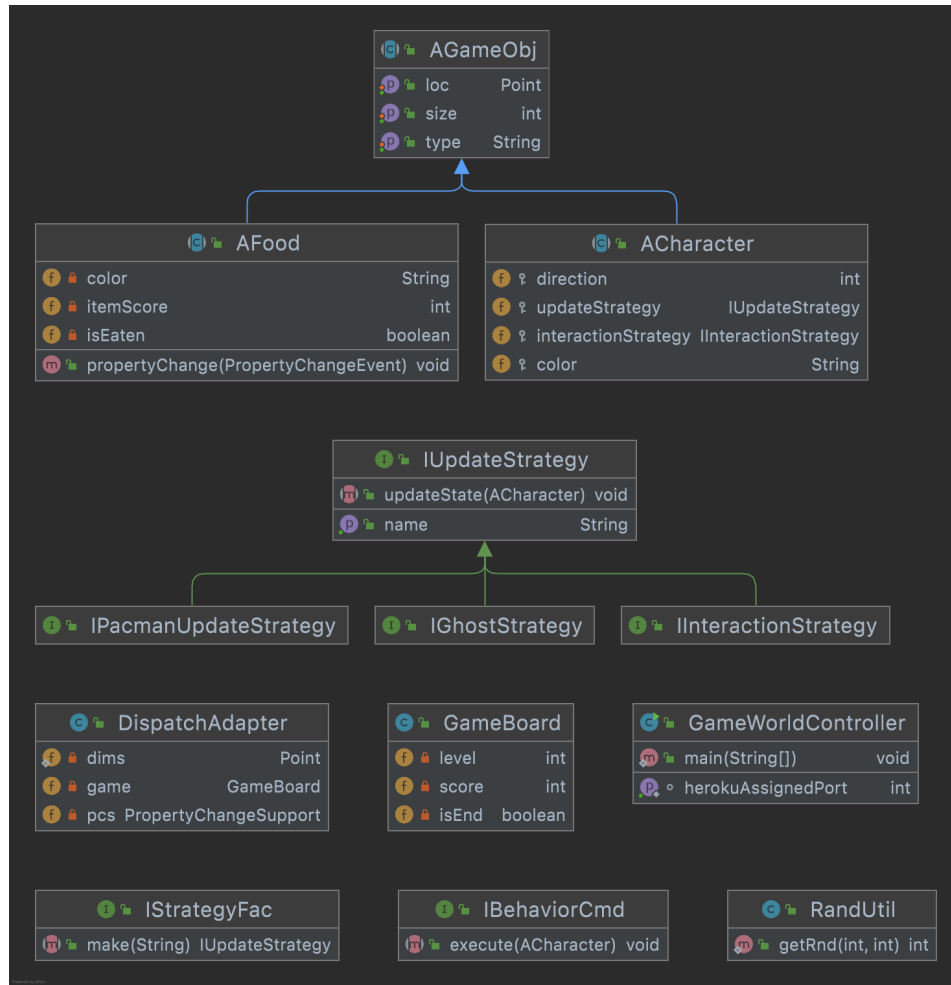


Fig. 2 The overview of the UML diagram

## 2.1. Model

For better implementation, the model section is separated into four sub-packages:

AGameObj (including A character and Afood), strategy, cmd, and GameBoard.

### 2.1.1. AGameObj

The "AGameObj" abstract class defines the features of all the objects in the map, including Pac-Man, ghosts, big dots, small dots, fruits, and walls. For better implementation, "AFood" abstract class and "ACharacter" class extend this AGameObj class. Fig. 3 shows the UML for three objects. Table 3.1 - 3.4 summarize the fields and methods of these three classes.

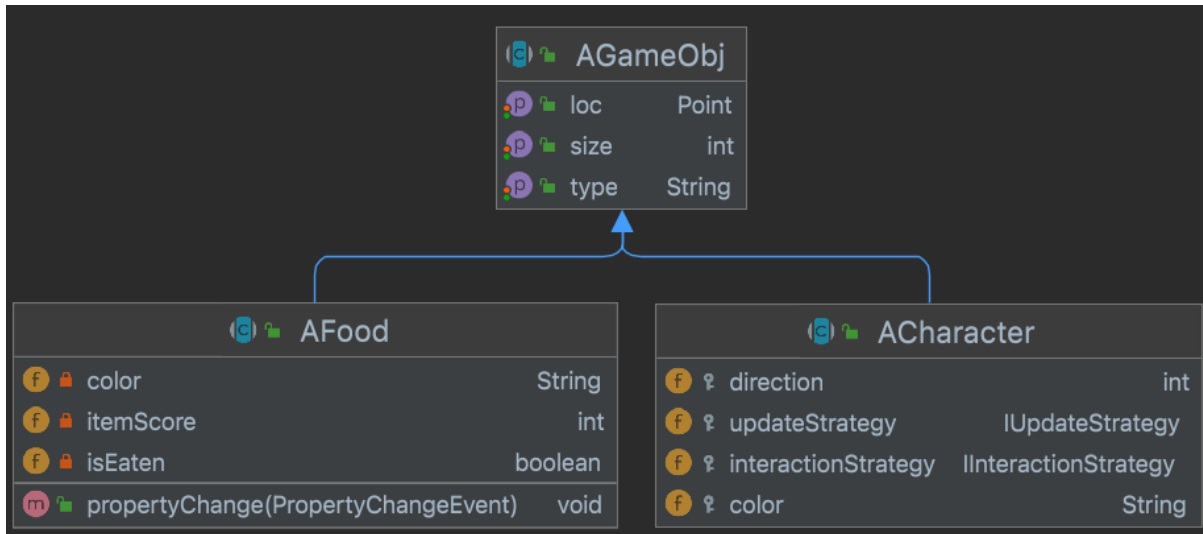


Fig. 3. UML for AGameObj, AFood, and ACharacter

Table 3.1. The fields in AGameObj

Access Level	Type	Field Name	Description
protected	Point	loc	the location of an object
protected	String	type	the type of object. i.e., "AFood" or "ACharacter"
protected	int	size	the size of an object

Table 3.2. The methods in Channel Class

Method name	Description
getLoc()	get the location
setLoc	set the location
getType()	get the type of this object
setType()	set the type of this object
getSize()	get the size
setSize()	set the size

Table 3.3. The fields in AFood

Access Level	Type	Field Name	Description
private	String	color	the color of the object
private	int	itemScore	the score earned after eating this object
private	Boolean	isEaten	tag of this object - whether it has been eaten or not

Table 3.4. The methods in Channel Class

Method name	Description
propertyChange(PropertyChangeEvent)	control property change using observe design pattern



Table 3.5. The fields in ACharacter

Access Level	Type	Field Name	Description
protected	int	direction	the color of the object
protected	IUpdateStrategy	updateStrategy	the update strategy
protected	IInteractionStrategy	interactionStrategy	the strategy for interaction
protected	String	color	the color of the object

### 2.1.2 Strategy

The strategy section contains a main interface "IUpdateStrategy", and three extended strategy interfaces: "IPacmanUpdateStrategy", "IGhostStrategy", and "InteractionStrategy". UML diagram for this design is presented in Fig.4, and table 4.1 summarize the corresponding method in IUpdateStrategy.

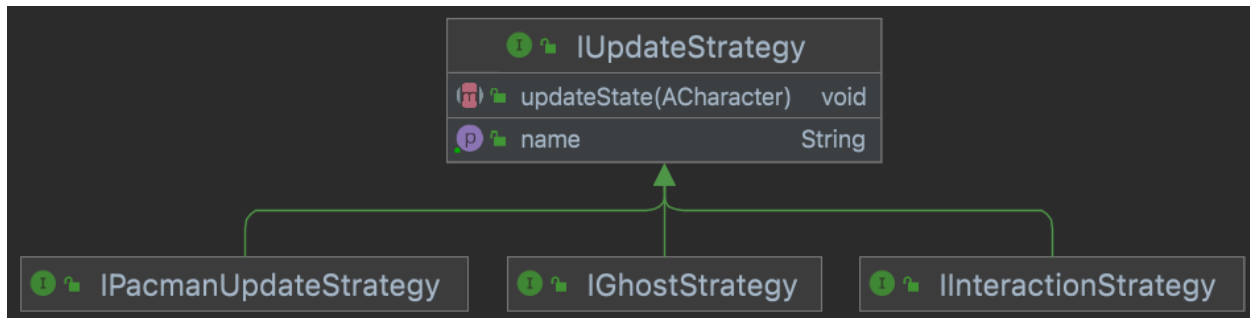


Fig. 4 UML for designing strategy interfaces

Table 4.1. The methods in IUpdateStrategy

Method name	Description
getName()	get the strategy name
updateState(ACharacter context)	update the state (i.e. color/location) of a character

In addition, the "IStrategyFac" interface helps to make a specific updateStrategy. The UML is presented in Fig.5, and table 4.2 shows the method.

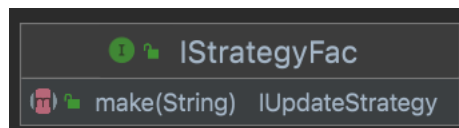


Fig. 5 UML for IStrategyFac

Table 4.2. The methods in IStrategyFac

Method name	Description
make(String strategy)	make a strategy

### 2.1.3 Command

This project also uses a command design pattern. Fig. 6 presents its UML, and table 5 summarizes the method for command interface "IBehaviorCmd".

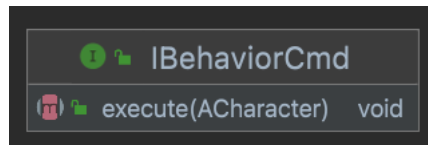


Fig. 6. UML for " IBehaviorCmd "

Table 5. The method in IBehaviorCmd

Method name	Description
execute(ACharacter context)	execute a context

### 2.1.4. GameBoard

The "GameBoard" organizes the operation of a character in one game. Fig.7 shows the structure of this class, and table 6 presents the detailed fields in this class.

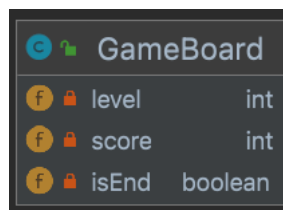


Fig. 7 UML for ChatAppWorld

Table 6. The fields in ChatAppWorld Class

Access Level	Type	Field Name	Description
private	int	level	the difficulty level of the current game
private	int	score	the score earned by the user
private	Boolean	isEnd	check whether the game is ended or not

## 2.2. Adapter

This APP uses an adapter ("DispatchAdapter" class) to connect the controller and model.

Fig. 8 and table 7 present the UML and fields of this class, respectively.

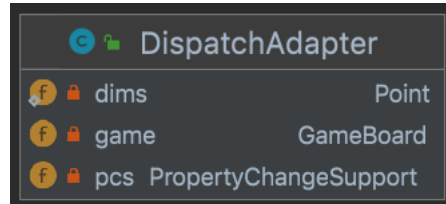


Fig. 8 UML of DispatchAdapter

Table. 7 The fields in DispatchAdapter

Access Level	Type	Field Name	Description
private	Point	dims	the dimension of the canvas
private	GameBoard	game	the gameBoard
private	PropertyChangeSupport	pcs	control the object events

## 3. API Endpoints

The API endpoints are in the controller. The detailed explanation is listed in table 8.

Table 8. Use case and the API explanation

Endpoint	Method	Use Case
"/initial"	get	The user starts a new game
"/update"	post	the AGameObject updates its movement
"/level/:id"	post	decide the difficulty level