Term Project Pac-Man

**[Absolute 10 marks]**

**OOP Spring 2022**

Dear Students

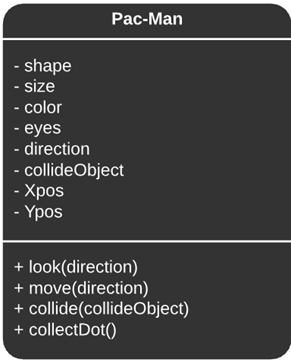
Please make a team of maximum 3 persons and start working on your term project Pac-Man.

*Note: any kind of plagiarism will result in an F grade in OOP.*

*The player controls Pac-Man through a maze, eating pac-dots or pellets. Four enemies [ghosts] roam the maze, trying to catch Pac-Man. Near the corners of the maze are four larger, flashing dots known as power pellets that provide Pac-Man with the temporary ability to eat the enemies. The enemies turn deep blue, reverse direction and usually move more slowly.*

# **Methods and Polymorphism**

Firstly, start by creating a class for the Pacman.



The three methods look, move, and collide are polymorphic and accept method overloading.

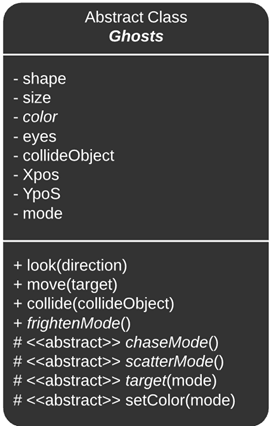
1. look(direction) : Based on user input, if move(direction) is not the same as look(direction), pacman changes look(direction) accordingly.
2. move(direction) : it accepts the direction the user enters through the input system and moves left/right/up/down accordingly.
3. collide(collideObject) : it accepts the property collideObject which is the object with which Pacman collides and functions accordingly.

*Coin : Ghosts enter FrightenMode*

*Ghost : If ghosts are in FrightenMode, Pacman gains points, the ghost gets eaten and returns to the GhostHouse, if not, Pacman gets eaten and the game restarts with one less life*

*Wall : If the Xpos and Ypos after moving are occupied by a wall, Pacman’s position remains unchanged.*

Then make the ghost class with basic properties like shape, color, eyes x-position and y-position and the mode.



As in the Pacman class, there are polymorphic methods along with 3 new methods named chaseMode(), frightenMode() and scatterMode().

setColor is also a method that accepts mode as the parameter, if the mode is Chase, then the method is left to the subclasses to define- which is why the method is abstract. If the mode is frightened then all ghosts are assigned the same color (blue).

# **Abstraction**

One thing to notice in that the ghost class is declared abstract with the attribute color and method target() is as abstract.

Now, one thing to know is that abstraction can be achieved by two methods.

1. Abstract classes

As ghost is an abstract class, it can have methods that are not defined along with the declaration. Any subclass which inherits the ghost class MUST, therefore, have the function definition.

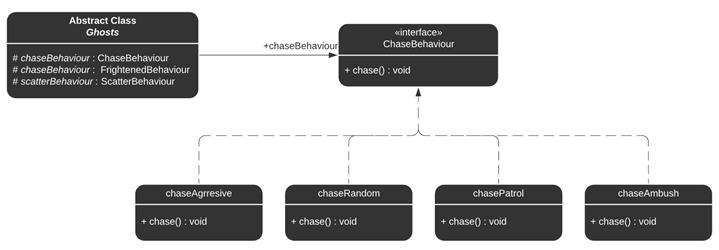
As the four ghosts have different methods of finding the target tile and then moving accordingly, the method is left abstract to be defined in the subclasses.

Now there are three modes a ghost can be in — Chase Mode, Scatter Mode, and Frighten Mode.

The ghost class implements the three interfaces ChaseBehaviour, FrightenBehaviour, and ScatterBehaviour.

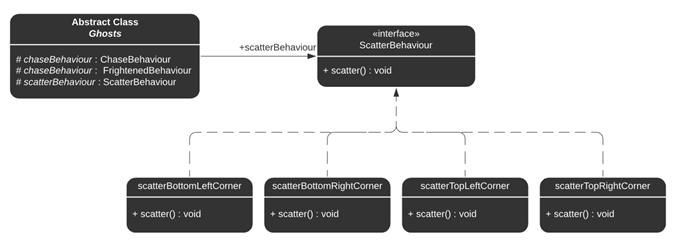
Also, as the ghost class is an abstract class, it need not implement the functions of the interfaces.

1. The ChaseBehaviour interface is used to define different ghostly behaviors during the chase mode of the game.



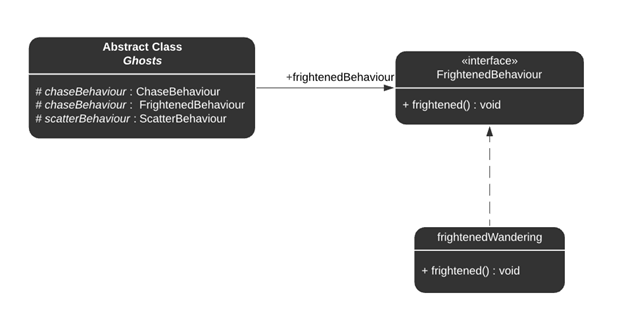
Chase Behavior Interface

2. The ScatterBehaviour interface is used to define different ghostly behaviours during the scatter mode of the Pac-Man game. In scatter mode, the ghosts give up the chase and head for their respective home corners.



Scatter Behavior Interface

3. The FrightenedBehaviour interface is used to define different ghostly behaviors during the frightened mode of the Pac-Man game. In frightened mode, the ghosts will all turn dark blue.

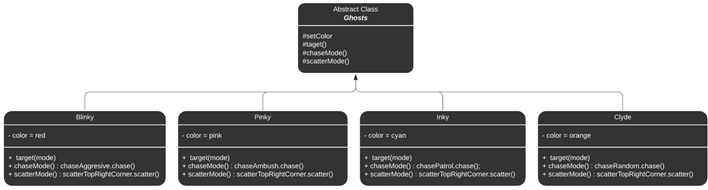


Scatter Behavior Interface

# Inheritance and abstract methods

Creating a new class from base/super class is called **inheritance**.The subclass inherits all the features from super class

Now that we are done with parent classes, let’s create the different ghosts as subclasses.



**Ghost subclasses**

We have 4 subclasses corresponding to the four ghosts-

* *Blinky the red ghost*
* *Pinky the pink ghost*
* *Inky the cyan ghost*
* *Clyde the orange ghost*

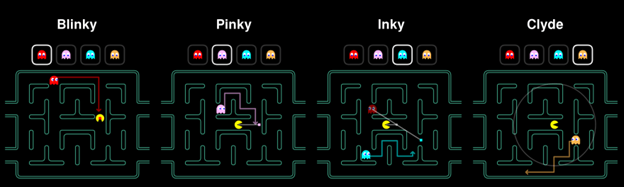
Firstly, they all inherit the basic properties and methods of the ghost class.

But each of them have different movement strategies in chaseMode and scatterMode which are declared as abstract in the parent class and hence different subclasses can implement different chase and scatter function according to the color.

The algorithm to find the target tile to move towards also depends on the color and hence the chase mode.

## Chase Mode

1. Blinky : It is very aggressive in its approach while chasing Pac-Man and will follow Pac-Man once located. : chaseAggresive()
2. Pinky : It will attempt to ambush Pac-Man by trying to get in front of him and cut him off. : chaseAmbush()
3. Inky : It will patrol an area and is not very predictable in this mode. : chasePatrol()
4. Clyde : It is moving in a random fashion and seems to stay out of the way of Pac-Man. : chaseRandom()

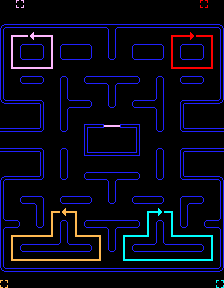


Chase Mode Visualized

## Scatter Mode

1. Blinky : It moves towards the top right corner : scatterTopRightCorner()
2. Pinky : It moves towards the top left corner : scatterTopLeftCorner()
3. Inky : It moves towards the bottom right corner : scatterBottomRightCorner()
4. Clyde : It moves towards the bottom left corner : scatterBottomLeftCorner()

This mode lasts only for a few seconds and then changes back to the “Chase” mode.



Scatter Mode Visualised