# Use Cases

## Pac-Man

1. Pac-Man’s direction is changed by direction keys.

* Covered by: setVel(Point vel) function in Character class.

1. Pac-Man does not move when colliding with a wall.

* Covered by: detectCollisionWithWall(Wall wall) function in AMovingElement class.

1. Pac-Man should be able to exit one side of the game board and enter on the other side of the game board.

* Covered by: boolean variable exit in Wall class.

1. Pac-Man loses 1 life when colliding with a non dark blue or non-flashing ghost.

* Covered by: detectCollisionWithGhost(Ghost ghost) function in AMovingElement class.

1. Pac-Man starts with 3 lives.

* Covered by: int variable life in gameStore class

1. Pac-Man advances to the next level if Pac-Man eats all the dots before losing 3 lives.

* Covered by: void loadGame(int level) in GameStore class.

## Ghosts

1. Ghosts will appear in the middle with different colors.

* Covered by: initializeGhosts() function in GameStore class.

1. Ghosts will use some behaviors to move towards the Pac-Man and some behaviors to move away from Pac-Man when the Pac-Man eats a fruit or a large dot.

* Covered by: updateState(AMovingElement character, AMovingElement context) function in Ghost0MovementStrategy, Ghost1MovementStrategy, Ghost2MovementStrategy, Ghost3MovementStrategy classes.

1. Ghosts turn dark blue and then start flashing when Pac-Man eats large dots.

* Covered by: int variable status in Ghost class will change to mark the ghost as dark blue or flashing dark blue.

1. Dark blue or flashing ghosts become two eyes and travel quickly to the square box in the middle of the screen when colliding with Pac-Man.

* Covered by: updateState(AMovingElement character, AMovingElement context) function in GhostReturningStrategy class.

## Board

1. There are 240 dots and 4 blinking big dots in corners.

* Covered by: It is stored and specified in the json file we used to store the map.

1. A piece of fruit will periodically appear on the board.

* Covered by: When the int variable timer in GameStore mod certain number equals 0, we use placeFruit(int type, int score) function to place different kinds of fruit.

1. The fruit or dots disappear when Pac-Man eats them.

* Covered by: detectCollisionWithDot(Dot dot) and detectCollisionWithDot(Dot dot) functions in AMovingElement class.

1. Game level, number of lives, and score are displayed on the board

* Covered by: the logic in the front end and the variables in GameSore class.

1. A Start game button and a Pause/Resume button are located at the bottom.

* Covered by: Start game button will call the gameRestart() function in GameStore class. Pause/Resume button will change the value of the gamePause variable in the front end, so the board will stop updating.

1. The game ends if Pac-Man loses all 3 lives. There should be a "Game Over" message shown on the game.

* Covered by: When the int variable in GameStore class equals 0, the front end will pop up a “Game Over” alert window.

1. Each level’s difficulty gradually increases.

* Covered by: We have maps with different size and complexity stored in json files for different levels.

1. The board keeps track of Pac-man's score for the game. The score doesn't need to be saved when a new game is started.

* Covered by: score variable in GameStore class.

1. The Pac-Man game should be extensible in some way that can be selected by a user.

* Covered by: We can manually select the level we want to play, so the int variable level in GameStore class will change

# API Specification

## APaintObj (Abstract Class)

The APaintObj is any given element with a given position and name in the pacmanWorldStore.

|  | **Member fields** |
| --- | --- |
| loc (Point) | The location of the object on the canvas. |
| name (String) | The name of the object. |

|  | **Methods** |
| --- | --- |
| Point getLoc() | Get the paint object location in the pac man world. |
| void setLoc(Point loc) | Set the paint object location in the canvas. The origin (0,0) corresponds to the top left corner of the canvas.  Parameters:   1. loc: The paint object coordinate. |
| String getName() | Get the paint object’s name. |
| void setName(String name) | Set the paint object’s name.  Parameters:   1. name: The new paint object’s name |

## AMovingElement (Abstract Class)

AMovingElement is any given moving element in the pacManWorld.

|  | **Member fields** |
| --- | --- |
| vel (Point) | The velocity of the element. |
| movementStrategy (IUpdateStrategy) | The element’s movement strategy. |
| collisionStrategy (ICollisionStrategy) | The element’s collision strategy. |
| Color (String) | The color of the element. |
| Size (String) | The size of the element. |

|  | **Methods** |
| --- | --- |
| String getColor() | Get the element’s color. |
| void setColor(String c) | Set the element’s color.  Parameters:   1. c: New color of the element. |
| Point getVel() | Get the velocity of the ACharacter. |
| void setVel(Point vel) | Set the velocity of the element.  Parameters:   1. vel: New velocity of the element. |
| IUpdateStrategy getMovementStrategy() | Get the element movementStrategy. |
| void setMovementStrategy(IUpdateStrategy movementStrategy) | Set the movementStrategy of the element.  Parameters:   1. movementStrategy: The new movementStrategy |
| ICollisionStrategy getCollisionStrategy() | Get the element’s collisionStrategy. |
| void setCollisionStrategy(ICollisionStrategy collisionStrategy) | Set the collisionStrategy of the element.  Parameters:   1. collisionStrategy: The new collisionStrategy |
| int getSize() | Get the size of the element. |
| void setSize(int size) | Set the size of the element.  Parameters:   1. size: The element’s size. |
| void move(Point dims) | Move the element.  Parameters:   1. dims: The canvas dims. |
| boolean detectCollisionWithWall(Wall wall) | Detects collision between an element and a wall in the pac man world.  Parameters:   1. wall: The wall to be detected. |
| boolean detectCollisionWithFruit(Fruit fruit) | Detects collision between an element and a fruit.  Parameters:   1. fruit: The fruit to be detected. |
| boolean detectCollisionWithGhost(Ghost ghost) | Detects collision between an element and a ghost.  Parameters:   1. ghost: The ghost to be detected. |
| void propertyChange(PropertyChangeEvent evt) | Items respond to property change events.  Parameters:   1. evt: Changed event. |

## AStationaryElement (Abstract Class)

AItem is any given stationary StationaryElements in the pacManWorld.

|  | **Member fields** |
| --- | --- |
| score (int) | The score of the element. |
| isEaten (transient boolean) | Whether the element is eaten. |

|  | **Methods** |
| --- | --- |
| int getScore() | Get the AItem score. |
| boolean isEaten() | Get if the AItem is eaten. |
| void setEaten(boolean eaten) | Set if the AItem is eaten.  Parameters:   1. eaten: If AItem is eaten. |
| void propertyChange(PropertyChangeEvent evt) | Items respond to property change events.  Parameters:   1. evt: Changed event. |

## IPacmanCmd (Interface)

The IPacmanCmd is an interface used to pass commands to objects in the Pacman. The MovingElements must execute the command.

|  | **Methods** |
| --- | --- |
| void execute(APaintObj context) | Execute the command.  Parameters:   1. context: The receiver paintobj on which the command is executed. |
| void execute(APaintObj context, APaintObj other); | Execute the command.  Parameters:   1. context: The receiver moving element on which the command is executed. 2. other: The paintobj to be collided with |

## ICollisionFac (Interface)

Factory for collision strategy.

|  | **Methods** |
| --- | --- |
| ICollisionStrategy make() | Make a new collision strategy. |

## IMovementFac (Interface)

Factory for moving elements.

|  | Methods |
| --- | --- |
| AMovingElement make() | Make a new moving element. |

## IStationaryFac (Interface)

Factory for stationary elements.

|  | **Methods** |
| --- | --- |
| AStationaryElement make() | Make a new moving element. |

## IUpdateFac (Interface)

Factory for moving strategy.

|  | **Methods** |
| --- | --- |
| IUpdateStrategy make() | Make a new movement strategy. |

## ICollisionStrategy (Interface)

An interface for element strategies that determine the element collision behavior with each other.

|  | **Methods** |
| --- | --- |
| String getName() | Get the name of the strategy. |
| IUpdateStrategy make() | Make a new movement strategy. |

## IUpdateStrategy (Interface)

The IUpdateStrategy interface is used to determine the behavior of a MovingElements in the canvas over time.

|  | **Methods** |
| --- | --- |
| void updateState(AMovingElement context) | Update the state of the MovingElements.  Parameters:   1. context: The MovingElements to apply the strategy to. |
| void updateState(AMovingElement character, AMovingElement context) | Update the state of the MovingElements.  Parameters:   1. character: the Pac-Man MovingElements to be compared with.. 2. context The MovingElements to apply the strategy to. |
| String getName() | Get the name of the strategy. |

# Design Decisions

| **Decision** | **Reason** |
| --- | --- |
| GameStore | Single point of storage for all the game objects like the APaintObjects and the Strategies. |
| Null Objects wherever necessary(for all strategies and paint objects) | To handle erroneous data and create a null object for all possible types of bad input so that the User Experience and UI isn’t affected. |

| **Design Pattern** | **Reason to use** |
| --- | --- |
| Strategy Design Pattern | To implement the behavior and collision of ghosts- how the ghosts behave when they are near each other and how the ghosts movements work. |
| Singleton Design Pattern | To store the factories and the game store since only one instance of all the factories and the store is required. |
| Factory Design Pattern | To create the movable and stationary objects and to create collision and movement strategies. |
| Command Design Pattern | To pass command of movement and collision to a ghost or a character during the game. |
| Composite Design Pattern | To create composite ghost movement and collision strategies. |