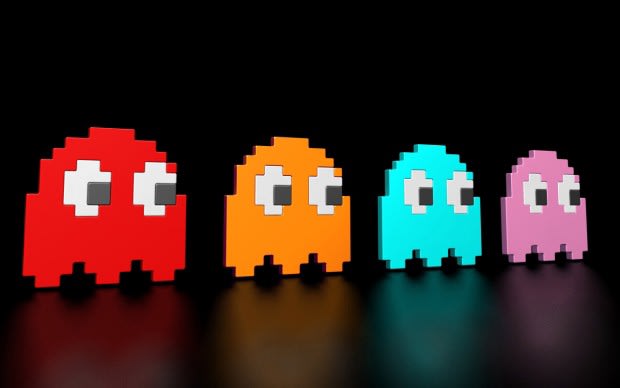
The idea of using Pac-Man to illustrate the usage of Design Patterns came to me while I was reading an article about Pac-Man by Jamey Pittman. His article was about the design and AI lessons within the classic Namco game Pac-Man. I was really amazed at the complexity in the design and AI of a game that seems so simple while playing at the time. I then decided to create a series of articles which illustrate the use of Design Patterns to implement some of the complex design issues in the game Pac-Man.

The first article in the “Pac-Man Patterns” series will focus on the different behaviour each of the four ghosts have during the different modes of the game. We will also look at how the Strategy Pattern can help with the design of the different movement behaviours.

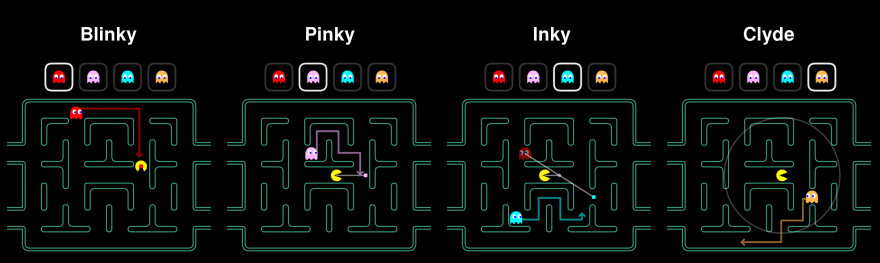
Pac-Man is an arcade game that was first release in 1980. The player navigates Pac-Man through a maze and has to collect all the dots (Pac-Dots) in order to complete the stage. Pac-Man is being chased by four ghosts in the game whose main objective is to kill him. The four ghosts, Blinky, Pinky, Inky and Clyde, each has different behaviour depending on the mode of the ghosts. The ghosts change mode during game play from scattering to the corners of the maze, to chasing Pac-Man and also to being frightened when Pac-Man picks up a Power-Pellet.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--2Uo1Z-4Q--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/cdn-images-1.medium.com/proxy/0%2A1GwJvRgVTwygmEfl.jpg)

This article discusses the different movements and behaviour of the ghosts in Pac-Man and how it relates to implementing Reusable Object-Oriented software also known as Design Patterns.

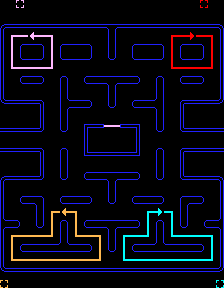
**Chase**

In “Chase” mode, the ghosts are trying to find and capture Pac-Man. Each of the four ghosts has a unique behaviour while chasing Pac-Man. Blinky the red ghost is very aggressive in its approach while chasing Pac-Man and will follow Pac-Man once located. Pinky the pink ghost will attempt to ambush Pac-Man by trying to get in front of him and cut him off. Inky the cyan ghost will patrol an area and is not very predictable in this mode. Clyde the orange ghost is moving in a random fashion and seems to stay out of the way of Pac-Man.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--tMDeCSH4--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/cdn-images-1.medium.com/proxy/0%2ACoMAiBNHxshA7HIw.png)

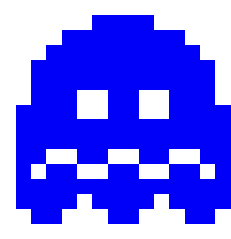
**Scatter**

In “Scatter” mode, the ghosts will stop chasing Pac-Man and each will move into its respective corners for a few seconds. Blinky the red ghost moves towards the top right corner, while Pink the pink ghost moves towards the top left corner. Inky the cyan ghost moves towards the bottom left corner and Clyde the orange ghost moves towards the bottom left corner. This mode lasts only for a few seconds and then changes back to the “Chase” mode.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--Rvtrl2Gc--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/cdn-images-1.medium.com/proxy/0%2AzNDnbLILf0WwS3co.png)

**Frightened**

The “Frightened” mode occurs when Pac-Man eats an energizer within the maze. There are four energizers located in the maze and all four ghosts change mode. The ghosts turn dark blue and wander around in the maze being vulnerable. They will flash moments before they return to either the Scatter or Chase mode.

[](https://res.cloudinary.com/practicaldev/image/fetch/s--AY3cmOy1--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/cdn-images-1.medium.com/proxy/0%2AXDKqZ9aXMYhUmXxc.png)

So, to sum up the movement requirements of the ghosts, the following table illustrates the types of movement and also how the individual ghosts behave during those types of movement.

Ghost Name Chase Scatter Frightened Blinky (Red) Aggressive Top Right Corner Wandering Pinky (Pink) Ambush Top Left Corner Wandering Inky (Cyan) Patrol Bottom Right Corner Wandering Clyde (Orange) Random Bottom Left Corner Wandering

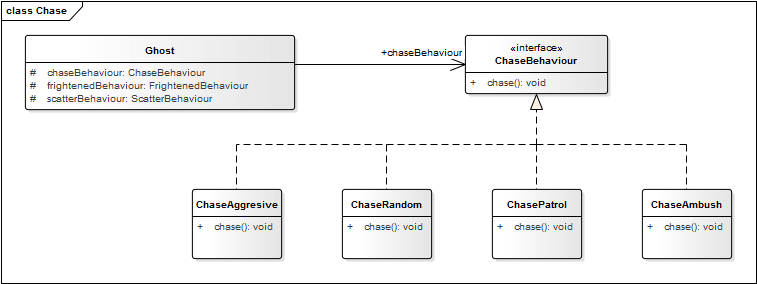
To implement the different ghost behaviours, the instances of the Ghost class will use the behaviour represented by an interface (ChaseBehaviour, ScatterBehaviour & FrightenedBehaviour) to ensure that the different implementations of each behaviour is not implemented within the Ghost class.

According to the Strategy Pattern, the behaviour that varies is placed into a separate class to allow you to make changes to those behaviours without affecting the parts that stays the same. Furthermore, the pattern aligns to the design principle to “Program to an Interface, and not to an Implementation” so that the three modes of a ghost can be defined as interfaces, and the different implementations of the modes of movement can be implemented in separate classes.

The Ghost class has a ChaseBehaviour, ScatterBehaviour and a FrightenedBehaviour. The HAS-A relationship refers to the composition of the classes. Now it is possible to compose the Ghost class to in such a way that each of the different types of ghosts can be composed of different implementations for each of the 3 modes of movement.

The chase behaviour of the ghosts varies, and the implementation of each of the parts are removed into a separate class. Hence, the algorithm to chase aggressively, ambush, patrol and random is placed in separate classes. An interface (ChaseBehaviour) is created to allow for the composition between the Ghost class and the different implementations of the Chase behaviour.

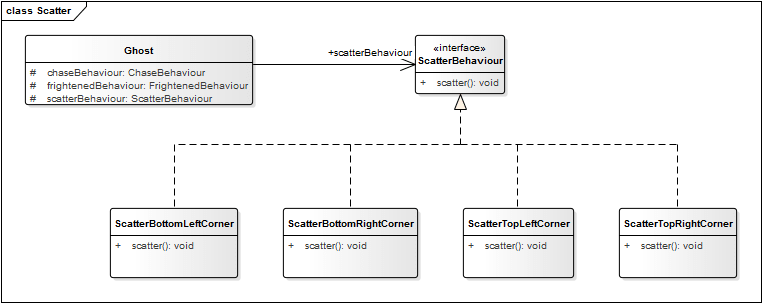
The following diagram illustrates the composition and implementation of the Ghost class and the different implementations of the ChaseBahaviour:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--W5ns_qpi--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/cdn-images-1.medium.com/proxy/0%2AgxvDPPeGZny9DzhU.png)

* [Ghost](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/Ghost.java) — The Ghost class contains the different behaviour that the different ghosts have in the Pac-Man game. There are three distinct modes a ghost can be in: chase, scatter and frightened.
* [ChaseBehaviour](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ChaseBehaviour.java) — The ChaseBehaviour interface is used to define different ghostly behaviours during the chase mode of the Pac-Man game. In chase mode, the ghosts will have different behaviours associated with their personalities.
* [ChaseAggressive](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ChaseAggressive.java) — The ChaseAggressive class contains the behaviour of a ghost in the Pac-Man game. In chase mode, the ghost chases aggressively and will usually take the shortest route to you, and tends to follow.
* [ChaseAmbush](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ChaseAmbush.java) — The ChaseAmbush class contains the behaviour of a ghost in the Pac-Man game. In chase mode, the ghost will attempt to ambush Pac-Man. The ghost tends to take a more wounding way towards Pac-Man with deadly effect.
* [ChasePatrol](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ChasePatrol.java) — The ChasePatrol class contains the behaviour of a ghost in the Pac-Man game. In chase mode, the ghost patrols around his designated block by default, only chasing Pac-Man if he comes near enough.
* [ChaseRandom](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ChaseRandom.java) — The ChaseRandom class contains the behaviour of a ghost in the Pac-Man game. In chase mode, the ghost will move in a random manner around the board and is not much of a threat.

The scatter behaviour of the ghosts varies, and the implementation of each of the parts are removed into a separate class. Hence, the algorithm to scatter to the top-left, top-right, bottom-left and bottom right corners is placed in separate classes. An interface (ScatterBehaviour) is created to allow for the composition between the Ghost class and the different implementations of the Scatter behaviour.

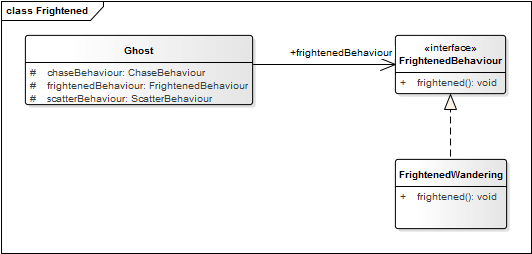
The following diagram illustrates the composition and implementation of the Ghost class and the different implementations of the ScatterBahaviour:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--leZWakkC--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/cdn-images-1.medium.com/proxy/0%2A4_IhhGxPSU_MsVBk.png)

* [Ghost](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/Ghost.java) — The Ghost class contains the different behaviour that the different ghosts have in the Pac-Man game. There are three distinct modes a ghost can be in: chase, scatter and frightened.
* [ScatterBehaviour](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ScatterBehaviour.java) — 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.
* [ScatterTopLeftCorner](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ScatterTopLeftCorner.java) — The ScatterTopLeftCorner class contains the behaviour of a ghost in the Pac-Man game. In scatter mode, the ghost will give up the chase and head for the top left corner of the board using its regular path-finding methods.
* [ScatterTopRightCorner](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ScatterTopRightCorner.java) — The ScatterTopRightCorner class contains the behaviour of a ghost in the Pac-Man game. In scatter mode, the ghost will give up the chase and head for the top right corner of the board using its regular path-finding methods.
* [ScatterBottomLeftCorner](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ScatterBottomLeftCorner.java) — The ScatterBottomLeftCorner class contains the behaviour of a ghost in the Pac-Man game. In scatter mode, the ghost will give up the chase and head for the bottom left corner of the board using its regular path-finding methods.
* [ScatterBottomRightCorner](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/ScatterBottomRightCorner.java) — The ScatterBottomRightCorner class contains the behaviour of a ghost in the Pac-Man game. In scatter mode, the ghost will give up the chase and head for the bottom right corner of the board using its regular path-finding methods.

The frightened behaviour of the ghosts may does not vary, however the implementation of is still removed into a separate class. Hence, the algorithm to wander around is placed in separate class. An interface (FrightenedBehaviour) is created to allow for the composition between the Ghost class and the implementation of the Wandering behaviour.

The following diagram illustrates the composition and implementation of the Ghost class and the different implementations of the FrightenedBehaviour:

[](https://res.cloudinary.com/practicaldev/image/fetch/s--d_FeKLNV--/c_limit%2Cf_auto%2Cfl_progressive%2Cq_auto%2Cw_880/https:/cdn-images-1.medium.com/proxy/0%2A1dg8YMVXIp6hLu9S.png)

* [Ghost](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/Ghost.java) — The Ghost class contains the different behaviour that the different ghosts have in the Pac-Man game. There are three distinct modes a ghost can be in: chase, scatter and frightened.
* [FrightenedBehaviour](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/FrightenedBehaviour.java) — The FrightenedBehaviour interface is used to define different ghostly behaviours during the frightened mode of the Pac-Man game. In frightened mode, the ghosts will all turn dark blue.
* [FrightenedWandering](https://github.com/Code2Bits/Design-Patterns-Java/blob/master/Behavioral%20Patterns/Strategy/Source/src/main/java/com/code2bits/designpatterns/behavioral/strategy/FrightenedWandering.java) — The FrightenedWandering class contains the behaviour of a ghost in the Pac-Man game. In frightened mode, the ghosts will all turn dark blue and aimlessly wander around in the maze for a few seconds.

The Strategy Pattern assists with the design of the different behaviours of the ghosts in Pac-Man. Applying the Strategy Pattern to the ghost movement design, the solution become reusable, extensible, maintainable and allows for change without major impact to the rest of the code.

I hope this article rekindled your memories of the good days of playing Pac-Man and also implementing Reusable Object-Oriented software, also known as Design Patterns.