# PACCAR

**Paccar**

|  |  |  |
| --- | --- | --- |
| RESUMEN En este documento se define el QUÉ y el CÓMO se hizo en lo referente al desarrollo del juego del pacman, se explican los métodos principales de cada clase implementada y el funcionamiento del juego.  También define los patrones de diseño utilizados. PALABRAS CLAVES: pacman, paccar, fantasmas, hilos, llama, clase, movimiento.ABSTRACT *This document defines WHAT and HOW we did the game named Pacman, it is explained the main methods of each implemented class and the functionality of the game.*  *It is also defined the design patterns used.*  KEYWORDS: *pacman, paccar, ghosts, threads, flame, class, movement.* |  | **Carolina Jiménez Gómez**  **1.112.786.793**  Estudiante Ingeniería de Sistemas y Computación.  Universidad Tecnológica de Pereira  krito\_j\_g@utp.edu.co  **Karen Lopez Segura**  **1.088.338.015**  Estudiante Ingeniería de Sistemas y Computación.  Universidad Tecnológica de Pereira  **stefanny@utp.edu.co** |

### 1. INTRODUCCIÓN

Se pretende explicar el funcionamiento del juego desarrollado en el lenguaje de programación Java, definir cada una de sus clases con sus métodos principales, y una breve explicación de la estructura manejada para la implementación de este.

**2. METODOLOGÍA**

Metodología simple, definición de cada uno de los elementos del programa y su estructuración.

**3. DESARROLLO**

El juego desarrollado fue un Pacman, con algunas modificaciones para darle nuestra marca personal.

El juego consiste en que hay un carro que puede disparar llamas de fuego, para matar a los fantasmas; si un fantasma se come la fruta que está en cierta parte del laberinto, el pacman se baja del carro, quedando vulnerable a que los fantasmas lo ataquen y lo maten, y sin poder disparar.

Utilizamos la biblioteca Swing de java, la cual nos permite un desarrollo de interfaces gráficas con independencia de la plataforma, también nos permite el manejo de hilos, muy importante a la hora de mover a los personajes.

Los patrones de diseño que utilizamos para la realización de este juego fueron los siguientes:

* **Prototype (Creacional):**

Tiene como finalidad crear nuevos objetos a partir de un modelo, duplicándolos, clonando una instancia creada previamente. Clonar objetos es más rápido que crearlos y setear luego cada valor.

Este prototipo se utiliza cuando se tengan objetos con datos repetitivos.

El cliente solicita la creación de un nuevo objeto a partir de los prototipos.

Elprototipo concretoposee características concretas que serán reproducidas para nuevos objetos e implementa una operación para clonarse.

El prototipo declara una interfaz para clonarse, a la que accede el cliente.

Este patrón lo utilizamos creando una clase abstracta llamada “movimiento”, de la cual algunos objetos de nuestro juego van a heredar de él, y al hacerlo, implementarán todos sus métodos.

Este patrón lo utilizamos para no copiar y pegar el mismo código, puesto que los fantasmas tienen todos métodos muy similares, difiriendo únicamente en sus movimientos.

* **Observer (Comportamental):**

Define una dependencia del tipo *uno-a-muchos* entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes.

Cada uno de los fantasmas deben observar al pacman, puesto que de esto depende sus movimientos.

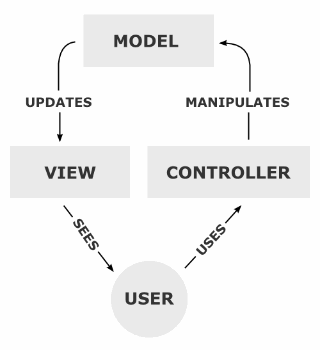
* **State (Comportamental):**

Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Busca que un objeto pueda reaccionar según su estado interno.

Este patrón se utiliza cuando un determinado objeto tiene diferentes estados y también distintas responsabilidades según el estado en que se encuentre en determinado instante. También puede utilizarse para simplificar casos en los que se tiene un complicado y extenso código de decisión que depende del estado del objeto.

Este patrón es utilizado ya que el PacCar tiene dos estados diferentes, el primero es estadoNormal, donde se ve la imagen del carro y puede disparar llamas, y el otro estado es el estadoVulnerable en el que el que aparece la imagen del Pacman, y es allí donde los fantasmas lo pueden matar.

* **Modelo vista-controlador (Creacional):**



El modelo envía a la 'vista' aquella parte de la información que en cada momento se le solicita para que sea mostrada. Las peticiones de acceso o manipulación de información llegan al modelo a través del controlador.

El controlador responde a eventos e invoca peticiones al modelo cuando se hace alguna solicitud sobre la información.

La vista presenta el modelo en un formato adecuado para interactuar (usualmente la [interfaz de usuario](https://es.wikipedia.org/wiki/Interfaz_de_usuario)) por tanto requiere de dicho modelo la información que debe representar como salida.

Este patrón se utilizó en el desarrollo del juego de Pacman para manejar la interacción del usuario con la interfaz del juego, cuando éste oprima un botón, por ejemplo, el controlador recibe la instrucción y actualiza el modelo, y después esta actualización pasa a la vista para que lo muestre como salida, todo esto para asegurar una interacción exitosa entre la parte gráfica y la parte lógica del juego.

A continuación se explicará cada una de las clases creadas, su funcionalidad, y cómo son internamente.

Todas las imágenes del juego son del tipo Imagen, la cual extiende de Jpanel, que es un contenedor, por ende se tienen todas las imágenes estructuradas y simplificadas en estos.

La clase Image trabaja con imágenes, que las trabaja como íconos. Tiene métodos como getIcon para devolver el ícono de la imagen que se ha establecido, setIcon para setear un ícono sobre ese objeto tipo imagen, y el más importante paintComponent() que se encarga de pintar el componente en el Jpanel cuando es necesario.

Cabe destacar que cada imagen que trabajamos tiene una dimensión no mayor a 40X40 pixeles.

Tenemos 3 clases principales que están encargadas de diferentes aspectos dentro del programa, la primera es la clase Vista que se encarga del manejo de imágenes y posicionarla dentro del Jframe del juego; esta clase se puede ver como la que interactúa con el usuario.

Por otro lado tenemos la clase Modelo, que trabaja sobre la clase Tablero, ambas, se encargan del manejo de la matriz la cual tiene toda la información de los movimientos y del juego en si, es decir, es el manejo de la parte lógica del juego.

Por último tenemos la clase principal Control, que se encarga de hacer una comunicación entre la vista y el modelo, es decir, se implementó el patrón de diseño modelo-vista-controlador. Es una forma de mantener el código estructurado y organizado.

En la clase Vista se instancian todos los objetos tipo imagen: utilizamos objetos tipo imagen para el pacman, los fantasmas, la llama, la cereza y para los letreros de “ganaste” o “perdiste”.

En el constructor de la clase se cargan todas las imágenes en el Jframe.

Se tienen diferentes métodos que retornan y setean todos los atributos del objeto.

Tenemos un método setImagenVulnerable() y setImagenNormal(), moveLeft(), moveRight(), moveDown(), moveUp() que se encargan de actualizar la imagen del pacman dependiendo de su estado actual y de la tecla que se encuentre activa.

El método Terminar() se encarga de dejar de mostrar todos los elementos del juego (personajes).

El método eliminar() se encarga de eliminar cualquier imagen puesta en la posición X,Y (la cual se recibe como parámetro) y también la borra de la matriz del tablero, por ende, todos quedan con la misma información.

Dentro de esta clase se encuentra contenida otra clase llamada “MyKeyListener” la cual utilizamos para darnos cuenta si el usuario tecleó algo, y si es una de las teclas que se utiliza en el juego, guarda esa tecla actual. El KeyListener queda de alguna forma atento a alguna excitación del teclado que se haga sobre la clase vista, que es nuestro Jframe, y cuando ocurre llama a los métodos correspondientes.

La clase Tablero solo es accedida por la clase Modelo. En esta clase se carga la información del laberinto a una matriz llamada “matriz” desde un archivo, esta es la matriz sobre la que trabajaremos todo el tiempo, es la representación del estado del juego.

Dentro de esta clase se tienen métodos que modifican y retornan los elementos de la matriz.

La clase Control se encarga de inicializar todos los objetos del juego, y de crear el Modelo y la Vista. También se tiene un método Terminar() que se encarga de cortar todos los hilos de ejecución.

La clase “movimientoPacman” se encuentra definido todos los atributos del personaje principal de este juego, y los métodos que permiten un acceso a ellos. El movimiento del carro es constante, ya que utiliza el método run(), que se implementa por la interface Runnable, realizando este proceso independientemente de los otros.

El movimiento del carro está definido de esta manera: el carro comienza moviéndose hacia la derecha, y sigue este camino mientras no se haya presionado ninguna otra tecla, a menos que sea la tecla de la barra espaciadora, donde no se actualiza esta, pues dejaríamos de movernos. Cada que cambia de posición, se llama a la clase “Vista” y se cambia la posición de la imagen donde se encontraba el carro, también se llama a la clase “Modelo” para modificar la matriz sobre la cual estamos trabajando.

Nótese que no tenemos el problema de borrar la imagen del tablero en la posición anterior, puesto que una imagen en realidad es un Jpanel. Cuando se ha movido el carro, se cambia en la matriz del tablero la nueva posición con un “9”, el cual es nuestra representación del carro, y donde no queda nada se pone un “0”.

Cada uno de las clases siguientes heredan de la clase abstracta movimiento: Clyde, Blynky, Inky, Pinky, Llama. Donde los primeros cuatro son los fantasmas del juego, cada uno con una velocidad de movimiento diferente, una imagen asociada diferente y un movimiento diferente.

La clase Llama también tiene su propia velocidad, imagen particular y movimiento particular. La llama tiene un alcance de 3 recuadros. A medida que se va disparando, la llama va avanzando, si es un fantasma, llama a la función de la clase Vista “eliminar” la cual se encarga de eliminar el elemento en la posición enviada, y luego lo eliminamos de la matriz del laberinto por medio de la clase Modelo.

Siempre se está llamando al movimiento de la llama, puesto que esto es lo que está definido en el método run(), y este se ejecuta cuando inicializamos el objeto en la clase principal.

Todos los fantasmas implementan la interfaz Observer, puesto que están observando constantemente al pacman para saber si ha cambiado su estado o no, y para moverse de cierta forma.

Para el movimiento de los fantasmas se crearon dos funciones, en ellas sólo se analiza un posible movimiento; Maximiza() retorna la posición que haga que se aleje del PacCar, está basada en la fórmula de distancia entre dos puntos. El método Minimiza() es muy similar, solo que busca el movimiento que haga que se hacerque más al Pacman.

Cada uno de estos métodos se utilizan dependiendo del estado en el que se encuentre el Pacman, ya sea vulnerable o estado normal.

Los fantasmas se mueven de forma diferente dependiendo del estado del Pacman.

Cada que un fantasma va a realizar un movimiento, se pregunta a la matriz (desde Modelo, claro está) si la posición donde se va a mover existe un elemento, y realiza los cambios necesarios sobre la vista (Vista) y sobre la matriz (Modelo). Siempre se hace un llamado a Control, que es el intermediario.

Se gana el juego cuando se han matado todos los fantasmas, o se pierde el mismo cuando se ha eliminado el pacman del tablero, deteniendose todos los hilos que se estaban ejecutando y mostrando una imagen en la pantalla.

**4. CONCLUSIONES**

El desarrollo de un juego como Pacman utilizando las funciones que nos proporciona java lo vuelven mucho más fácil de tratar y manejar; los patrones de diseño simplifican problemas de orden, de estructura y demás a la hora de programar. La programación orientada a objetos permite el manejo natural de un conjunto de funciones, acá llamados métodos, que están relacionados con cierta estructura, acá llamada objeto, es una forma más natural de programar y de acercarnos a este mundo informático.

**5. ANEXOS**

Se adjunta el programa, el ejecutable del programa, la presentación en power point de la exposición realizada y un README.txt donde se explica cómo se debe manejar el instalador del juego.

### 6. WEBGRAFÍA

* [http://migranitodejava.blogspot.com.co/2011/05/](http://migranitodejava.blogspot.com.co/2011/05/prototype.html)
* https://es.wikipedia.org/wiki/Patr%C3%B3n\_de\_dise%C3%B1o
* <http://www.nebrija.es/~abustind/Informatica/MetodologiaII/Polimorfismo.pdf>