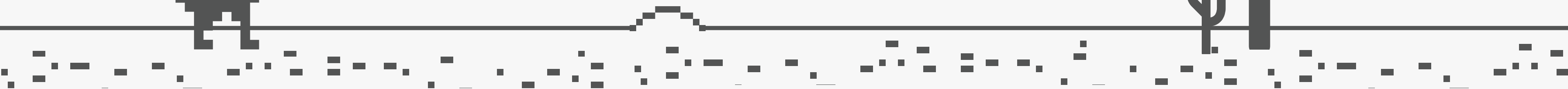




# STRATEGY



START



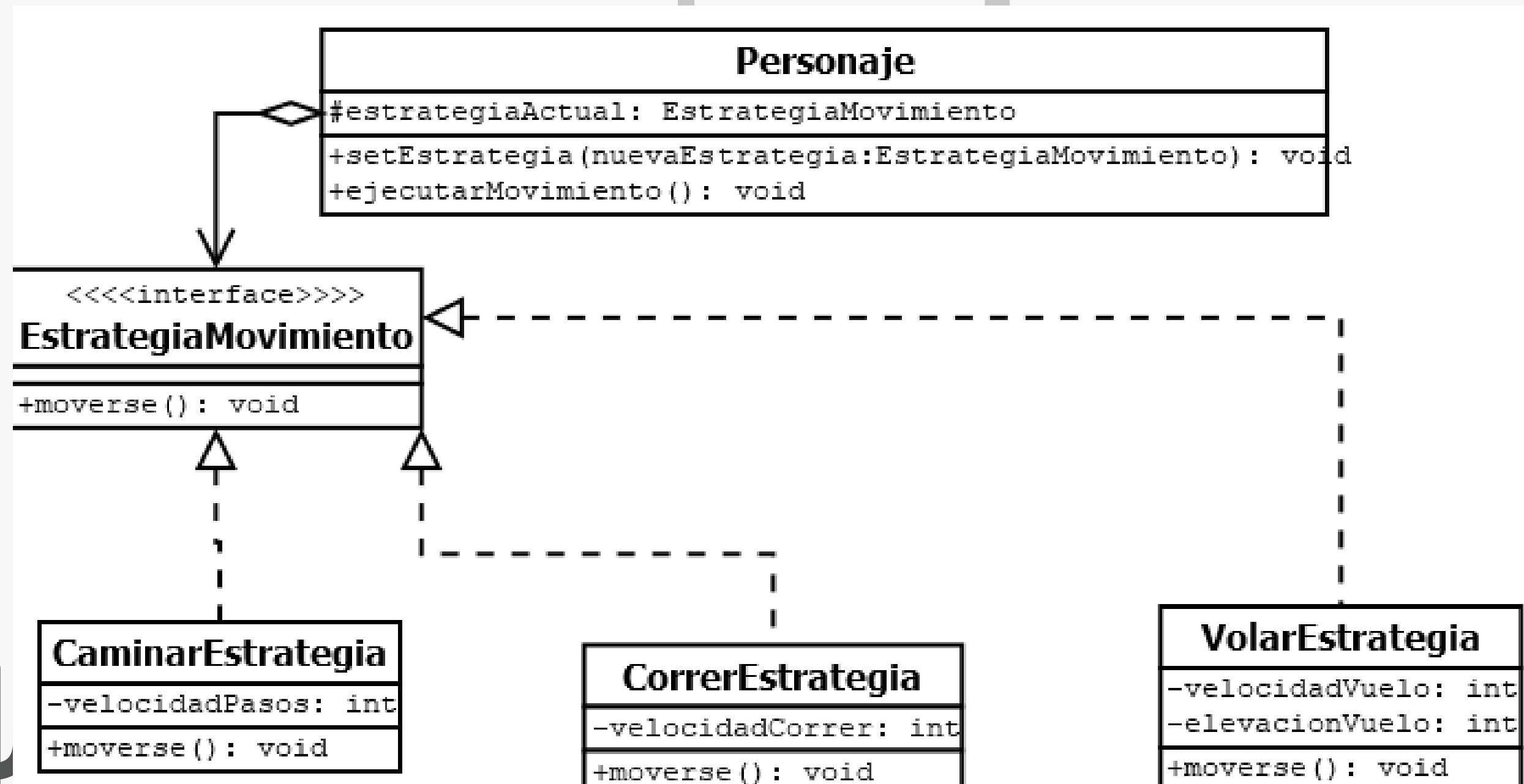
# ESCENARIO PROBLEMA

EN UN VIDEOJUEGO SE NECESITA QUE UN PERSONAJE PUEDA MOVERSE DE DISTINTAS FORMAS: CAMINAR, CORRER O VOLAR. AL IMPLEMENTAR TODOS LOS COMPORTAMIENTOS DENTRO DE LA CLASE PERSONAJE, EL CÓDIGO SE VUELVE DIFÍCIL DE MANTENER Y MODIFICAR CADA VEZ QUE SE AGREGA UN NUEVO TIPO DE MOVIMIENTO.

PARA SOLUCIONAR ESTO, SE APLICA EL PATRÓN STRATEGY, SEPARANDO CADA FORMA DE MOVIMIENTO EN ESTRATEGIAS INDEPENDIENTES E INTERCAMBIABLES. ASÍ, EL PERSONAJE PUEDE CAMBIAR SU MODO DE DESPLAZAMIENTO DINÁMICAMENTE SIN ALTERAR SU ESTRUCTURA INTERNA.



# STRATEGY APLICADO





CÓDIGO JAVA  
[HTTPS://GITHUB.COM/MADEHUN777/STRATEGY](https://github.com/MADEHUN777/STRATEGY)

# ¿CUÁNDO APLICAR ESTE PATRÓN?

CUANDO TIENES UNA CLASE CON MULTIPLES VARIANTES DE UN MISMO ALGORITMO.

CUANDO QUIERES EVITAR CONDICIONALES COMPLEJOS (IF-ELSE O SWITCH) PARA SELECCIONAR UN COMPORTAMIENTO.

CUANDO NECESITAS CAMBIAR ESOS ALGORITMOS EN TIEMPO DE EJECUCIÓN (COMO HICIMOS EN LA DEMO)

PARA CUMPLIR EL PRINCIPIO DE ABIERTO/CERRADO (ABIERTO A EXTENSION, CERRADO A MODIFICACION)





# USOS CONOCIDOS DE STRATEGY

## SISTEMAS DE COMPRESIÓN O ENCRIPCIÓN

PERMITE CAMBIAR DINÁMICAMENTE EL ALGORITMO DE COMPRESIÓN (ZIP, RAR, 7Z) O DE ENCRIPCIÓN (AES, RSA).

UN EJEMPLO ES LA BIBLIOTECA JAVA.UTIL.ZIP EN JAVA USA CLASES INTERCAMBIABLES PARA COMPRESIÓN DE DATOS.

## ORDENAMIENTO DE COLECCIONES

EL PATRÓN STRATEGY SE USA PARA PERMITIR DISTINTOS ALGORITMOS DE ORDENAMIENTO, SIN CAMBIAR LA ESTRUCTURA DE LA COLECCIÓN. EN JAVA, EL MÉTODO COLLECTIONS.SORT() PERMITE PASAR DIFERENTES COMPARADORES (COMPARATOR), QUE ACTÚAN COMO ESTRATEGIAS DE ORDENAMIENTO.

## MÉTODOS DE PAGO EN APLICACIONES DE COMERCIO ELECTRÓNICO

PERMITE CAMBIAR LA FORMA DE PROCESAR UN PAGO (TARJETA, PAYPAL, CRIPTOMONEDAS) SIN MODIFICAR LA LÓGICA DEL SISTEMA PRINCIPAL, POR EJEMPLO PLATAFORMAS COMO SHOPIFY O MERCADOPAGO IMPLEMENTAN DISTINTOS PAYMENT STRATEGIES.

## INTELIGENCIA ARTIFICIAL EN VIDEOJUEGOS

SE USA PARA DEFINIR DISTINTOS COMPORTAMIENTOS DE ENEMIGOS O PERSONAJES (AGRESIVO, DEFENSIVO, EVASIVO). UN EJEMPLO SON MOTORES COMO UNITY Y UNREAL ENGINE APLICAN ESTRATEGIAS DE MOVIMIENTO O DECISIÓN.



# REFERENCIAS



- GAMMA, E., HELM, R., JOHNSON, R., & VLISSIDES, J. (1995). DESIGN PATTERNS: ELEMENTS OF REUSABLE OBJECT-ORIENTED SOFTWARE (CAPITULO "STRATEGY").[HTTPS://WWW.JAVIER8A.COM/ITC/BD1/ARTICULO.PDF](https://www.javier8a.com/ITC/BD1/ARTICULO.PDF)
- GAMMA ET AL. (1995), GOF – STRATEGY PATTERN EXAMPLE (SORTING ALGORITHMS).  
[HTTPS://REFACTORING.GURU/DESIGN-PATTERNS/STRATEGY](https://refactoring.guru/design-patterns/strategy)
- FREEMAN, E., & FREEMAN, E. (2004). HEAD FIRST DESIGN PATTERNS  
[.HTTPS://WWW.GOODREADS.COM/BOOK/SHOW/58128.HEAD\\_FIRST\\_DESIGN\\_PATTERNS](https://www.goodreads.com/book/show/58128.Head_First_Design_Patterns)
- MILLINGTON, I., & FUNGE, J. (2009). ARTIFICIAL INTELLIGENCE FOR GAMES.  
[HTTPS://WWW.ROUTLEDGE.COM/AI-FOR-GAMES-THIRD-EDITION/MILLINGTON/P/BOOK/9780367670566](https://www.routledge.com/AI-for-Games-Third-Edition/Millington/P/book/9780367670566)

