Principios SOLID

1. S: Single Responsibility

Este principio se refiere a que una clase debería tener una y solo una, razón para cambiar. Como podemos observar a lo largo del proyecto este principio si se cumple, y podemos verlo en las diferentes clases implementadas (card, player, etc).

2. O: Open/Closed

Este principio se refiere a que se debería de poder extender el comportamiento de una clase, sin modificarla. De igual forma nuestro proyecto si cumple, ya que al estar tan generalizado el código, logramos hacer que una clase pueda extenderse con facilidad, y no necesariamente tener que modificarse.

3. L: Liskov Substitution

Este principio dice que las clases derivadas deben poder sustituirse por sus clases bases, el proyecto cumple con lo mencionado, esto lo podemos observar por ejemplo en las clases de Files y FIlesPanjpar, que podriamos sutituir la clase y el programa seguirá funcionando de forma perfecta, y así lo podemos ver en las diferentes clases derivadas.

4. I: Interface Segregation

Este principio dice que se deben de hacer interfaces que sean específicas para un cliente, o sea para una finalidad en concreto. Consideramos que este principio se cumple, ya que la interfaz creada apunta todo a un mismo objetivo y si bien es cierto que contiene muchos métodos, todos están funcionando y no hay ninguno que esté implementado en vano, por lo tanto el principio se cumple.

5. D: Dependency Inversion

Se refiere a que el código debería depender de abstracciones, y no de clases concretas. Si bien es cierto que en la mayoría del código dicho principio se cumple, no en la totalidad del código, por lo tanto en algunos puntos si se utilizan abstracciones, pero no en todas, por lo tanto el principio se cumple pero no al 100%.