1. Czym różni się wzorzec Strategia od zwykłej implementacji interfejsu? Jakie są wady i zalety tego wzorca?

Strategia to wzorzec projektowy pozwalający zdefiniować rodzinę algorytmów, umieścić je w osobnych klasach i uczynić obiekty tych klas wymienialnymi.

Główne różnice między strategią, a zwykłą implementacją interfejsu:

- Wzorzec Strategia jest używany do zmiany algorytmów lub zachowań w trakcie działania programu. Zwykła implementacja interfejsu natomiast jest po prostu konkretną realizacją danej funkcjonalności. Nie zmienia się ona w trakcie działania programu.
- Wzorzec Strategia pozwala rozproszyć odpowiedzialność za różne algorytmy pomiędzy różne klasy. Każda strategia jest reprezentowana przez oddzielną klasę, co ułatwia zarządzanie kodem. Zwykła implementacja interfejsu zazwyczaj nie rozbija funkcjonalności na różne klasy i nie rozprasza odpowiedzialności w taki sposób.
- Dzięki wzorcowi Strategia zmiana algorytmu jest stosunkowo prosta. Wystarczy stworzyć nową klasę implementującą interfejs strategii. W przypadku zwykłej implementacji interfejsu, zmiana algorytmu mogłaby wymagać modyfikacji istniejącej klasy, co może być bardziej kłopotliwe i ryzykowne.
- Wzorzec Strategia ułatwia testowanie, ponieważ poszczególne strategie mogą być testowane niezależnie. Ponadto, dodawanie nowych strategii nie wpływa na już istniejący kod, co czyni go bardziej elastycznym i łatwiejszym do rozszerzania. Zwykła implementacja interfejsu może wymagać zmian w istniejącym kodzie, co może prowadzić do wprowadzenia błędów i problemów w testowaniu.

Powyżej w różnicach wypisano zalety tego wzorca takie jak rozproszenie odpowiedzialności, łatwość zmiany algorytmów, ułatwienie testowania.

Wady zastosowania wzorca strategia to:

- Zwiększona złożoność: Implementacja wzorca Strategia może prowadzić do zwiększenia liczby klas w programie, co może być uciążliwe w przypadku prostych problemów lub małych projektów.
- Potrzeba dodatkowej abstrakcji: Wzorzec Strategia wymaga dodatkowej abstrakcji w postaci interfejsu strategii oraz oddzielnych klas implementujących różne strategie, co może być zbędne w przypadku prostych zastosowań.
- Wymaga zrozumienia kontekstu: Wzorzec Strategia wymaga od programisty zrozumienia kontekstu, w którym zostanie zastosowany, aby odpowiednio zaprojektować interfejsy i strategie.
- Nadmiarowa elastyczność: W niektórych przypadkach zbyt duża elastyczność wzorca Strategia może prowadzić do nadmiernego skomplikowania kodu, gdy nie ma potrzeby zmiany algorytmów w trakcie działania programu.