|  |  |
| --- | --- |
| STRATEGY PATTERN | * Używanie kompozycji a nie dziedziczenia * Wprowadza się abstrakcyjne "strategie", które potem są implementowane w konkretne wersje algorytmów.   Przykład:  Mamy kaczki z których każda musi kwakać. Zamiast tworzyć metodę quack() w superklasie Duck z której dziedziczą wszystkie inne kaczki, tworzymy np. Interfejs albo Subklasę IQuackStrategy. W interfejsie IQuackStrategy jest abstrakcyjna metoda quack() i potem implementacje np: SimpleQuackStrategy i LoudQuackStrategy z których każda będzie miała własną implementację quack();  <IQuackStrategy>  {quack()}  \*  |  /\  / \  SimpleQuackStrategy LoudQuackStrategy  { @Override quack() } { @Override quack() }   * Główna klasa Duck powinna wtedy posiadać pole IQuackStrategy w które w konstruktorze wstrzykniemy odpowiednią implementację tego interfejsu. * Dzięki temu możemy np. w trakcie trwania programu podmienić implementację danej abstracji w naszym obiekcie. |
| FACTORYPATTERN | * Wzorzec do tworzenia klasa posiadającej metodę do automatycznego tworzenia instancji obiektów. |

===