Wzorzec projektowy Pyłek to wzorzec strukturalny, który optymalizuje wykorzystanie pamięci, umożliwiając wielu obiektom współdzielenie wspólnego stanu zamiast przechowywać niepotrzebne nadmiarowe dane dla każdego obiektu z osobna.

Głównym celem wzorca Flyweight jest zmniejszenie zużycia pamięci RAM przez aplikację poprzez umożliwienie podobnym obiektom współdzielenia jak największej ilości informacji. Działa to na zasadzie takiej, że, wydzielamy sobie stan wewnętrzny i zewnętrzny z obiektu. Stan wewnętrzny to dane, których obiekt potrzebuje do samodzielnego działania, a stan zewnętrzny to dane kontekstowe, które mogą udostępniać powiązane obiekty. Dzieląc stan zewnętrzny i oddzielając stan wewnętrzny, pyłek może zmniejszyć zużycie pamięci przez aplikację bez zmiany sposobu jej działania.

Aby zaimplementować wzorzec pyłek potrzebujemy kilku elementów:

- Współdzielony pyłek - deklaruje metody, przez które konkretne pyłki mogą otrzymać stan zewnętrzny i wykorzystać go do wykonania odpowiedniej logiki.

ważne – musimy zapewnić aby stan był niemutowalny, że stan obiektu pyłku nie może zostać zmieniony, ponieważ może być używany w różnych kontekstach. Stan pyłku powinien być inicjowany tylko raz za pomocą parametrów konstruktora. Inne obiekty nie powinny mieć dostępu do żadnych seterów

- Konkretny pyłek - przechowuje stan wewnętrzny obiektu, niezależny od kontekstu działania obiektów, może implementować interfejs czy tam klase współdzielonego pyłku, ale nie musi, różne są implementacje tego wzorca

- Fabryka Pyłków - odpowiedzialna za tworzenie i zarządzanie pyłkami; dba o to, że pyłki są odpowiednio współużytkowane.

- Klient - inicjuje działanie logiki aplikacji z wykorzystaniem wzorca, przechowuje referencje do pyłków, oblicza i przetwarza zewnętrzny stan pyłków.

najczęstsze zastosowanie tego wzorca jest w grach, ponieważ gry przechowują w pamięci często bardzo dużą ilość obiektów, i w grach zazwyczaj zapisywany jest tylko snapshot z danej rozgrywki, stan gry w danym momencie, który zazwyczaj jest implementowany też przy pomocy wzorca ale memento, czyli pamiątka,  
  
natomiast w przypadku np. aplikacji webowych, bardzo rzadko można znaleźć zastosowanie do tego typu wzorca, ponieważ w takich crud'ach aktualny stan przechowują bazy danych a nie odbywa się to w pamięci serwera  
  
co do samego kodu który stworzyłem, najpierw do głowy mi wpadł jakich edytor tekstu, na wzór worda, ponieważ każdy znak ma dany rozmiar, kolor oraz znak, ale porzuciłem temat, bo musiałbym to bardziej rozbudować,

Przykład użycia wzorca pyłek w aplikacji webowej:

- Załóżmy, że chcemy stworzyć prostą aplikację do tworzenia i edytowania dokumentów tekstowych online.

- Możemy zdefiniować klasę Znak, która będzie reprezentować wspólny stan wewnętrzny dla wszystkich znaków, takich jak czcionka, rozmiar i kolor.

- Możemy również zdefiniować klasę Dokument, która będzie reprezentować specyficzny stan zewnętrzny dla każdego dokumentu, taki jak tytuł, autor i treść.

- Zamiast tworzyć nowy obiekt dla każdego znaku w dokumencie, możemy użyć wzorca pyłek i stworzyć fabrykę pyłków, która będzie zarządzać pulą obiektów Znak i zwracać istniejące obiekty o żądanych parametrach.

- W ten sposób możemy zaoszczędzić pamięć i poprawić wydajność aplikacji, ponieważ nie będziemy tworzyć wielu podobnych obiektów.