Wzorce programowania

Piotr Dobosz

Czym jest wzorzec

- "Wzorzec to sprawdzona koncepcja, która opisuje problem powtarzający się wielokrotnie w określonym kontekście, działające na niego siły, oraz podaje istotę jego rozwiązania w sposób abstrakcyjny"

Christopher Alexander

Wzorzec architektoniczny

- Tyczą się całego systemu informatycznego
- Wskazuje rozwiązanie określonego architekturę systemu informatycznego
- Wskazują funkcje i metody (oraz zasady), które realizowane są przez element

Ojcowie wzorców

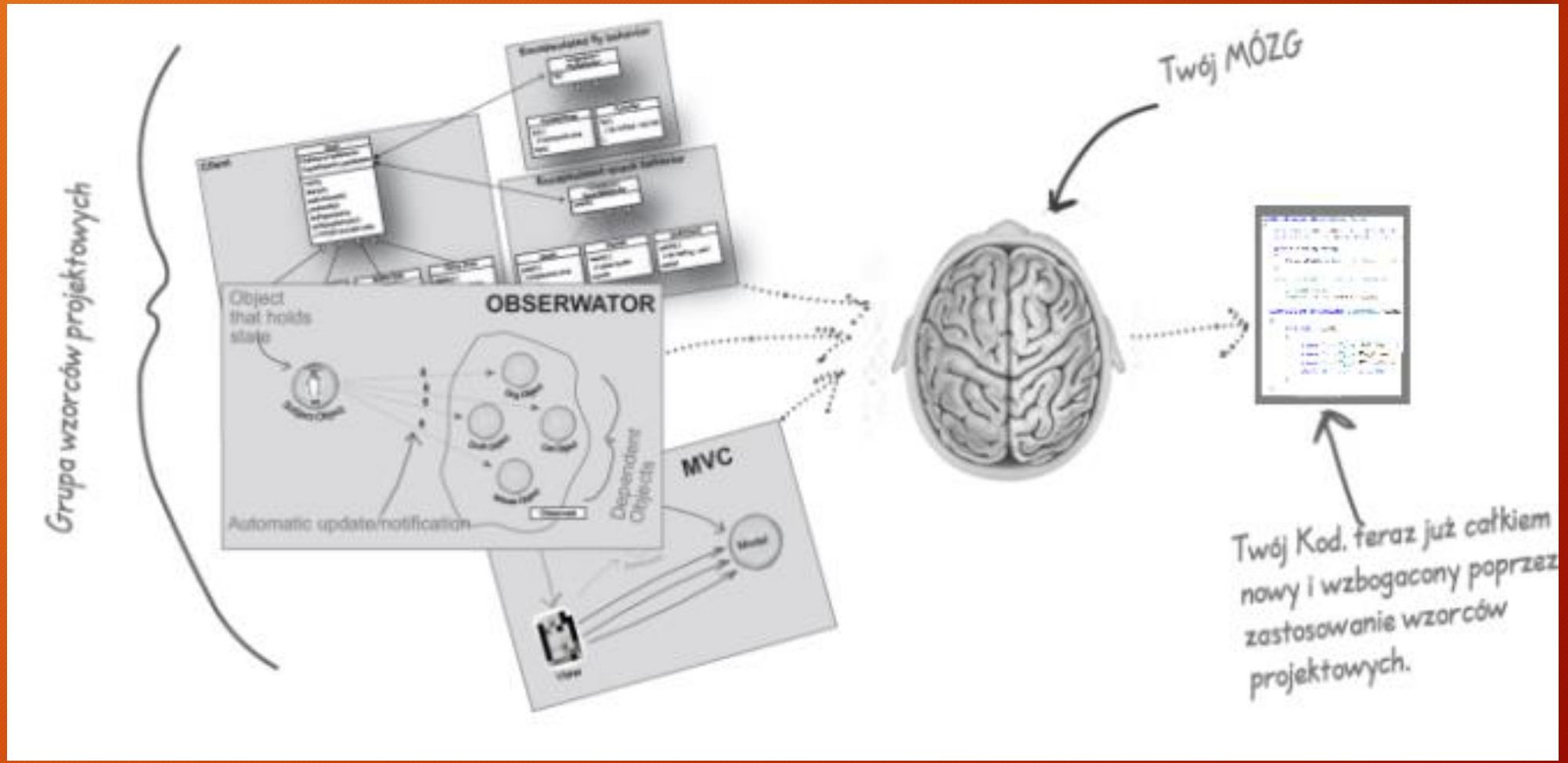
- Wzorzec projektowy identyfikuje i opisuje pewną abstrakcję, której poziom znajduje się powyżej poziomu abstrakcji pojedynczej klasy, instancji lub komponentu.

E. Gamma, R. Helm, R. Johnson, J. Vlissides (1994)

Wzorzec projektowy

- Sprawdzone rozwiązanie często pojawiających się problemów w programowaniu (w fazie projektowania)
- Najważniejszym aspektem jest utrzymywanie czytelności kodu dla późniejszego rozwoju/modyfikacji
- Ma zastosowanie jedynie do obiektowych języków programowania

Jak działają wzorce



Ogólny opis wzorca

- Nazwa
- Problem
- Rozwiązanie
- Konsekwencje

Wzorzec projektowy

Źródło: https://infotraining.bitbucket.io/cpp-dp/_images/atrybuty.png

Model, View, Controller

- Jeden z podstawowych wzorców projektowych
- Wprowadza podział w kodzie źródłowym na trzy sekcje, dzięki czemu kod ten staje się czytelniejszy i łatwiejszy w późniejszym rozwijaniu projektu
- Zaproponowany w latach 70 jako rozwiązanie jednego z problemów programistycznych w języku SmallTalk-80

Moduły MVC

- Model – do tej części trafiają obiekty i metody odpowiadające za całokształt określonego przedsięwzięcia; to po prostu obiekty i metody, które odpowiadają za wykonywanie instrukcji naszej aplikacji. Muszą być niezależne od zdarzeń generowanych przez program oraz postaci graficznej programu
- View – odpowiada za wygląd tworzonej aplikacji (pliki z grafiką)
- Controller – element odpowiadający za reakcję na zdarzenia wywoływane np. Przez użytkownika

Podział wzorców

- Kreacyjne – tworzenie obiektów i ich hermetyzacja od reszty kodu
- Strukturalne – tworzenie struktur obiektowych, łączonych ze sobą
- Zachowawcze – opis algorytmów i charakterystyka interkacji pomiędzy obiektami

Szablon wzorca projektowego

- strukturę - graficzną reprezentację klas składowych wzorca
- uczestników - nazwy i odpowiedzialności klas składowych wzorca
- współdziałania - opis współpracy między uczestnikami
- konsekwencje - efekty zastosowania wzorca
- implementację - opis implementacji wzorca w danym języku
- przykład - kod stosujący wzorzec
- pokrewne wzorce - wzorce używane w podobnym kontekście

Szablon wzorca projektowego

- nazwę - lakoniczny opis istoty wzorca
- klasyfikację - kategorię, do której wzorzec należy
- cel - do czego wzorzec służy
- aliasy - inne nazwy, pod którymi jest znany
- motywację - scenariusz opisujący problem i rozwiązanie
- zastosowania - sytuacje, w których wzorzec jest stosowany

Szablon wzorca projektowego

Sklep z elektroniką

- Stan magazynowy
- Zakupy klientów
- Baza dystrybutorów
- Kontakt z klientem

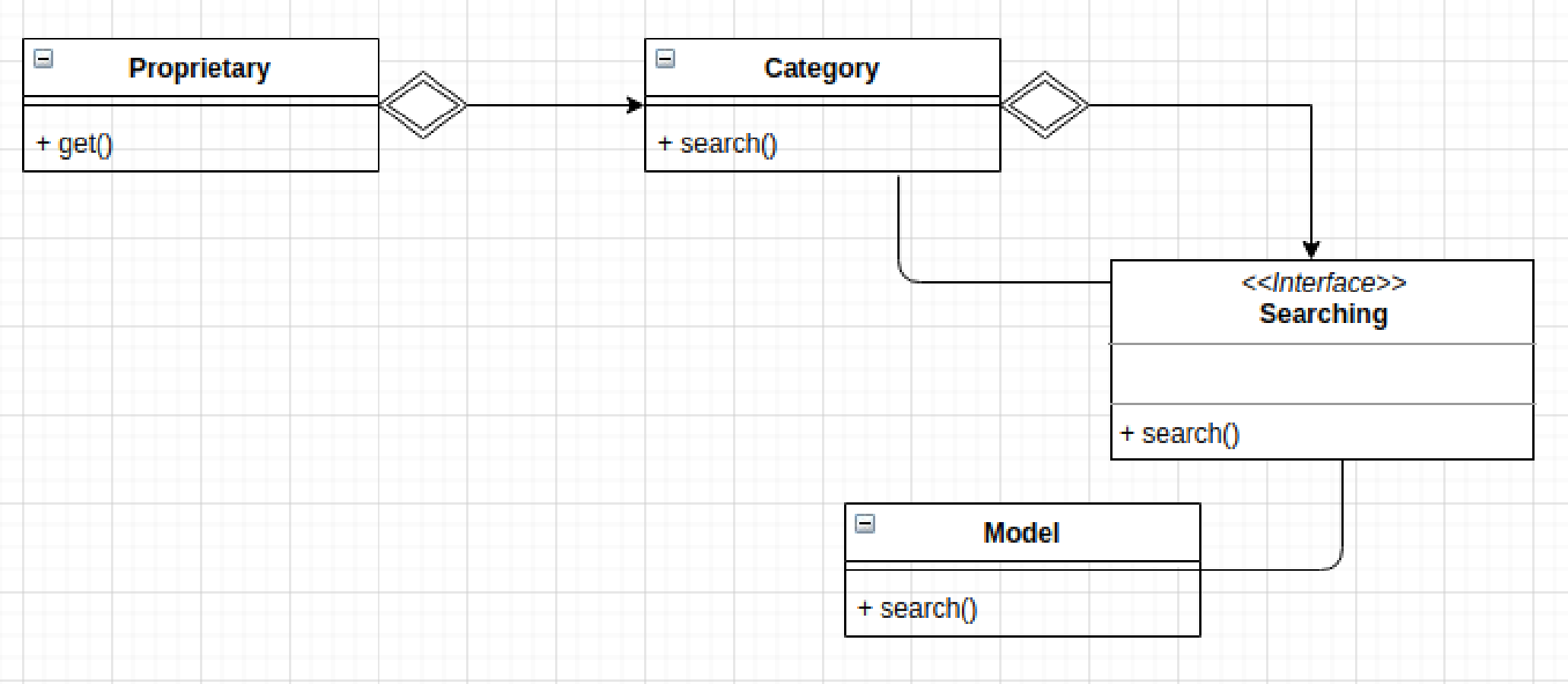
Magazyn

- Produkty mogą być przechowywane wedle: nazwy, producenta, funkcjonalności, przeznaczenia, podzespołów itp.
- Wymienione wyżej kryteria mogą być proste (nazwa urządzenia -> wyszukiwanie np.. Alfabetyczne) bądź złożone (funkcjonalność, podzespoły); te ostanie moża określić jako kategorię własności
- Kategorie własnościowe to najczęściej struktura drzewiasta złożona z wymienionych wcześniej kategorii

Kategoria własnościowa - podzespoły

1. Procesor
1.1 Wydajność operacyjna
1.1.1 Częstotliwość
1.1.2 pamięć cache
1.1.3 rozszerzenia i dodatkowe technologie
1.2 Architektura
1.2.1 Ilość rdzeni
1.2.2 Ilość wątków
1.2.3 Rodzina procesorów

Schemat przeszukiwania magazynu



Wzorzec kompozytowy

- Category to kompozyt (obiekt tworzony z innych komponentów)
- Interfejs Searching jest komponentem
- Model to liść (element końcowy)

Wzorzec kompozytowy

- Kompozyt to węzeł z referencjami do potomków; deleguje polecenia od potomków do pozostałych węzłów
- Komponent deklaruje interfejs dla obiektów struktury
- List to element terminalny

Wzorzec kompozytowy - konsekwencje

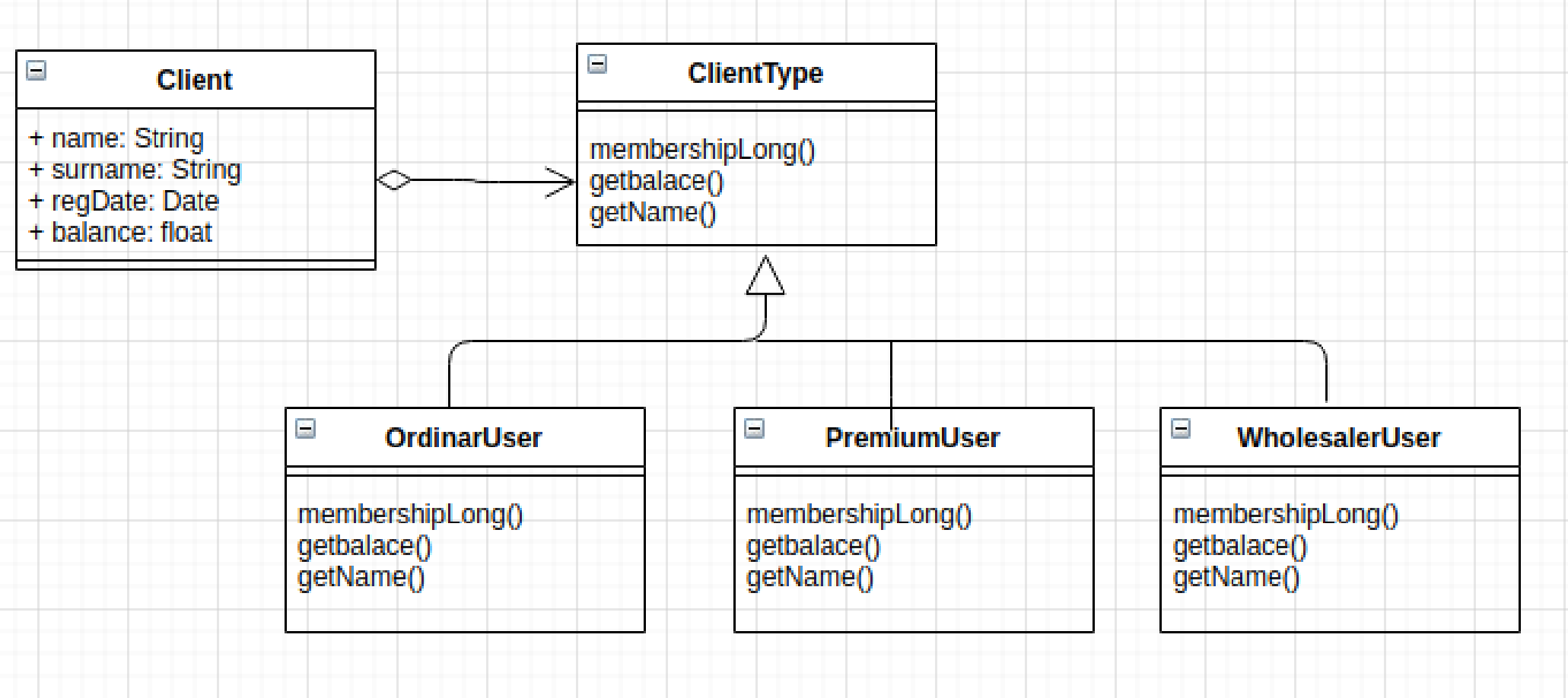
- Pozwala na definiowanie rozgałęzionych struktur, dodawanie ich w ramach kompozytu i pozwala na zarządzanie dużą ilością elementów tejże struktury

Zakupy klientów

- Typ klienta:

1) Zwykły
2) Hurtownik
3) Premium (stały)

Wzorzec stanu



Wzorzec stanu

- Client to kontekst; jest w stanie relacji docelowego obiektu
- ClientType to typ abstrakcyjny; hermetyzuje zachowanie z każdym stanem obiektu docelowego
- OrdinarUser, PremiumUser, WholesalerUser to stan konkretny; może posiadać własne metody dla własnego kontekstu

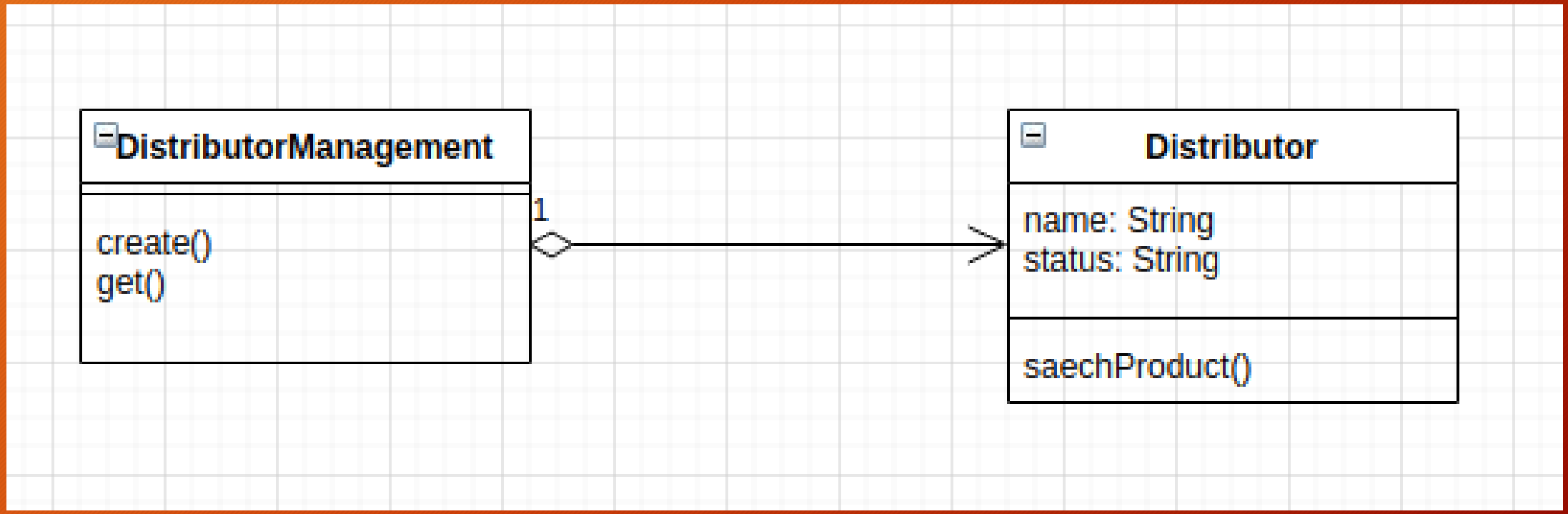
Wzorzec stanu - konsekwencje

- Każdy stan posiada własny obiekt, przez co jego stan jest związany konkretnie z nim
- Zmiana stanu jest realizowana poprzez zmianę stanu obiektu na inny
- Eliminuje niespójność
- Obiekty stanu definują tylko zachowanie, same zaś są bezstanowe

Baza dystrybutorów

- Liczba dystrybutorów może być teoretycznie dowolna
- Każdy z dystrybutorów posiada indywidualne warunki współpracy
- Dystrybutor może dostarczać do naszego magazynu różne produkty, z różnych działów

Wzorzec puli obiektów



Wzorzec puli obiektów

- DistributorManagement stanowi pulę obiektów; to ona odpowiada za tworzenie nowych, pojedynczych obiektów i przechowywanie ich zamist tworzyć takowy obiekt każdorazowo
- Distributor to obiekt bazowy dla puli obiektów

Wzorzec puli obiektów - konsekwencje

- Zwiększenie wydajności poprzez racjonalne wykorzystywanie zasobów sprzętowych
- Lepsza hermetyzacja obiektowa

Wzorzec pyłkowy

- Od poprzednika rozrównia się następującymi cechami:

1) DistributorManagement staje się fabryką; jego zadanim jest tworzyć i zarządzać obiektami, lecz w tym momencie następuje podział na dane współdzielone i własne. Dane współdzielone są takie same (i niemodyfikowane) dla kolejnych instancji (obiektów)
2) Distributor jest pojedynczym obiektem, z którego korzysta fabryka
3) Dane współdzielone przechowywane są w osobnej bazie danych

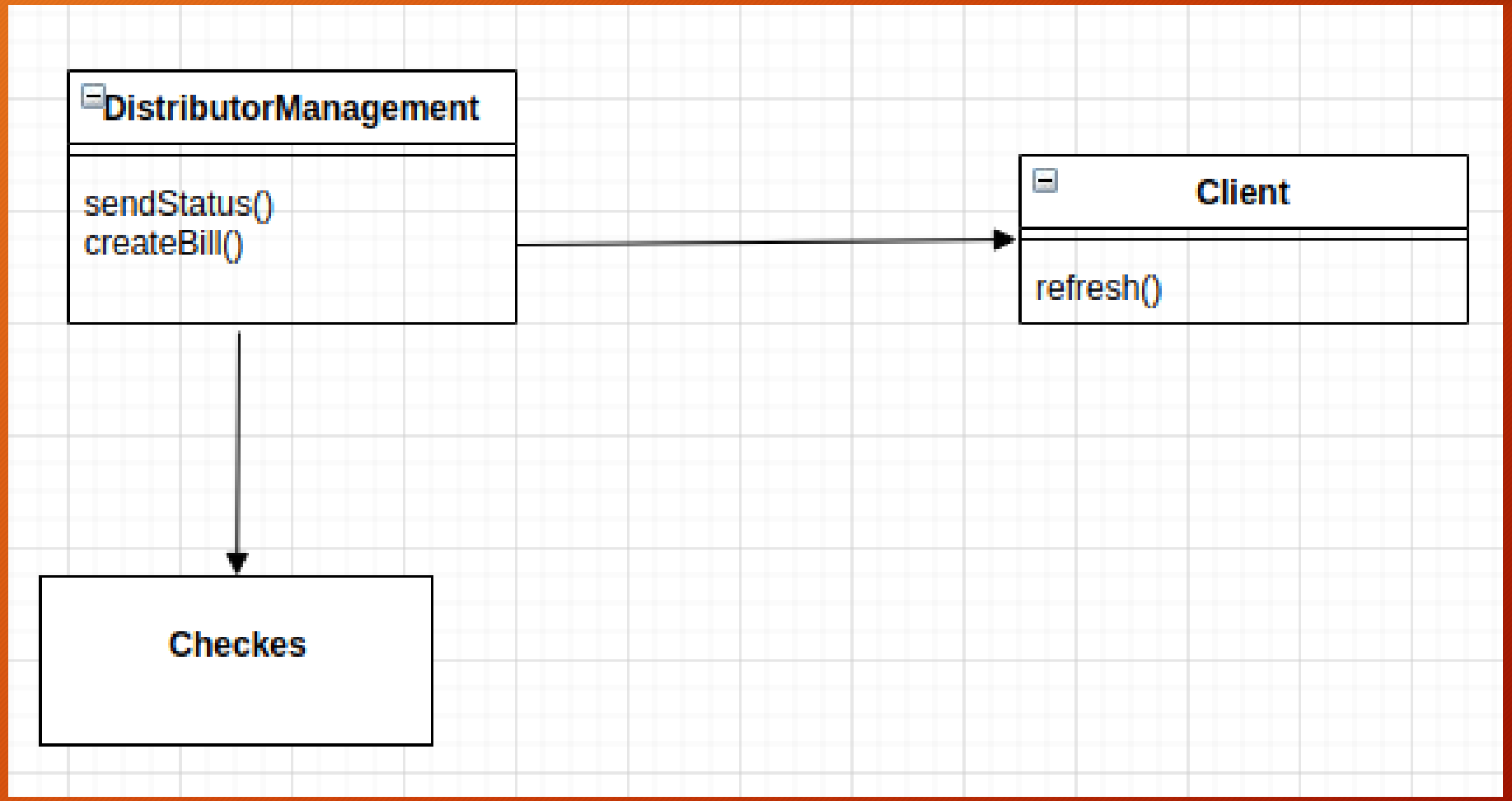
Wzorzec pyłkowy - konsekwencje

- Mniejsze zapotrzebowanie na pamięć
- Zewnętrzny stan może być przechowywany lub wyliczany
- Jednak należy pamiętać, że wzorzec wprowadza dodatkowy narzut na zarządzanie stanem zewnętrznym

Kontakt z klientem

- Klient może zakupić elementy w sklepie internetowym
- Musimy posiadać informacje o stanie produktu na magazynie
- Musimy posiadać informacje u kogo domówić towar
- KLient może dostawać informacje o ponownej dostępności wybranego towaru
- Klientowi możemy wysyłać informacje o statusie zamówienia

Powiadomienia



Wzorzec powiadomienia

- DistributionManagement posiada rejestr dystrybutorów oraz ich dostaw
- Pozwala na zamówienie elementów, które interesują klienta
- Rozsyła zmiany swojego statusu
- Client odbiera wysłane informacje i aktualizuje swój stan na ich podstawie

Wzorzec powiadomień - konsekwencje

- Luźne powiązania pomiędzy obiektami
- Rozgłaszanie komunikatów w oparciu o system zdarzeń
- Spójność stanu biorących udział klas

Materiały i odnośniki

- http://smurf.mimuw.edu.pl/external_slides/Wzorce_projektowe/Wzorce_projektowe.html
- http://zasoby.open.agh.edu.pl/~09sbfraczek/wzorce-projektowe-wstep%2C1%2C57.html
- https://refactoring.guru/pl/design-patterns/csharp
- http://smurf.mimuw.edu.pl/external_slides/Wzorce_projektowe/Wzorce_projektowe.html