Wzorce projektowe

Projektowanie obiektowe
Laboratorium

Czym jest wzorzec projektowy?

Ogólnie składa się z czterech kluczowych elementów:

1) **Nazwa wzorca** - skrót, którego można użyć do opisania w jednym lub dwóch słowach problemu projektowego, jego rozwiązania i konsekwencji zastosowania tego rozwiązania. Nazwanie wzorca bezpośrednio powiększa słownik projektowy i pozwala tworzyć projekty na wyższym poziomie abstrakcji.
2) Opis **problemu** określa, kiedy należy stosować wzorzec. Ta część wyjaśnia trudność i jej kontekst. Może to być omówienie specyficznego problemu projektowego, np. zapisu algorytmów w formie obiektów. Może to też być opis specyficznych dla nieelastycznych projektów struktur klas lub obiektów.

Czym jest wzorzec projektowy?

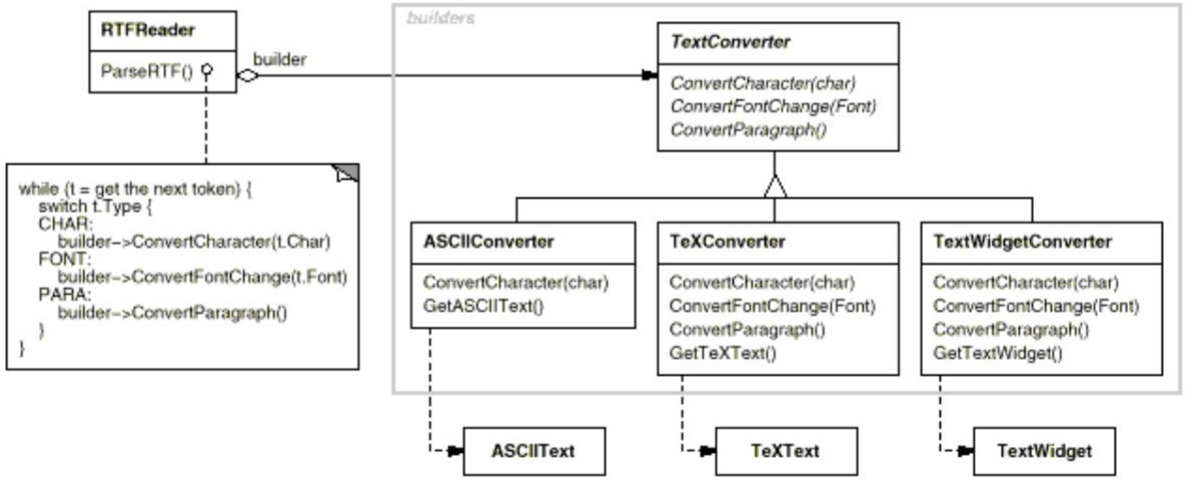
Ogólnie składa się z czterech kluczowych elementów:

3) **Rozwiązanie** to opis elementów składających się na projekt, ich przeznaczenia oraz relacji i współdziałania między nimi. Rozwiązanie nie jest wyjaśnieniem konkretnego projektu lub określonej implementacji, ponieważ wzorzec przypomina szablon - można go zastosować w wielu różnych sytuacjach.
4) **Konsekwencje** - efekty oraz koszty i zyski wynikające z zastosowania wzorca. Choć w opisie decyzji projektowych konsekwencje często nie są jawnie przedstawiane, mają kluczowe znaczenie przy ocenie różnych możliwości i pozwalając zrozumieć koszty oraz korzyści związane z danym wzorcem.

Wzorce konstrukcyjne - Builder

1. **Przeznaczenie**:
   Oddziela tworzenie złożonego obiektu od jego reprezentacji, dzięki czemu ten sam proces konstrukcji może prowadzić do powstawania różnych reprezentacji.
2. **Uzasadnienie**:
   Chcemy mieć czytnik dokumentów RTF, który powinien móc przekształcać dokumenty na wiele formatów tekstowych. Rozwiązaniem może być skonfigurowanie klasy RTFReader w taki sposób, aby za pomocą obiektu TextConverter przekształcała dokumenty RTF na inną reprezentację tekstową. Podklasy klasy TextConverter są wyspecjalizowane pod kątem różnych konwersji i formatów. Na przykład klasa ASCIIConverter ignoruje żądania związane z konwersją elementów innych niż zwykły tekst. Z kolei TexConverter obejmuje implementację operacji obsługujących wszystkie żądania, co umożliwia utworzenie reprezentacji w formacie TEX, uwzględniające wszystkie informacje na temat stylu tekstu.

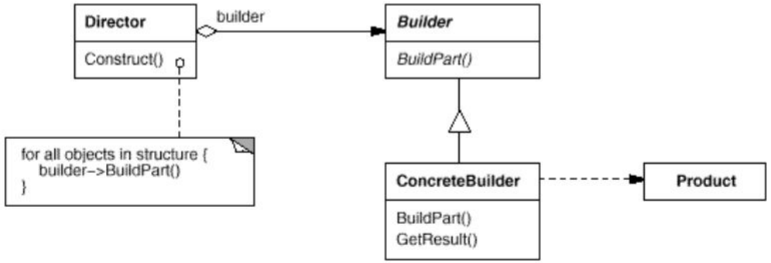
Wzorce konstrukcyjne - Builder



Każda klasa konwertująca przyjmuje mechanizm tworzenia i składania obiektów złożonych oraz ukrywa go za abstrakcyjnym interfejsem. Konwerter jest oddzielony od czytnika odpowiadającego za analizowanie dokumentów.

Wzorce konstrukcyjne - Builder

3. **Struktura:**

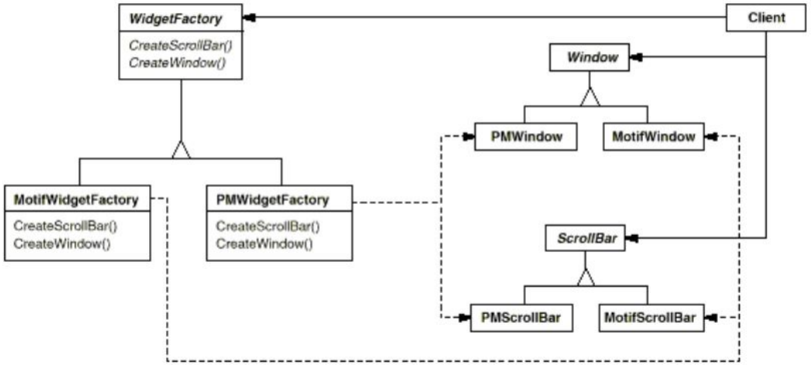


- Budowniczy (ang. Builder): określa interfejs abstrakcyjny do tworzenia składników obiektu Product.
- Budowniczy konkretny (ang. ConcreteBuilder): tworzy i łączy składniki produktu w implementacji interfejsu Builder, definiuje i śledzi generowane reprezentacje, udostępnia interfejs do pobierania produktów.
- Kierownik (ang. Director) (np. RTFReader): tworzy obiekty za pomocą interfejsu klasy Builder.
- Product: reprezentuje generowany obiekt złożony, obejmuje klasy definiujące składowe elementy obiektu.

Wzorce konstrukcyjne - Fabryka abstrakcyjna

1. **Przeznaczenie**:
   Udostępnia interfejs do tworzenia rodzin powiązanych ze sobą lub zależnych od siebie obiektów bez określenia ich klas konkretnych.
2. **Uzasadnienie**:
   Zastanówmy się nad pakietem narzędziowym do tworzenia interfejsów użytkownika, obsługującym różne standardy wyglądu i działania (np. Motif i Presenetation Manager). Poszczególne standardy wyznaczają różny wygląd i inne zachowania widgetów interfejsu użytkownika, takich jak paski przewijania, okna i przyciski. Aby aplikacja była przenośna pomiędzy różnymi standardami, nie należy w niej na stałe zapisywać określonego wyglądu i sposobu działania widgetów. Tworzenie określających te aspekty egzemplarzy klas w różnych miejscach aplikacji utrudnia późniejszą zmianę jej wyglądu i zachowania. Możemy w związku z tym stworzyć klasę abstrakcyjną WidgetFactory i zadeklarować w niej interfejs do tworzenia podstawowych widgetów. Należy przygotować też klasy abstrakcyjne dla poszczególnych rodzajów widgetów oraz podklasy konkretne z implementacją określonych standardów wyglądu i działania.

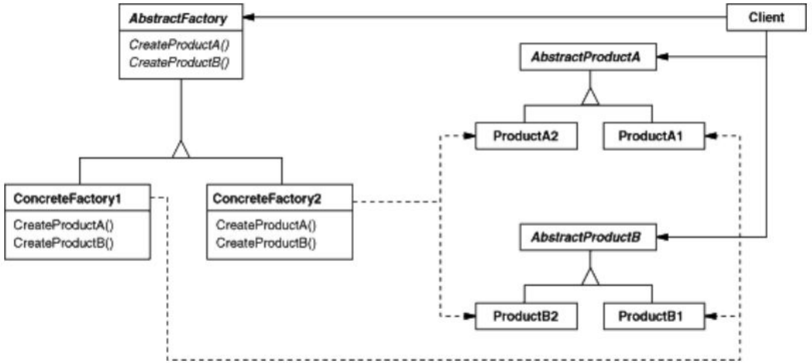
Wzorce konstrukcyjne - Fabryka abstrakcyjna



Dla każdego standardu wyglądu i działania istnieje podklasa konkretna klasy WidgetFactory. W każdej takiej podklasie zaimplementowane są operacje do tworzenia widgetów odpowiednich dla danego standardu. Na przykład operacja CreateScrollBar klasy MotifWidgetFactory tworzy i zwraca egzemplarz paska przewijania zgodnego ze standardem Motif.

Wzorce konstrukcyjne - Fabryka abstrakcyjna

3. **Struktura:**



- AbstractFactory (WidgetFactory) - obejmuje deklarację interfejsu z operacjami tworzącymi produkty abstrakcyjne,
- ConcreteFactory (MotifWidgetFactory, PMWidgetFactory) - obejmuje implementację operacji tworzących produkty konkretne,
- ConcreteProduct (MotifWindow, MotifScrollBar) - definiuje obiekt-produkt tworzony przez odpowiadającą mu fabrykę konkretną,
- Client - korzysta jedynie z interfejsów zadeklarowanych w klasach AbstractFactory i AbstractProduct.

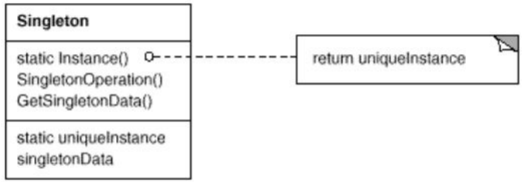
Wzorce konstrukcyjne - Singleton

1. **Przeznaczenie**:
   Gwarantuje, że klasa będzie miała tylko jeden egzemplarz i zapewnia globalny dostęp do niego.
2. **Uzasadnienie**:
   W przypadku niektórych klas ważne jest, aby miały tylko jeden egzemplarz. Choć w systemie może działać wiele drukarek, powinien być tylko jeden program buforujący drukowania. Potrzebny jest tylko jeden system plików i menadżer okien.

   Aby zagwarantować, że mamy tylko jeden egzemplarz można przydzielić klasie śledzenie swojego jedynego egzemplarza. W ten sposób klasa może zagwarantować, że nie powstanie kolejny pod warunkiem, że już jeden istnieje.

Wzorce konstrukcyjne - Fabryka abstrakcyjna

3. **Struktura:**



- Singleton - definiuje operację Instance umożliwiająca klientowi dostęp do niepowtarzalnego egzemplarza klasy; Instance często (np. w C++) jest operacją statyczną. Może odpowiadać za tworzenie swojego własnego, niepowtarzalnego egzemplarza (choć nie musi - tą rolę może przejąć np. fabryka).

Przydatne dodatkowe materiały

- https://sourcemaking.com/design_patterns/creational_patterns
- https://refactoring.guru/design-patterns/creational-patterns
- https://www.javatpoint.com/builder-design-pattern