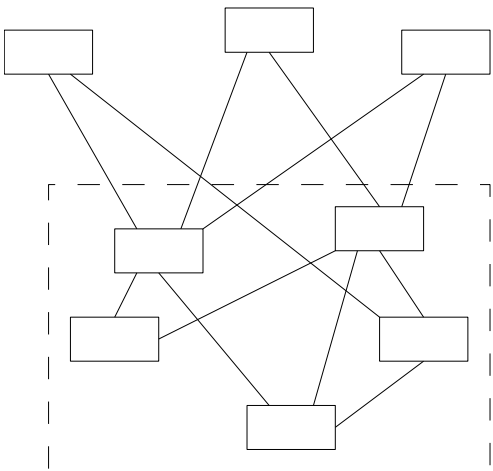
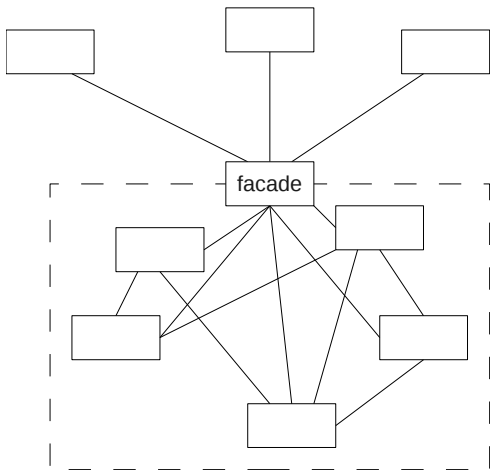
Façade (fasada)

Cel:

- Dostarcza prosty interfejs do złożonego podsystemu.

Struktura:





Składniki:

- Facade (fasada)
  - ma pełnię wiedzy o podsystemie, wie które klasy za co odpowiadają i w jaki sposób można ich użyć
  - oddelegowuje żądania klientów do właściwych obiektów podsystemu
- Klienci
  - zwracają się do podsystemu z żądaniami
  - oczekują bardziej abstrakcyjnego, wysokopoziomowego punktu dostępu, nie chcą polegać na wiedzy o strukturze podsystemu
- Klasy podsystemu
  - implementują funkcjonalność podsystemu
  - wykonują żądania fasady
  - nie mają żadnej informacji o fasadzie, ani od niej nie zależą (cały podsystem mógłby być używany bez niej)

Zależności:

Klient komunikuje się z podsystemem wysyłając żądania do fasady, ona deleguje je do odpowiednich obiektów.

Zastosowanie:

- zapewnia prosty interfejs do skomplikowanego podsystemu – domyślny widok, dobry dla większości klientów
- gdy istnieje wiele zależności między klientem a klasami podsystemu – fasada oddziela podsystem od klienta i podsystemy od siebie nawzajem, podnosząc przenośność i niezależność
- w systemie rozdzielonym na warstwy fasada może służyć jako punkt wejścia do danej warstwy

Konsekwencje:

1. oddziela klienta od komponentów podsystemu, zmniejszając liczbę obiektów, z którymi klient musi się dogadywać, co czyni podsystem łatwiejszym w użyciu
2. wspiera luźne powiązanie klienta z podsystemem, podczas gdy komponenty podsystemu są ściśle powiązane ze sobą nawzajem; pomaga to też przy przenoszeniu aplikacji na inne platformy, zmniejsza czas kompilacji, etc.
3. jeśli aplikacja chce korzystać z podsystemu poza fasadą, możemy jej na to pozwolić (może być to dostępne jako funkcjonalność „nieudokumentowana”, czyli „na własne ryzyko”)
4. jeśli fasada jest abstrakcyjna, jest możliwość łatwej wymiany całego podsystemu

Przykład:

```
class MessageFacade {
     @Resource(mappedName="jms/ConnectionFactory")
     private static cf connectionFactory;
     @Resource(mappedName="jms/Queue")
     private static Queue queue;
     private MessageProducer producer;
     public MessageFacade() {
          Connection connection = cf.createConnection();
          Session session = connection.createSession(false,
                                   Session.AUTO_ACKNOWLEDGE);
          producer = session.createProducer(queue);
     }
     public void send(String msg) {
          TextMessage message = session.createTextMessage();
          message.setText(msg);
          producer.send(message);
     }
}
```

Powiązania:

- Abstract Factory – podobnie jak fasada, ukrywa klasy specyficzne dla danej implementacji
- Mediator – podobnie jak fasada skupia pewną funkcjonalność i zapewnia łączność między elementami systemu
- Singleton – zazwyczaj chcemy tylko jednej fasady