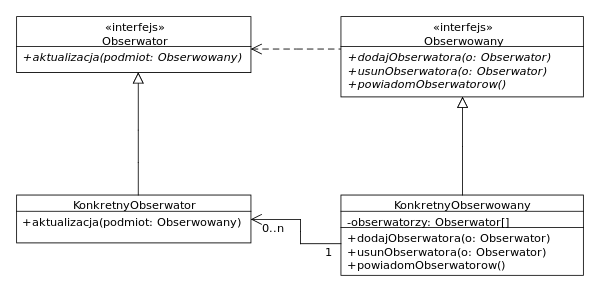
***Projekt z przedmiotu Wzorce Projektowe***

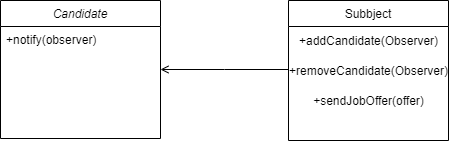
**Wykonał:** Arkadiusz Pizon 10262.

**Obserwator** znany też jako:Event-Subscriber, Listener, Observerto czynnościowy / operacyjny (behawioralny) wzorzec projektowy pozwalający zdefiniować mechanizm subskrypcji w celu powiadamiania wielu obiektów o zdarzeniach dziejących się w obserwowanym obiekcie.

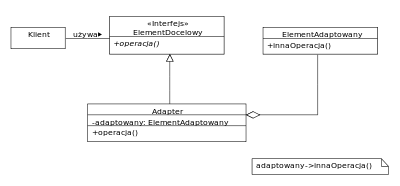


**Przykładem** zastosowania jest fragment kodu strony internetowej który odpowiada za rozsyłanie ogłoszeń w odpowiedzi na reakcje zasubskrybowanie powiadomień o ofertach przez użytkowników. Obiekt Subject przechowuje listę obserwatorów, którzy go nasłuchują w oczekiwaniu na zmianę jego stanu. Kiedy ona nastąpi Obiekt Subject rozsyła informacje do wszystkich obserwujących go użytkowników wywołując Metodę sendJobOffer.

Co jest widoczne w przykładzie zamieszczonym w folderze Obserwator > Przykład, reprezentowanym na poniższym diagramie UML:



**Adapter** znany jest także jako: Opakowanie, Nakładka, Wrapper jest strukturalnym wzorcem projektowym pozwalającym na współdziałanie ze sobą obiektów o niekompatybilnych interfejsach.



**Przykładem** zastosowania jest fragment gry komputerowej w grze komputerowej występują różne wrogie jednostki, tj. czołg lub robot. Każda z tych jednostek ma możliwość użycia swojej broni, jazdy do przodu oraz przypisania operatora, sterującego tą jednostką. W tym fragmencie znajdują się interfejs nazwany EnemyAttacker klasa EnemyTank implementująca klasę EnemyAttacker oraz klasę TestEnemyAttacker która ma sterować czołgiem. Występuje również klasa EnemyRobot posiadająca odmienny interfejs aby klasy enmyAttacker oraz EnemyRobot mogły ze sobą współpracować użyto adaptera w postaci klasy EnemyRobotAdapter.

Co jest widoczne w przykładzie zamieszczonym w folderze Adapter > Przykład, reprezentowanym na poniższym diagramie UML:



**Singleton** jest kreacyjnym wzorcem projektowym, który pozwala zapewnić istnienie wyłącznie jednej instancji danej klasy. Ponadto daje globalny punkt dostępowy do tejże instancji.

Obraz zawierający stół

Opis wygenerowany automatycznie

**Przykładem** zastosowania wzorca Singleton jest utworzenie klasy która umożliwia połączenie z bazą danych, gdyż użytkownik łączy się z nią tylko raz w trakcie jednej sesji. Mamy naszą klasę, która implementuje statyczną metodę getInstance. Jej zadaniem jest sprawdzenie czy instancja nie została już utworzona. Jeżeli nie, to taki obiekt zostaje stworzony a następnie zostaje wywołana metoda, nawiązująca połączenie z bazą (makeConnection). Jest tu również prywatny konstruktor, który zapobiega przed utworzeniem instancji tej klasy w innym miejscu w kodzie. Czyli instancja klasy DatabaseConnector może być utworzona jedynie przez samą siebie, za pośrednictwem zaimplementowanej metody statycznej.

Co jest widoczne w przykładzie zamieszczonym w folderze Singleton > Przykład, reprezentowanym na poniższym diagramie UML:

