**SOLID:**

**S(SRP)** – Klasa ma tylko jedną odpowiedzialność : Zmiana zależności pomiędzy klasami.

**O(OCP)** - otwarty na rozszerzenia, zamknięty na modyfikacje: Składniki oprogramowania(klasy) zostały otwarte na rozszerzenia i zamknięte(na modyfikacje wpływające od klientów).

L(LSP) - Każda obiekt klasy w kontekście swojego użycia powinien być zastępowalny przez obiekt klasy potomnej : Brak dziedziczenia, nie dotyczy.

**I(ISP**) - Klient nie powinien być zmuszany do zależności od metod, których nie używa : Klient jest w zależności jedynie z rachunkiem, z którą ma interakcję.

D(DIP) - Moduły wyższego poziomu zależą od abstrakcji, nie od implementacji : Brak abstrakcji, nie dotyczy.

**GRASP:**

**Creator** - Określa, kiedy pierwszy obiekt powinien tworzyć inny obiekt : Modyfikacja strzałek przy recepcjoniście. Przydzielone zobowiązanie tworzenie instancji klas rezerwacja, rachunek w klasie recepcjonista. Zmiana sposobu tworzenia w diagramach sekwencji.

**Information Expert** - Której klasie przypisać daną odpowiedzialność/zadanie? : Recepcjonista posiada wszystkie informacje o rezerwacji i rachunku. Jest „information expertem”.

Controller - Który obiekt poza GUI (interfejsem użytkownika) powinien obsłużyć żądania systemu? : Nie dotyczy, brak typowej klasy odpowiedzialnej za kontrole.

**Low Coupling** - Jak ograniczyć zakres zmian w systemie w momencie zmiany fragmentu systemu? : Elementy są ze sobą sprężone gdy między nimi występuje agregacja/kompozycja oraz gdy element implementuje/rozszerza inny element. Modyfikacja zależności pomiędzy klasami.

**High Cohesion** - Jak sprawić by obiekty miały jasny cel, były zrozumiałe i łatwe w utrzymaniu? Klasy wewnątrz modułu są ze sobą ściśle sprzężone (korzystają z siebie nawzajem). Przykład Klient<->Recepcjonista

Indirection  - Komu przydzielić zobowiązanie, jeśli zależy nam na uniknięciu bezpośredniego powiązania między obiektami? Brak klasy pośredniej w projekcie. Nie dotyczy. Zbyt niskie zaawansowanie projektu by używać klasy pośredniej.

Polymorphism - Co zrobić, gdy odpowiedzialność różni się w zależności od typu? – Brak polimorfizmu w projekcie. Nie dotyczny.

Protected Variations - Jak projektować obiekty, by ich zmiana nie wywierała szkodliwego wpływu na inne elementy? – Brak interfejsów w aktualnej fazie tworzenia projektu. Nie dotyczny

Pure Fabrication - Jak przydzielić odpowiedzialność by nie naruszyć zasad High Cohesion i Low Coupling a nie odpowiada nam rozwiązanie sugerowane przez Information Expert? – Brak klasy pomocniczej w projekcie. Nie dotyczy.