**Java – projekt kasy fiskalnej**

**Wymagania Funkcjonalne**

1. **Interfejs Konsolowy**: Aplikacja powinna posiadać prosty interfejs użytkownika w konsoli, który pozwala na wprowadzanie danych produktów i obsługę transakcji.
2. **Wprowadzanie Produktów**: Użytkownik powinien mieć możliwość wprowadzenia nazwy produktu, ceny jednostkowej oraz ilości.
3. **Obliczanie Podsumowania**: Aplikacja powinna automatycznie obliczać łączną kwotę do zapłaty po wprowadzeniu każdego produktu.
4. **Generowanie Paragonu**: Po zakończeniu wprowadzania produktów i potwierdzeniu transakcji, aplikacja powinna generować paragon z listą produktów, ich cenami, łączną kwotą oraz datą i godziną transakcji.
5. **Obsługa Błędów**: Aplikacja powinna obsługiwać błędy wprowadzania danych (np. nieprawidłowy format ceny) i wyświetlać odpowiednie komunikaty.
6. **Historia Transakcji**: Opcjonalnie, aplikacja może przechowywać historię transakcji do późniejszego wyświetlenia.

**Wymagania Niefunkcjonalne**

1. **Język Programowania**: Aplikacja powinna być napisana w Javie, z minimalnym wykorzystaniem niestandardowych klas z API Javy.
2. **Prostota**: Aplikacja powinna być prosta w obsłudze i przeznaczona dla użytkowników bez zaawansowanej wiedzy technicznej.
3. **Wydajność**: Aplikacja powinna szybko przetwarzać dane i generować paragony bez znaczących opóźnień.
4. **Skalowalność**: Kod powinien być napisany w taki sposób, aby umożliwić łatwe rozszerzenie funkcjonalności aplikacji w przyszłości.
5. **Przenośność**: Aplikacja powinna być łatwa do uruchomienia na różnych systemach operacyjnych obsługujących Javę.
6. **Testowalność**: Kod powinien być napisany w taki sposób, aby ułatwić testowanie jednostkowe i integracyjne.
7. **Dokumentacja**: Aplikacja powinna być dostarczona z dokumentacją, która opisuje jej działanie i sposób użycia.
8. **Bezpieczeństwo**: Chociaż aplikacja jest symulatorem, powinna być zaprojektowana z uwzględnieniem podstawowych zasad bezpieczeństwa oprogramowania.

# Przykładowy schemat klas

Na podstawie wymagań funkcjonalnych i niefunkcjonalnych dla aplikacji konsolowej symulującej działanie kasy fiskalnej, można zaprojektować następujący układ klas, rekordów i interfejsów:

### 1. Interfejs ITransaction

Reprezentuje pojedynczą transakcję. Może zawierać metody takie jak startTransaction(), addItem(Item item), finalizeTransaction().

### 2. Klasa Item

Reprezentuje pojedynczy produkt lub usługę. Zawiera pola takie jak:

* String name – nazwa produktu/usługi.
* double price – cena produktu/usługi.
* String category – kategoria produktu/usługi.

### 3. Klasa Receipt

Reprezentuje paragon/fakturę. Zawiera:

* List<Item> items – lista produktów/usług na paragonie.
* double totalPrice – łączna wartość paragonu.
* DateTime date – data i czas transakcji.

### 4. Interfejs IPayment

Reprezentuje płatność. Może zawierać metody takie jak processPayment(double amount).

### 5. Klasa CashPayment implementująca IPayment

Reprezentuje płatność gotówką. Zawiera implementację metody processPayment.

### 6. Klasa CardPayment implementująca IPayment

Reprezentuje płatność kartą. Zawiera implementację metody processPayment.

### 7. Klasa Transaction implementująca ITransaction

Reprezentuje realizację transakcji. Zawiera:

* Receipt receipt – paragon/faktura.
* IPayment paymentMethod – metoda płatności.
* Metody z interfejsu ITransaction.

### 8. Klasa CashRegister

Główna klasa symulująca kasę fiskalną. Zawiera:

* List<Transaction> transactions – historia transakcji.
* Metody do zarządzania transakcjami i płatnościami.

### 9. Interfejs IUserInterface

Reprezentuje interfejs użytkownika. Może zawierać metody do wyświetlania menu, przyjmowania wejść od użytkownika itp.

### 10. Klasa ConsoleInterface implementująca IUserInterface

Reprezentuje konsolowy interfejs użytkownika. Zawiera implementację metod z IUserInterface.

### Zależności i przepływ danych:

* CashRegister tworzy i zarządza Transaction.
* Transaction używa Item, Receipt i implementacji IPayment.
* CashRegister komunikuje się z użytkownikiem przez ConsoleInterface.