Przeanalizuj powyższy kod klasy Employee i zidentyfikuj naruszenia zasad SOLID. Następnie przeprojektuj kod tak, aby spełniał wszystkie zasady SOLID. W szczególności zwróć uwagę na:

Single Responsibility Principle (SRP)
Open/Closed Principle (OCP)
Liskov Substitution Principle (LSP)
Interface Segregation Principle (ISP)
Dependency Inversion Principle (DIP)

Wskazówki:

Klasa Employee ma obecnie zbyt wiele odpowiedzialności.
Obliczanie podatku powinno być bardziej elastyczne i otwarte na rozszerzenia.
Rozważ użycie interfejsów do oddzielenia różnych funkcjonalności.
Zastanów się nad wprowadzeniem abstrakcji dla operacji zapisu do bazy danych.

Twoim zadaniem jest przeprojektowanie tego kodu, tworząc odpowiednie klasy, interfejsy i implementując zasady SOLID. Pamiętaj o dobrym nazywaniu klas i metod oraz o dodaniu komentarzy wyjaśniających wprowadzone zmiany.

```
public class Employee {
    private String name;
    private String position;
    private double salary;

    public Employee(String name, String position, double salary) {
        this.name = name;
        this.position = position;
        this.salary = salary;
    }

    public void saveEmployee() {
        // Zapis do bazy danych
        System.out.println("Zapisano pracownika do bazy danych");
    }

    public void calculateTax() {
        if (position.equals("Manager")) {
            System.out.println("Podatek dla managera: " + (salary * 0.2));
        } else if (position.equals("Developer")) {
            System.out.println("Podatek dla developera: " + (salary * 0.15));
        } else {
            System.out.println("Podatek dla innych stanowisk: " + (salary * 0.1));
        }
    }

    public void printEmployeeDetails() {
        System.out.println("Imię: " + name);
        System.out.println("Stanowisko: " + position);
        System.out.println("Wynagrodzenie: " + salary);
    }
}
```