

SE322: Software Design and Architecture

Activity 6: Principles of Detailed Design

Objective: In this activity, the students will learn how to analyze an existing software architecture or design to check if the code violates principles or not

Expected Outcome: After completing this exercise, the students will be able to:

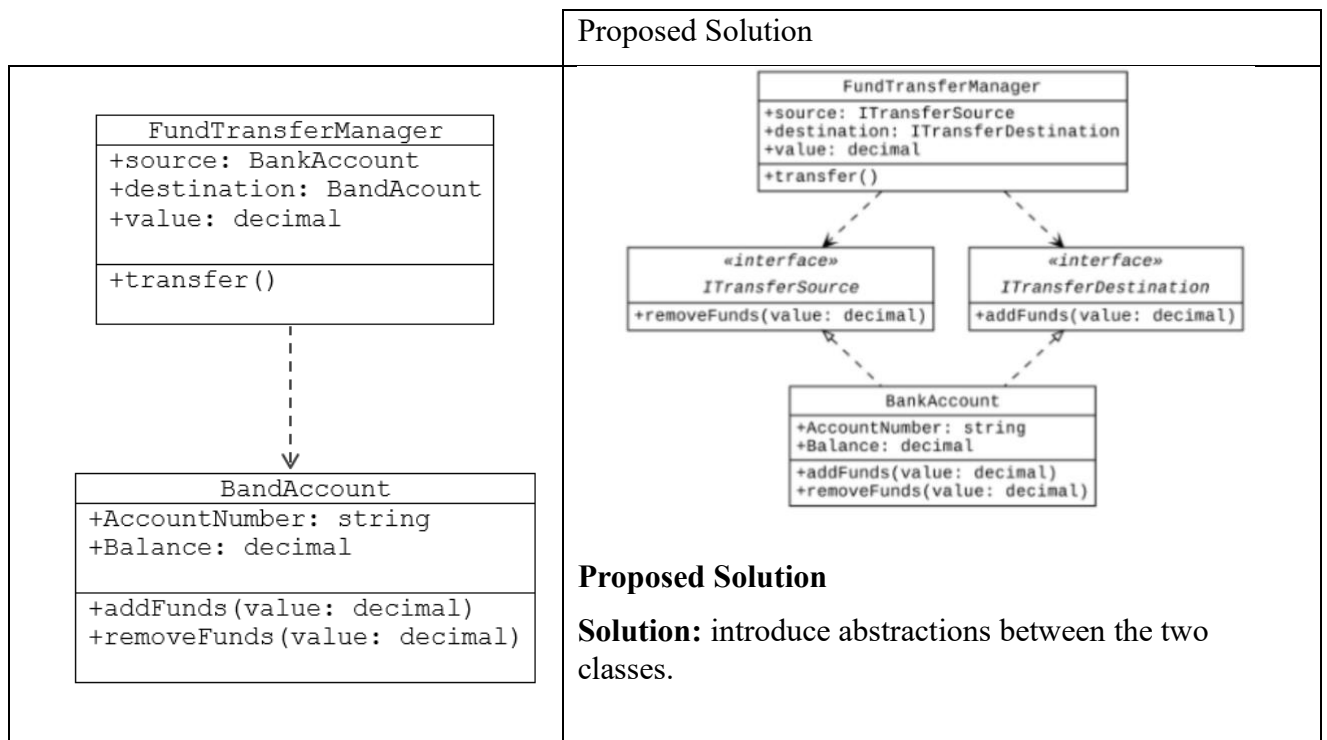
1. Analyze code to check if the code violates principles or not
2. Analyze class diagram to check if the diagram violates principles or not

Course Learning Outcome:

1. CLO 1: Recognize the principles, strategies, and process of software design and architecture.
2. CLO 2: Analyze an existing software architecture or design.

Question1:

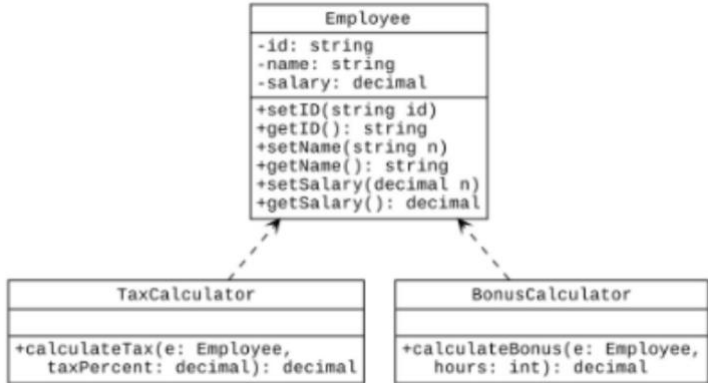
1. Consider the class diagram does it violate the design principles?
2. If it is violated, redraw the diagram in a way to comply with the principles.



	• ITransferSource declares removeFunds(); • ITransferDestination declares addFunds(). • FundTransferManager now depends only on the two interfaces (dashed arrows). • BankAccount implements both interfaces (realization arrows).
Violated Principle is Dependency Inversion Principle (DIP)	Both high-level and low-level classes now depend on abstractions, so any new account type that implements the interfaces can be transferred without changing FundTransferManager. Splitting into two small interfaces also keeps each client using only the methods it needs (ISP).

Question2:

1. Consider the class diagram does it violate the design principles?
2. If it is violated, redraw the diagram in a way to comply with the principles.

	Proposed Solution
<pre> classDiagram class Employee { -id: string -name: string -salary: decimal +setID(string id) +getID(): string +setName(string n) +getName(): string +setSalary(decimal n) +getSalary(): decimal +calculateTax(decimal taxPercent): decimal +calculateBonus(int hours): decimal } </pre>	 Proposed Solution Solution: split the responsibilities into separate classes.

Violated Principle is: Single Responsibility Principle (SRP)	• Employee only stores and manages employee data. • TaxCalculator only calculates tax. • BonusCalculator only calculates bonus. The calculators use Employee (dashed dependency arrows) to read the salary. Each class now has one responsibility and one reason to change, so it is easier to maintain, test and extend.
--	---