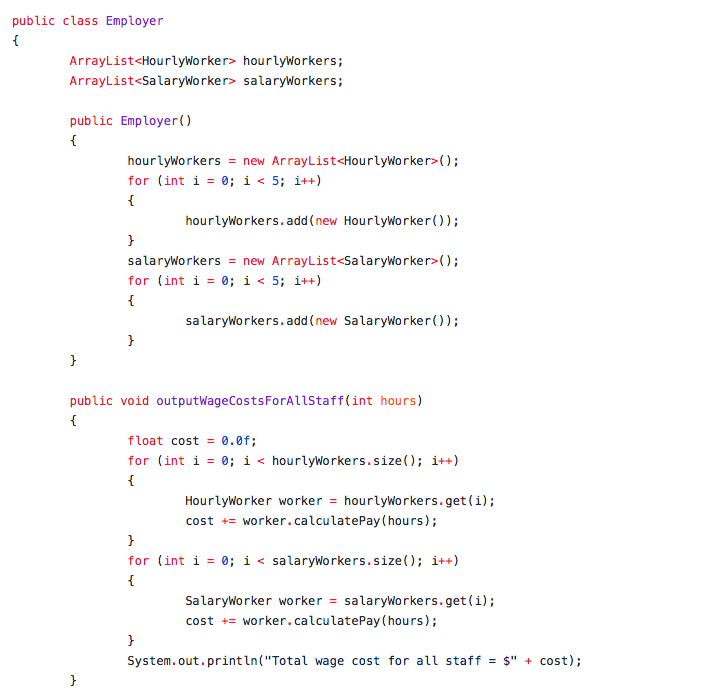
**Assignment 2**

**Question 1**

1. **Which S.O.L.I.D. principle does the Employer class violate?**

The Employer Class violates Dependency Inversion Principle (DIP) which states that a class must depend on abstraction rather than concretion.

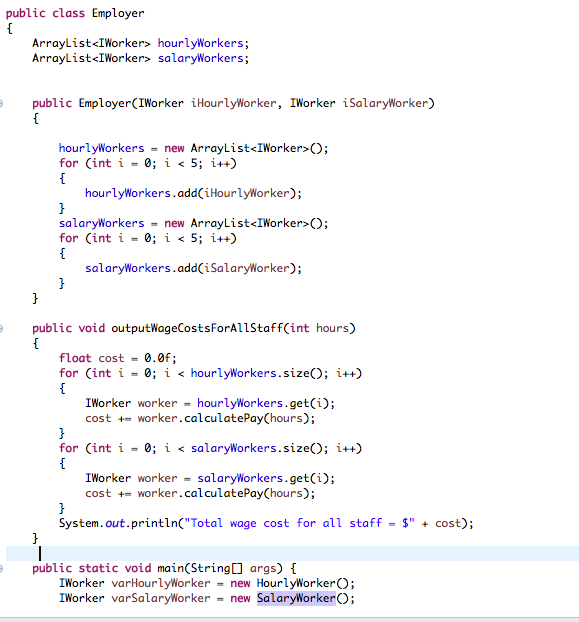
1. **Why does the code violate the principle?**

****

Employer class depend on other concrete classes such as HourlyWorker and SalaryWorker. Any change in these classes affect Employer class also. This is an example of tight coupling.

1. **Write code that fixes the violation.**

* An interface is created IWorker that establishes a contract on the implementing class.
* Both the classes HourlyWorker and SalaryWorker implement IWorker interface
* By employing this technique, loose coupling is established.
* Employer class is dependent on abstraction rather than concretion.
* The modified code files are present in Question1 folder



**Question 2**

1. **Which S.O.L.I.D. principle does the following code violate?**

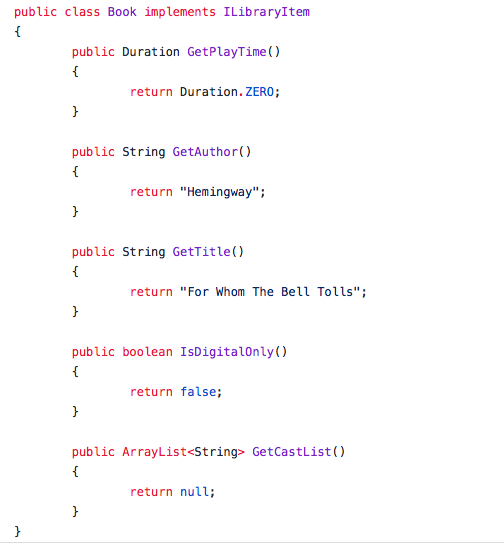
The question violates Interface Segregation Principle (ISP). which states to use many client specific interfaces than a general purpose interface.

1. **Why does the code violate the principle?**

* ILibrayItem interface consists of several methods such as GetPlayTime(), GetAuthor() and IsDigitalOnly() that are common to both DVD and Book classes.



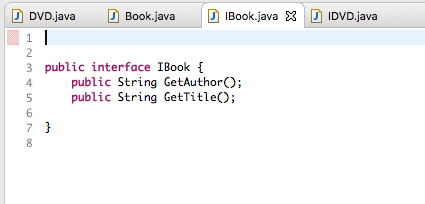
* The common interface forces the implementing classes to define irrelevant methods.

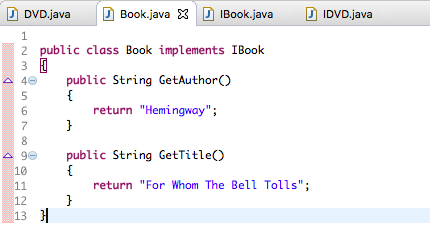


* This is an example of overhead complexity.

1. **Write code that fixes the violation.**

* Two interfaces namely, IDVD and IBook are created with segregated method implementations.
* IDVD is implemented by DVD class and IBook interface is implemented by Book class.
* The modified code files are present in Question2 folder.





**Question 3**

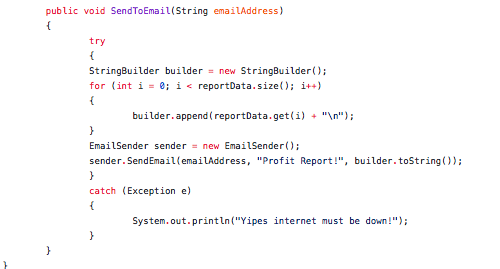
1. **Which S.O.L.I.D. principle does the Profit Report class violate?**

The question violates Single Responsibility Principle (SRP). which states that a class must take responsibility over a single function of the system. A class must be changed for a single reason only.

1. **Why does the code violate the principle?**

* Profit Report class consists of three major responsibilities such as report generation, printer and email functionalities.

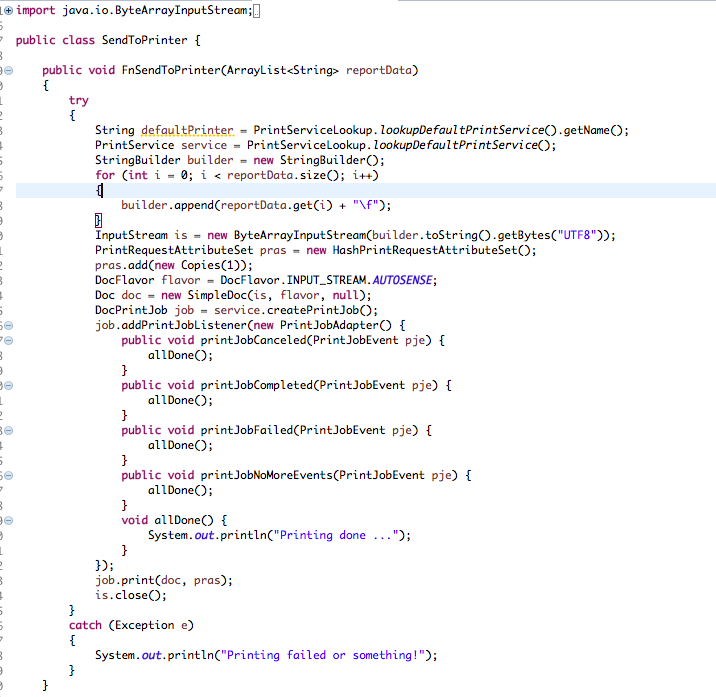




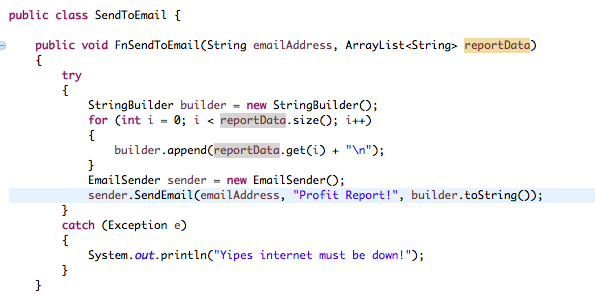
* Since the class has three functionalities, any change in these functionalities require to change this base class. This affects other functionalities available in that class.
* In an ideal architecture, a class must change only for one reason.

1. **Write code that fixes the violation.**

* Two new classes SendToPrinter and SendToEmail are generated.
* SendToPrinter consists of printer functionality which is removed from ProfitReport class. ReportData is sent as parameter to the function.



* Similarly, SendToEmail consists of email functionality which is removed from ProfitReport class. ReportData is sent as parameter to the function.



* ProfitReport class is now responsible for the single responsibility of report generation.
* The modified code files are present in Question3 folder.

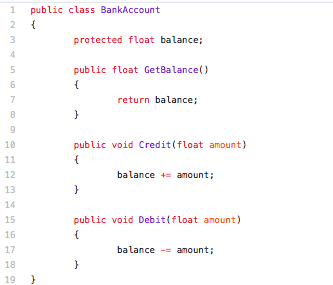
**Question 4**

1. **Which S.O.L.I.D. principle does the following code violate?**

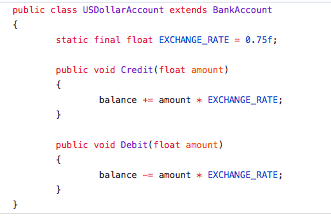
The question violates Open - Closed Principle (OCP). which states that software entities must be open for extension but closed for modification.

1. **Why does the code violate the principle?**

* BankAccount class consists of methods such as Credit(), Debit() that perform addition and subtraction functionalitites.



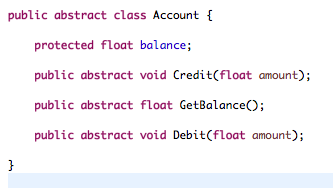
* USDollarAccount class extends BankAccount class and it modifies Credit() and Debit() methods with additional functionalities..



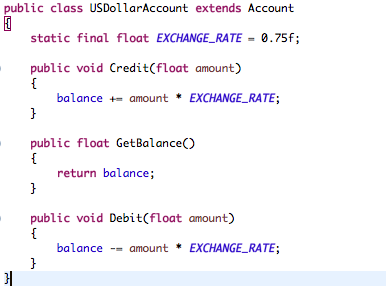
* If the requirements further change, then the methods in the base class is further modified which causes coupling and code complexity.

1. **Write code that fixes the violation.**

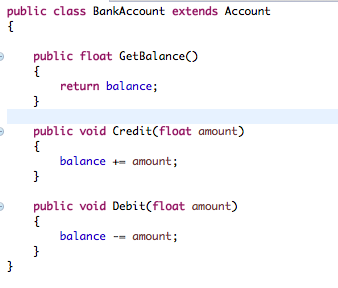
* A new abstract class Account is created with all the abstract methods.



* BankAccount extends this abstract class and define their corresponding methods.



* Similarly, USDollarAccount extends this abstract class and define their corresponding methods.



* By this way, loose coupling is established. Also, Open – Closed Principle is followed.
* The modified code files are present in Question4 folder.

**Question 5**

1. **Which S.O.L.I.D. principle does the following code violate?**

The Employer Class violates Dependency Inversion Principle (DIP) which states that a class must depend on abstraction rather than concretion.

1. **Why does the code violate the principle?**

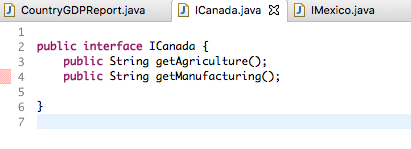
* CountryGDPReport class is dependent on concrete classes Canada and Mexico to access methods such as getAgriculture() and getManufacturing().



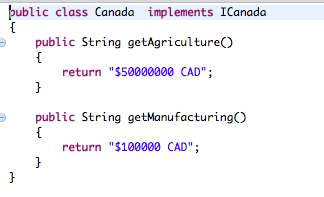
* This results in tight coupling and CountryGDPReport class is dependent on Canada and Mexico classes. The classes are vulnerable to code change complexities.

1. **Write code that fixes the violation.**

* Two interfaces ICanada and IMexico are created.



* The interfaces are implemented by the respective classes.



* By employing this technique, loose coupling is established.



* Employer class is dependent on abstraction rather than concretion.
* The modified code files are present in Question5 folder

**Question 6**

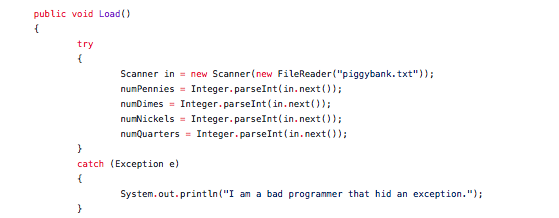
1. **Which S.O.L.I.D. principle does the Piggybank class violate?**

The question violates Single Responsibility Principle (SRP). which states that a class must take responsibility over a single function of the system. A class must be changed for a single reason only.

1. **Why does the code violate the principle?**

* Piggybank class consists of three major responsibilities such as loading, printing and adding functionalities.

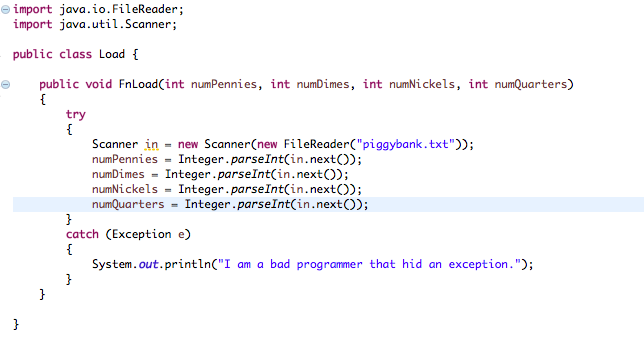


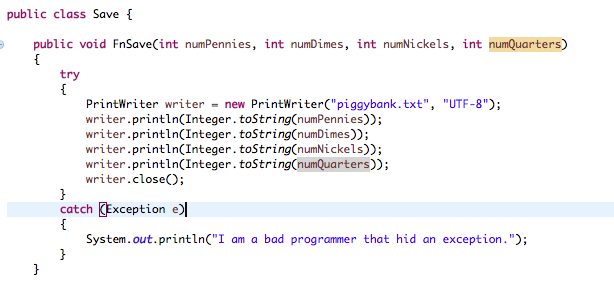


* Since the class has three functionalities, any change in these functionalities require to change this base class. This affects other functionalities available in that class.
* In an ideal architecture, a class must change only for one reason.

1. **Write code that fixes the violation.**

* Two new classes Load and Save are generated.
* Load class consists of loading functionality while Save class consists of saving functionality.





* Piggybank class is now responsible for the single responsibility of report generation.
* The modified code files are present in Question6 folder.

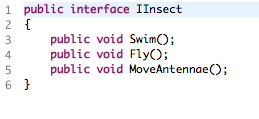
**Question 7**

1. **Which S.O.L.I.D. principle does the following code violate?**

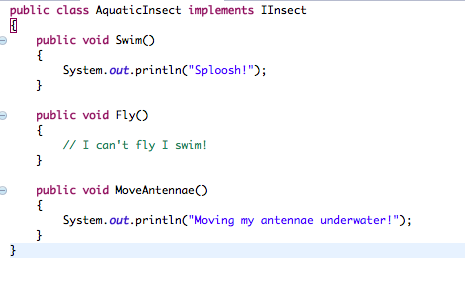
The question violates Liskov Substitution Principle (LSP). which states that the objects in a program must be replaceable with the instances of the subtypes.

1. **Why does the code violate the principle?**

* Interface IInsect class consists of methods such as Swim(), Fly() and MoveAntennae().



* AquaticInsect class has implementations for all interface methods except one.



.

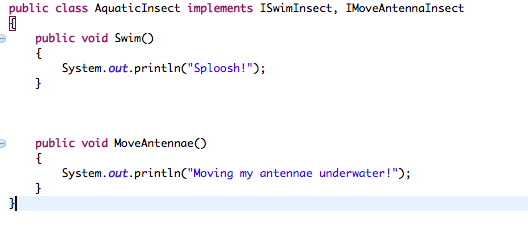
* The Fly() method in AquaticInsect class does not have a proper definition and this could lead to NotImplemented Exception.
* Therefore, this case violates Liskov Substitution Principle.

1. **Write code that fixes the violation.**

* This violation could be solved by multiple interfaces.
* Three interfaces IFlyInsect, ISwimInsect and IMoveAntennaInsect are created with appropriate method contracts.



* Since AquaticInsect class has proper implementations for Swim() and MoveAntenna() methods, it implements ISwimInsect and IMoveAntennaInsect interfaces.



* The modified code files are present in Question7 folder.