**ASSIGNMENT SOLUTION ON SOLID PRINCIPLES**

**QUESTION1:**

1. **The S.O.L.I.D principle violated is: Dependency Inversion Principle**.
2. **It violates the principle because:** **The high-level class Employer depends on the low-level classes HourlyWorker and SalaryWorker, if we added more worker types or more methods in the Employer class we’d have to do a lot of work in the Employer class.**
3. **See attached Code**

**QUESTION2:**

1. **The S.O.L.I.D principle violated is:  Interface Segregation Principle**
2. **It violates the principle because:** ILibraryItem **interface has too many methods. A module should never be forced to implement an interface that it doesn't use. Hence module shouldn't be forced to depend on methods they do not use.**

**Code Implementation Description:** Split the interface ILibraryItem into smaller interfaces.

We need to create two interfaces called:

1. INTBook that has the GetAuthor and GetTitle method, then the Book Class can implement this interface.
2. INTDVD that has the GetCastList and GetPlayTime, GetTitle method, then the Book Class can implement this interface.

**QUESTION3:**

1. **The S.O.L.I.D principle violated is: Single Responsibility Principle**
2. **It violates the principle because: class ProfitReport does more than one functionality such as SendToPrinter, SendToEmail, CreateReport. This violates the SRP, a class should only do one thing.**

**Code Implementation Description:** We need to create separate classes and delegate the SendToPrinter and SendToEmail responsibility to the classes respectively.

i. MyPrintReport Class has the method SendToPrinter and handle printing jobs.

Ii. MySndReprtToEmail Class has the method SendToEmail method that handles report to emails.

**QUESTION4:**

1. **The S.O.L.I.D principle violated is:**  **Liskov Substitution Principle**
2. **It violates the principle because: USDAccount class which is a subtype of BankAccount couldn’t provide the necessary substitution behavioral of BankAccount, it made changes to it within its own call.**

**Code Implementation Description**: I created an interface MYExchangeRate and the class BankAccount would implement the interface

**QUESTION5:**

1. **The S.O.L.I.D principle violated is:  Dependency Inversion principle**
2. **It violates the principle because: CountryGDPReport being the higher module is made to depend on both lower modules Canada and Mexico respectively, but the principles states that the dependency should be on abstraction and not on *concretions***

**Code Implementation Description**: I created interface INTCanada and INTMexico since the dependency should through Abstraction.

**QUESTION6:**

1. **The S.O.L.I.D principle violated is:   Single Responsibility Principle**
2. **It violates the principle because: PiggyBank has to many functionalities such as Load and Save. The class should just be written to perform only one functionality.**

**Code Implementation Description:**  Create two separate class MYLoadCurrency and MYSaveCurrency to handle the load and reduce the functionality of PiggyBank.

**QUESTION7:**

1. **The S.O.L.I.D principle violated is:  Interface Segregation Principle**
2. **It violates the principle because: IInsect interface is overloaded too many methods. A client should never be forced to implement an interface that it doesn't use, however, clients shouldn't be forced to depend on methods they do not use**

**Code Implementation Description:**  Split the interface IInsect into smaller interfaces, by creating another interface called:

 INTFlyingInsect that has the Fly method, then the FlyingInsect Class can implement this interface.

 INTAquaticInsect that has the Swim method, then the AquaticInsect Class can implement this interface.