CSCI5308
Quality Assurance
Assignmen-2

Bhumi Patel
B00824756

---

1) **Question – 1**

   Principle Violation: **Single responsibility principle.**

   Solution: Class 'Customer' violate the single responsibility principle, because this class has method called 'emailCustomerSpecialOffer' which violate the single resposibilty of Customer class. It is irrelevant method in Customer Class. There should be separate class which has methods related to email. (May be in future emailValidation method can be added).

   To solve this issue, I create a new class called 'Email'. Which has method 'emailCustomerSpecialOffer'. I also created 'IEmailSender' interface and 'EmailSender' class implements this interface. In future it can be possible to different possible way to sending Email to user. Interface make this situation flexible and make it independent.

2) **Question – 2**

   Principle Violation: **Liskov-Substitution-Principle.**

   .

   Solution: 'USDollarAccount' violate the liskov-Substitution principle. This class extends the 'Account' class and change the behaviour of the methods of super class which is violation of principle.

   To solve this issue, I made Account class as abstract class and made two methods 'credit' and 'debit' as abstract methods.

3) **Question – 3**

Principle Violation: **Single responsibility principle**.

Solution: Student class violate a Single responsibility principle, because this class has methods ‘save’ and ‘load’ which is not related to Student class which violate the single responsibility of this class.

To solve this issue, I created new interface called ‘IStudentFileStorage’ and declare two methods save and load in it. Class ‘studentFileStorage’ implements these two methods. Now Student class has single responsibility having getter setter methods of its student class fields. ‘studentFileStorage’ class also have single type of responsibility read and write file.

4) **Question – 4**

Principle Violation: **Dependency Inversion Principle.**

Solution: Employer class depends on low level module classes ‘HourlyWorker’ and ‘SalaryWorker’, which violate the Dependency Inversion Principle.

To solve this issue, I created new interface which has method ‘calculatePay’. Both classes **‘**HourlyWorker’ and ‘SalaryWorker’ implements this method. So ‘Employer’ does not depend on low level module classes which solved the Dependency inversion principle violation.

5) **Question – 5**

Principle Violation: **Interface Segregation Principle.**

Solution: Interface ‘IInsect’ is general purpose interface, and class ‘AquaticInsect’ and ‘FlyingInsect’ both implement this interface which force both classes to implement some methods which are not used by that classes.

For example, ‘AquaticInsect’ class does not use or support ‘fly’ method same as in ‘FlyingInsect’ class swim method is not useful.

So, to solve this issue, I created other Two interface and declare methods which is related to AquaticInsect’ in ‘IAquaticInsect’’ interface and same for the FlyingInsect’ class (IFlyingInsect’).

Put common method in IInsect’ interface and divide other methods into other two interface, which solve the Interface Segregation principle.

6) **Question – 6**

Principle Violation: **<u>Dependency Inversion Principle and open Closed principle.</u>**

Solution: Class 'CountryGDPReport' depends on low level modules 'Canada' and 'Mexico', which violate the condition of the Dependency inversion principle.

'printCountryGDPReport' method violate the **Open Closed Principle,** because in future it is possible to add one more country in system for that method need to be updated. Even in Canada class it is also possible to add/ delete method according to new requirements (GetTorism) which need to be added in "printCountryGDPReport' which make this method open for modification which violate the open closed principle.

To solve the Dependency inversion principle, I created new interface 'ICountryList' which has two methods. 'Canada' and ' Mexico' both implements this methods.So now 'CountryGDPReport' does not depends on low level modules. Implements Two methods 'getCountryName' and 'getCountryGDPReportDetails' in both class 'Canada' and 'Mexico' which also solved the issue of open closed principle.

So, In future if other country class will be added or updated then no need to change in method 'printCountryGDPReport'.

7) **Question – 7**

Principle Violation: **Interface Segregation Principle.**

Solution: Interface IlibraryItem is general purpose interface, and class Book and DVD both implement this interface which force both classes to implement some methods which are not used by that classes.

For example: Book class is not using getPlayTime, isDigitalOnly, getCastList method in its class, so this method is not useful for Book class.

So, to solve this issue, I created other Two interface and declare methods which is related to book in 'IBookIteam' interface and same for the DVD class (IDigitalIteam).

Put common method in IlibraryItem interface and divide other methods into other two interface