**Assignment -2**

**Name: Sudarshan Suresh Srikant**
**Banner ID: B00808452**

**Question 1:**

1. The Customer class violates the Single Responsibility principle.

2. An object of the Customer class must have only Customer related information instead of having business logic code such as sending an email. Probably in the future, if a customer wants to send a letter through regular postal service, then the Customer class needs to be changed in order to implement that particular method. In order to avoid this, the emailCustomerSpecialOffer() can be written in another class. The Customer class should only be responsible for Customer related functionalities (such as setting an ID, name, etc).

3. The answer is present in sudarshan19943/QualityAssurance_Assignments/A2/Q1.

**Question 2:**

1. The USDollarAccount class violates the Liskov substitution principle.

2. The USDollarAccount class extends Account, however, it changes the way credit() and debit() functions work. This is a violation of the Liskov Substitution principle. In order to avoid this, we can implement separate debit() and credit() functions for different account types by inheriting the same functions from the parent class without altering the parent class credit() and debit() functions.

3. The answer is present in sudarshan19943/QualityAssurance_Assignments/A2/Q2.

**Question 3:**

1. The Student class violates the Single Responsibility principle.

2. The Student class must only provide Student related functionalities and must not care about how the student data is being saved or loaded from the database. To do this, we can implement the save() and load() functions in a separate function through a database related interface.

3. The answer is present in sudarshan19943/QualityAssurance_Assignments/A2/Q3.

**Question 4:**

1. The Employer class violates the Dependency Inversion Principle.

2. The Employer class need not create objects of every single worker type. Instead we can use an interface to implement the payment related methods in each worker type classes rather than calling them in the Employer class for each object.

3. The answer is present in sudarshan19943/QualityAssurance_Assignments/A2/Q4.

**Question 5:**

1. This code violates the Interface Segregation principle.

2. Not all classes implement all the methods of an interface. This would leave many classes with empty methods. In order to avoid this, we can create interfaces for different groups of classes to identify the functionality that is common to a particular class group.

3. The answer is present in sudarshan19943/QualityAssurance_Assignments/A2/Q5.

**Question 6:**

1. This code violates the Open/Closed principle.

2. The class CountryGDPReport must always create new objects when a new country is created. To avoid this modification every time, we can create an interface that would print the GDP values of different countries. Different countries can implement that interface and have their own implementation of the GDP calculation. The class CountryGDPReport would be closed for modification every time a new country is added.

3. The answer is present in sudarshan19943/QualityAssurance_Assignments/A2/Q6.

**Question 7:**

1. This code violates the Interface Segregation principle.

2. Books and items such as DVD's cannot implement the same set of functions. Hence separate interfaces for digital and non-digital items must be created. So, the Book class would implement only the functions related to a non-digital interface and the DVD class would implement methods related to a digital interface. A common interface would have Book and DVD to implement all the methods of that interface, which could result in empty method bodies.

3. The answer is present in sudarshan19943/QualityAssurance_Assignments/A2/Q7.