|  |
| --- |
| **SOLID Principles** |
| **Question 1 –**  Which S.O.L.I.D. principle does the Employer class violate?  **Answer –**  The Employer class violates Dependency Inversion Principle (DIP). In this example Employer class should depend on the abstraction and not on the concretion. In other words, Employer need not explicitly know which different categories of employees it employs. |
| **Question 2 –**  Which S.O.L.I.D. principle does the following code violate?  **Answer –**  Violates the Interface Segregation principle. It is better to have Client specific interfaces rather than to have a general purpose interface. Since we have two implementations Book and DVD, GetPlayTime, GetCastList should applicable to DVD client only. So, divide the ILibraryItem interface into two interfaces IBookLibraryItem and IDVDLibraryItem. |
| **Question 3 –**  Which S.O.L.I.D. principle does the ProfitReport class violate?  **Answer –**  ProfitReport class violates Single Responsibolity principle, as it states every class should have only one job to do - ProfitReport had methods like SendToPrinter() and SendToEmail(), which makes the class violate this principle, as this class should have functions only related to Report. |
| **Question 4 –**  Which S.O.L.I.D. principle does the following code violate?  **Answer –**  In this question, it violates Liskov Substitution Principle (LSP), LSP states that derived classes should never do less than their base class. In this example, the base class has three methods, and the derived class has 2 methods. A derived class must not only implement the base class’s methods and properties, but also conform to its implied behaviour. |
| **Question 5 –**  Which S.O.L.I.D. principle does the following code violate?  **Answer –**  This is violating the Open-Closed Principle (OCP), OCP tells us that classes should be open for extension but closed for modification. In the example, the class CountryGDPReport is not open for extension as it is restricted to only two countries i.e.., Canada and Mexico. I created a interface ICountry which has PrintCountryGDPReport() , all the countries will implemented this interface. The method PrintCountryGDPReport(List<ICountry> countries) in the class CountryGDPReport now will accept the List of Countries which will print all GDP report from the countries. In this way, the class CountryGDPReport is open for extension. |
| **Question 6 –**  Which S.O.L.I.D. principle does the PiggyBank class violate?  **Answer –**  PiggyBank class violates Single Responsibolity Principle (SRP), as it states every class should have only one job to do – PiggyBank has AddPenny(), AddDime(), AddNickel(),AddQuarter() method, all these methods related to an single purpose. It also has Load() and Save() methods in the same class, Load() reads the data from piggybank.txt file and builds the PiggyBank object and Save() writes the data to the test file piggybank.txt from the object PiggyBank, which violates the cohesive nature of the SRP principle. |
| **Question 7 –**  Which S.O.L.I.D. principle does the following code violate?  **Answer –**  Violates the Interface Segregation principle. It is better to have Client specific interfaces rather than to have a general-purpose interface. We have two Aquatic and Flying Insects, Swim() and MoveAntennae() are methods speciafically to AquaticInsects and Fly() and MoveAntennae() are methods speciafically to FlyingInsects. So, we divide the IInsectinterface into two interfaces IAquaticInsect and IFlyingInsect. |