Question 1

1. Which S.O.L.I.D. principle does the **Employer class** violate?
    - Dependency Inversion Principle.
2. Why does the code violate the principle?
    - Because High-level class Employer is depending on Hourly worker and Salary worker. If we want to add any other type of worker here we need to create another array list, This way it is violating the Dependency Inversion Principle. The solution is to create one Interface Iworker, and implement it in both hourly and salary worker class. Now in employee class we only need to create one array list object of worker. And in this array list we can add different types of employee.

Question 2

1. Which S.O.L.I.D. principle does the following code violate?
    - Interface Segregation Principle
2. Why does the code violate the principle?
    - DVD and Book both classes had methods which were giving null result, that means those methods were useless for that classes. So it was violation of Interface Segregation Principle. We can create two interface, each with required functionality of classes. This way Interface segregation principle can be resolved.

Question 3

1. Which S.O.L.I.D. principle does the **ProfitReport class** violate?
    - Single Responsibility Principle
2. Why does the code violate the principle?
    - In profitReport class, all methods are implemented like createReport, SendtoPrinter, which is giving more than one responsibility to class. So it is violating the Single responsibility principle. To solve this two different classes are created for createreport and send to printer. And then they method is called in profitReport class using objects.

Question 4

1. Which S.O.L.I.D. principle does the following code violate?
    - Single Responsibility Principle
2. Why does the code violate the principle?
   Here, Credit and Debit both are operation and they have no relation with bank operation so they should be implemented in separate class. This is how Single Responsibility Principle is violated. To resolve this, Credit and debit methods must be implemented in separate classes

Question 5

1. Which S.O.L.I.D. principle does the following code violate?
    - Dependency Inversion Principle
2. Why does the code violate the principle?
    - Here, printGDPReport is dependent on Canada and Mexico classes so Dependency Inversion Principle is violated. According to principle High level and low level classes must not be dependebale on each other they should dependable on Abstraction. So to solve this problem , I created one interface and in it I added abstract PrintcountryGDPReport method and this interface was implanted in Canada and Maxico, Using object of this interface all methods were called. This is how Both the classes are now dependable on Abstraction.

Question 6

1. Which S.O.L.I.D. principle does the PiggyBank class violate?
    - Single Responsibility Principle
2. Why does the code violate the principle?
    - Here, Load and Save methods are not related to Piggybank operation, so it should not included into piggybank class. This violates single Responsibility principle. To solve this Create one Class and add load and save method. And now create main class and create objects of Load and save. Form here we can call that methods. This way We can resolve the violation of Single Responsible Principle.

Question 7

1. Which S.O.L.I.D. principle does the following code violate?
    - Interface Segregation Principle
2. Why does the code violate the principle?
    - Here, both AquaticInsect class doesn't require fly functionalities, FlyingInsect class doesn't require swim functionalities. And here, Iinsect class had both properties that is useless, that is why I created two interface for each with having separate needed properties.