SOLID PRINCIPLES LAB BOOK

Document Revision History

Date	Revision No.	Author	Summary of Changes
23-Feb-2018	1	Sangeetha C	Content Creation

Objective	This Lab will help you understand 1. Applying SOLID principles to C# code 2. Refactor & Resolve the Errors in Code
Time	75 Mins

Task1. Create a class named Employee with the following members:

Properties:

- EmployeeId
- EmployeeName

Methods:

- Method Name : AddEmployee
  Return Type : bool
  Parameter Type : Employee
  Parameter Name: emp

- Method Name : GenerateReport
  Return Type : void
  Parameter Type : Employee
  Parameter Name : emp

Task2. Identify the problem with the above implemented class.
Hint: Look into the operations implemented by the Employee class.

Task3. Resolve the problem using Single Responsibility Principle (SRP) of SOLID
Hint: Move the GenerateReport method to a new class named ReportGeneration.

Task4. Identify the problem with the ReportGeneration class.
Hint: Report could be in any format.

Task5. Resolve the problem using Open Closed Principle (OCP) of SOLID
Hint: Create new classes for the different types of Report and inherit from ReportGeneration base class.

Task6. Add a two virtual methods in the Employee class.

Methods:

- Method Name : GetProjectDetails
  Return Type : string
  Parameter Type : int
  Parameter Name : employeeId

- Method Name : GetEmployeeDetails
  Return Type : string
  Parameter Type : int
  Parameter Name : employeeId

Task7. Implement two child class named PermanentEmployee and ContractEmployee, which inherit from Employee class and overrides both virtual methods.

Note: Contract Employee's data is not stored in db. So in GetEmployeeDetailsImplementation throw NotImplementedException.

Task8. In the Main method of the Current Application implement the below code.

```
List<Employee> employeeList = new List<Employee>();
employeeList.Add(new ContractEmployee());
employeeList.Add(new PermanentEmployee());
foreach (Employee e in employeeList)
{
e.GetEmployeeDetails(1245);
}
```

Task9. Run the Application and resolve the Exception.
Hint: Liskov substitution principle (LSP)

Task10. Create a new interface named IEmployeeOperations add the following methods.

Methods:

- Method Name : AddEmployee
  Return Type : bool
  Parameter Type : Employee
  Parameter Name : emp

- Method Name : SearchEmployee
  Return Type : Employee
  Parameter Type : int
  Parameter Name : employeeId

Task11. Let the Employee class inherit this interface. Resolve the error in the class.

Task12. Interface segregation principle (ISP) states that any client should not be forced to use an interface which is irrelevant to it. Resolve the ISP issue. Identify and Resolve the Issue.

Task13. WRT the code given below. Implement Dependency inversion principle (DIP)

```
public interface IMessenger
{
   void SendMessage();
}

public class Email : IMessenger
{
   public void SendMessage()
   {
      // code to send email
   }
}

public class SMS : IMessenger
{
   public void SendMessage()
   {
      // code to send SMS
   }
}

public class Notification
{
   private IMessenger _iMessenger;

   public Notification()
   {
```



```
        _ iMessenger = new Email();
    }

    public void DoNotify()
    {
        _ iMessenger.SendMessage();
    }
}
```

Implement all three types of Dependency inversion principle (DIP) on the Notification class.

