**C Sharp HOL Part I**

**---------------------------**

**Lab 1. Defining a Class with Constructor, Fields and Properties**

Write a program to store the details of the marks scored in various modules.

**Task 1:** Create a library project and define a class called Participant. Define the private members and public properties as follows: EmpId, Name, Company Name, FoundationMarks, WebBasicMarks, DotNetMarks, Total Marks, ObtainedMarks, Percentage Initialise the Total Marks to 300. ObtainedMarks and Percentage are calculated fields.

**Task 2:** Add 3 constructors, one Default, one parameterised to initialize the members and one static constructor to initialise the company Name to “Corporate Unniversity”

**Task 3:** Add the following functions to the class: a. To calculate Total Marks. b. To calculate the percentage. c. To return the percentage

**Task 4:** Create a console application to accept the data about participants, and create the object. The console application should call the appropriate functions to calculate the Total Marks and Percentage. And then Display the percentage.

**Task 5:** Modify the class created above to valide the marks. Write the validators in properties. The valid range is 0 to 100. If any invalid value is passed, then assign 0 to that module. Later, you can create Exception classes and raise the exception in case of Invalid data.

**Q1. You need to write a program to manage the Inventory of the used cars.**

**Task 1:** Create a simple text-based “Console Application” in C# to maintain a catalog of used cars. The catalog keeps track of each car's make, model, year, and sale price.

The program begins with an empty catalog. The program can perform the following operations:

* Adding a new car
* Modify the details of a particular car
* Search for a particular car in the Catalog
* List all the cars in the Catalog
* Delete a car from the Catalog
* Quit

If an unknown command is entered, the user should be informed and asked to enter another command.

**Hint:**

1) Create a class called as Car. Create appropriate constructors (Default and Parameterized), Properties for the Car class.

2) Use **Array** to store the Objects of a car.

**Lab 2: Implementing polymorphism using overriding in C#**

Q. 1. Implementing polymorphism using overriding in C#

|  |  |
| --- | --- |
| **Objective** | This lab will help you understand   1. Implementing Inheritance and achieving Polymorphism through overriding 2. Understand the c# keywords required to achieve it |
| **Time** | 90 Mins |

Note : Use C# 6.0 and 7.0 features, wherever applicable

* While printing the output use String Interpolation
* Use using static
* Use is and as operator

**Q1. You have created Employee class in Lab 1. You need to extend this class and create two derived classes from this class. The derived classes will be ContractEmployee and PermanentEmployee.**

The contract Employee class will have Perks as an additional property. The PermanentEmployee will have NoOfLeaves and ProvidendFund Properties.

**Task 1:** Create these two classes by inheriting from the Employee class.

**Task 2:** Override the GetSalary Method in these two classes. For Contract employee the new salary will be Salary + Perks. For Permanent Employee the new salary will be Salary – Providend Fund.

**Task 3:** Create a console application to use these classes. Create a Menu driven application to select the Type of employee. Based on the user selection create the object and accept the details from the user.

Also display the salary of the Employee.

**Task 4:** As we only need to create instance of Contract Employee and Permanent Employee classes, Convert the Employee class to Abstract class. Also make GetSalary method Abstract in the Base class.

**Q2. Mahesh has created the following code. The purpose is to create Circle and Triangle class by inheriting the Shape Class. Both the inherited classes should override the WhoamI() method of the Shape Class. The code has some bugs. Identify the Bugs and fix them.**

publicclassShape

**{**

privatevoid **WhoamI()**

**{**

Console.WriteLine("I m Shape"**);**

**}**

**}**

classTriangle **:** publicShape

**{**

publicvirtualvoid **WhoamI()**

**{**

Console.WriteLine("I m Triangle"**);**

**}**

**}**

publicclassCircle **:** publicShape

**{**

void **WhoamI()**

**{**

Console.WriteLine("I m Circle"**);**

**}**

**}**

classProgram

**{**

staticvoid **Main(**string**[] args)**

**{**

Shape **s;**

**s =** newTriangle**();**

**s.WhoamI();**

**s =** newCircle**();**

**s.WhoamI();**

Console.ReadKey();

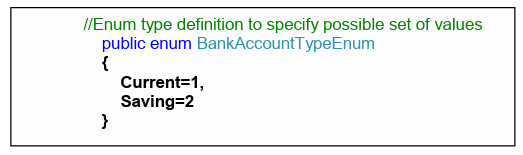
**}**

**}**

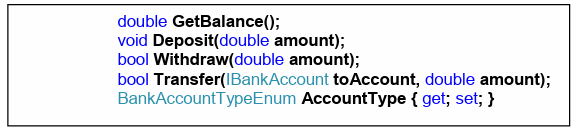
**Lab 3: Using Interface, Abstract and Concrete classes**

Q.1 Let’s build a sample banking program to perform the common tasks like Withdraw and Deposit.

**Task 1:** Create a class library project and Add a Class called BankAccount. This class needs to implement the IBankAccount Interface. Task 2: Define a enum as follows. This enum will be used as a property in the interface.



**Task 3:** Define IBankAccount interface and add the following fields to it.



**Task 4:** Create an abstract class called as BankAccount and implement the class with the interface defined above. Add a property called Balance in this class



**Task 5:** Implement only the Deposit method to increment the Balance. Keep the other two methods abstract in the class.

**Task 6:** Now let’s create concrete classes which are inherited from the BankAccount class.

//Concrete Bank Account Classes having their own rules for Minimum Balance

classICICI **// Inherit this from BankAccount**

**{**

Withdraw() // Override this method

{

// If Balance – amount is >= 0 then only WithDraw is possible.

// Write the code to achieve the same.

**}**

**Transfer()** //Override this method

**{**

// If Balance – Withdraw is >= 1000 then only transfer can take place.

// Write the code to achieve the same.

**}**

**}**

classHSBC **// Inherit this from BankAccount**

**{**

**Withdraw()** //Override this method

**{**

// If Balance – amount is >= 5000 then only WithDraw is possible.

// Write the code to achieve the same.

**}**

**Transfer()** //Override this method

**{**

// If Balance – Withdraw is >= 5000 then only transfer can take place.

// Write the code to achieve the same.

**}**

**}**

**Note**: GetBalance() and Deposit() code is shared by both classes via Base Abstract class.

**Task 7:** Create a console application. The Main() function needs to create the objects and execute the functionality as per the instructions.

**Main( ) //Write this function**

**{**

**// Task to be performed:**

Create a Object of ICICI

Set the Account type to Saving (Use enum)

Deposit Rs. 50000 to this account

Create another Object of ICICI

Set the Account type to Current (use enum)

Deposit Rs. 20000 to this account

Print the Balance of both these account objects.

Now call the Transfer function to transfer the money from Savings account to Current Account. The amount to be transferred is Rs. 5000.

e.g. a1.Transfer(a2,5000);

Now print the Balance after the Transfer from both the accounts.

Similarly, create two accounts of HSBC Bank. Transfer Rs. 30000 from Saving to Current and display the balance.

**}**

**Q2. There is a Change request from the customer. It is as follows:**

You need to calculate Interest paid by banks on Saving Account.

**Task 1 :** Add a function declaration “void CalculateInterest()” in the interface. Define the functions in the concrete classes such as ICICI accounts get 7% interest and HSBC gives 5% interest.