**Day 1**

**Section 1.1**

Create a Console Application with name "**Employee Management System**"

Add the variable with best suitable DataTypes to accept the following details from the user

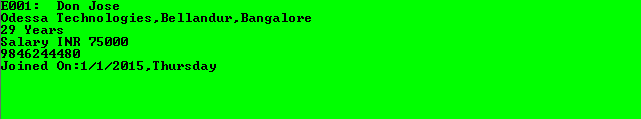
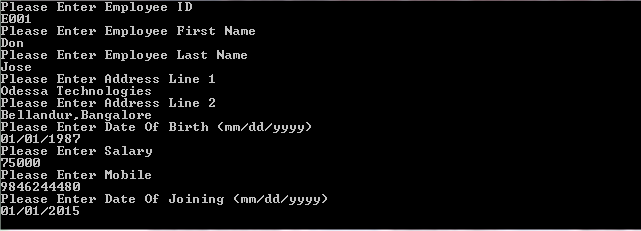
EmployeeId,FirstName,LastName,AddressLine1,AddressLine2,Dob,Age**(calculate from DOB)**,CTC, Mobile Number and Date of Joining .

**Notes:**

* Employee Id should be in this format **“E001”**, maximum characters allowed for First name and last name is 30, Mobile number should be eligible to accept the country code also.
* Use String.Format() and String.Concat() for manipulating with strings

**Section 1.2­**

Display the Employee details in the below format (color scheme also)



**Section 1.3**

Modify the above program in a way that after displaying the details it will ask the user “Do you want to close the screen immediately”, if yes the screen will close immediately or else will automatically close after 20 seconds

**Section 1.4**

Modify the above program by accepting all the values required for the Employee as a command line argument and display those details in the same format

**Section 1.5**

Modify the above program with the following changes

When you run the program it should show a **welcome message** and provide a menu to the user to select the operation they want to perform

1. Add new Employee details

2. View the Employee details

3. Change the screen settings

4. Exit

If the user selects **Menu option1** then screen should prompt to enter the details which mentioned in **section 1.1** along with the following validations/modifications with corresponding error messages in red color and all validations are mandatory and user can’t move further until he/she enters a proper value

* Maximum characters allowed for First Name and Last Name is 30 and both are mandatory fields
* Minimum 20 characters required for Address Line 1 and Address Line 2 with Address Line 1 as mandatory field
* Company accept only employees whose age is 18 years or above
* Should check for proper date by making sure date part is comes between 01 to 31 and month part is comes between 01 to 12 in order to avoid system to throw invalid format exception
* The above date format rule is applicable for Date of Joining also
* CTC should be minimum of 1Lakh(user will enter monthly salary only)
* Mobile number should be maximum 10 digits excluding country code, user should enter the mobile number in the following format, starting with “+” symbol followed by country code followed by a white space. *E.g. +91 9611671500*
* Add a new field to accept Employee’s preferred locations, use array to accompany this task.

Once the details entered by the user satisfy all the validations Menu should again display

If the user selects **Menu option2 (**User cannot select the Menu option 2 without adding the new employee details, appropriate error message should be displayed) all the entered details should be displayed according to the **section 1.2** with following additional information CTC instead of salary and the Preferred locations in bulleted format

After displaying the Employee details it should prompt the user to enter a choice to continue further for or exit the program. If the user wants to continue then the menu will be displayed and the operations will continue

If the user selects **Menu option3** which is to change the console styles it should display another menu

1. Change the back ground color

2. Change the font color

If the user selects any of the menu option **another menu should be displayed with the list of colors** which console can accept. Based on the user input you need to change either back ground/font color with the color chosen by the end user

If the user selects **Menu option4** the application should be closed in 2 seconds with displaying a thanks message

**Day 2**

**Section 2.1**

Add a class library project into the same solution with the name **“Employee Leave Management”.**

Create a public class Employee with the fields mentioned in **section 1.1** as private variables and add public properties for these fields.

All the validation that we added in the **section 1.5** will be added to this Employee class through the properties

In addition make the following modifications

* Modify the Employee ID property in such a way that it will be auto generated by appending a 3 digit random number generated concatenating with letter “E” and it should be read only.
* Add a read only field for Email id which will be auto generated by concatenating First Name with dot (.) and first letter of last name appended with **@odessatech.com**.

E.g. First Name: Don

Last Name: Jose then generated email id should be **Don.j@odessatech.com**

* Add a property to expose the auto generated password which can be changed by the end user.
* Create separate methods for Accepting the details and displaying the details with the names **AcceptEmployeeDetails** () and **DisplayEmployeeDetails** () respectively.
* Try to create separate methods for any additional operations performed here with proper return type, parameters and access specifiers to implement right abstraction and Encapsulation.

E.g. Create separate methods for Generating password, Generating Employee Id and Generating Email id

**Section 2.2**

Delete all the code snippets you have written in the Main method of “**Employee Management System**"

Console application, for the following changes.

Refer the class library project (**EmployeeLeaveManagement.dll**) here and the system should be modified in a way that this application is going to be used by “HR” team to manage the Employees in the company. So when you run the application it should ask the end user how many employees details you are going to add and based on that number you have to create array of Employee objects to perform the same task which you have already performed in **Section 1.5**

**Section 2.3**

We are developing this application to manage company’s day to day operations. So many places you may need to use company’s information like Name of the company, Address, Management details and other essential information regarding company. So you have to create a class which can store these details and there should be methods to access this information. Since we are developing this application for Odessa Technologies there is no requirement for multiple instances of company class, so create a class and corresponding methods to accomplish this task.

Modify the Main method you have developed in **Section 2.2** to add a menu option to **display company information**

**Section 2.4**

Company needs to store the details of the Clients which are very much light weighted information and they don’t want to make those Client objects to be garbage collected. Use the proper data structure to complete this task and you need to initialize those clients’ details through a constructor only.

We have all clients from US only so keep the Country code as Constant, whereas Zip code of the clients also not going to change but will come to know only at the time of creating the instance ,so use the proper access modifiers for the fields to perform the above requirements.

Write a method retrieve client information and modify the Main method you have developed in **Section 2.2** to add a menu option to **add and display client information**

**Day 3**

**Section 3.1**

As the part of the Employee Leave Management module you need to collect information like Employee Level(L1, L2, L3A, L3B, L4A, L4B, L4C, L5A, L5B, L5C, L6A, L6B, L6C, L7), Employee Role(Admin,AdminManager,Manager,Employee,HumanResource,Finance), Leave Status (Approve,Rejected,Cancelled,Pending) and Leave Type(Fullday,HalfDayFirstHalf,HalfDaySecondHalf).According to the requirement specified those fields should not take values other than mentioned. So implement proper mechanism so that invalid values can’t be collected for those fields.

**Section 3.2**

To work with external/ unmanaged resources like files, stream, database you have been asked to create a class named “External Resource” which is not going to be garbage collected. So implement proper mechanism in that class so that you are controlling the life of objects and the memory it consumes.

**Section 3.3**

To continue with employee leave management module you need to add one class library project which will contain the classes used for data transfer. Name the class library application as **Domain. Entity**; make sure you are adding all the projects in the same solution where the Console Application which you created in the **Section 1.1** is placed

Create the following classes with the given specification, since these classes are used for Data transfer, they will not have any operations associated but contains properties to store the corresponding details

|  |  |  |  |
| --- | --- | --- | --- |
| **Employee** | **Manager** | **User** | **Department** |
| EmployeeId(int)  FirstName(string)  LastName(string)  Email(string)  Mobile(string)  DateOfJoin(Date)  Gender(string)  IsActive(boolean)  DepartmentId(int)  ReportingTo(int)  PortfolioManagerId(int)  Role(EmployeeRole)(Enum Type)  EmployeeLevel(EmployeeLevel)(Enum Type) | ManagerId(int)  Name(string) | Id(int)  LoginName(String)  Password(String)  SecurityQuestion(String)  SecurityAnswer(String)  IsActive(boolean) | DepartmentId(int)  Name(string) |

|  |  |  |  |
| --- | --- | --- | --- |
| **EmployeePortfolio** | **LeaveApplication** | **LeaveAvailability** | **LeaveType** |
| EmployeeId(int)  ProjectId(int)  PortfolioId(int) | LeaveApplicationId(int)  EmployeeId(int)  LeaveTypeId(int)  StartDate(Date)  EndDate(Date)  AppliedDate(Date)  NoOfDays(decimal)  LeaveDay(LeaveType)(EnumType)  LeaveReson(String)  LeaveStatus(LeaveStatus)(EnumType)  AlternateContactId  ManagerId(List<int>)or Array | LeaveTypeId(int)  EmployeeId(int)  Carry(decimal)  Year(Date)  AvailedLeave(decimal)  AvailableLeave(decimal) | LeaveTypeId(int)  Name(string)  MaxNoOfDays(decimal)  ContinuesDays(decimal)  DaysInAdvance(decimals) |

|  |  |  |
| --- | --- | --- |
| **LeaveValidation** | **Portfolio** | **Project** |
| EmployeeId(int)  LeaveTypeId(int)  ContinuesDays(decimal)  DaysInAdvance(decimals)  AvailedLeave(decimal)  AvailableLeave(decimal) | PortfolioId(int)  ManagerId(int)  HrId(int)  IsActive(boolean) | ProjectId(int)  Name(int)  ManagerId(int)  PortfolioId(int)  IsActive(boolean) |

**Day 4**

**Section 4.1**

As per our requirements Employee Leave Management System for Odessa Technologies will have 5 types of users

**1. Employee**

All Employees of the Company (including employees in all other roles), who can avail the leave management services like Apply, Cancel, View Leave balances etc.

**2. HR**

HR will be getting a notification whenever any Employee’s comes under him/her uses the leave management services, apart from that all the HR related services will be also availed.

**3. Manager**

Manager can approve or rejects leave requests submitted by Employees coming under his/her team

**4. Admin**

Admin can register/update employee details to the system, add/edit new leave types and portfolios

**5. Admin Manager**

Admin manager can perform operations performed by both Admin and Manager

You need to create separate classes for each Roles with their operations. Add a new class Library project into our application with the name **Domain.DAL** (Data Access Layer) where you will be performing all the operations each roles can perform. Create the above classes as separate files in the Domain.DAL class library project with proper class hierarchy (inheritance)

**Section 4.2**

As part of requirement gathering we have found out the following functionalities which need to be implemented in the Employee Leave Management System.

|  |  |  |
| --- | --- | --- |
| **User Functionalities(Authentication)**   * Login * Registration * Forgot Password * Update Password * Update Security Question | **Leave Functionalities**   * Validating Leaves * Update Leave Balance * Cancel Leave * View Leave History(different ways) * View Current Status * View Leave Availability * Getting Manager’s details | **Admin Functionalities**   * Adding/Editing Employees * Adding/Editing Leave Types * Adding Portfolios * Updating Holiday Calendar * Adjusting Leave Balance |
| **Manager Functionalities**   * Viewing leave requests * Approving/Rejecting requests | **Portfolio Functionalities**   * Add/Edit Employee Portfolio details | **Project Functionalities**   * Add/Update Project details |

After examining the functionalities you must have understood that some functionalities are common across roles and some functionalities are implemented differently based on the roles.

As an object oriented developer you are asked to implement proper design approach to meet the above requirements

**Hint**. Create a Separate Class Library project **Domain.IDAL** with separate interface declaration based on the functionalities with proper inheritance then implement these interfaces to the Role classes which we created in **Domain.DAL** class library based on the functionalities each Role performs

**Section 4.3**

To start working with Employee Management Application we need some sample data. In the console application which we created in **Section 1.1** **Employee Management System** create a static class with the name **DataSource** with a method **PopulateData.** Create generic collections to store information for all the classes which we created in **Domain.Entity** and populate these classes with some dummy data in **PopulateData** method and call the same when the application runs.

**Day 5**

**Section 5.1**

Go back to **Domin.DAL** which you have create in **Section 4.1/4.2,**there you just created skeletons of methods for all classes now you are going to implement functionality to these classes.

**Note:**

* All your code should be adhere to the coding standards and object oriented design principles
* Proper error handling should be implemented

Your 5 classes in the **Domain.DAL** structure follows

**    **

**All the operation that you are going to perform will be based on the collection classes and data you filled in the Section 4.3**

Modify the Main method of Employee Management System in the following manner. As per the requirement mention in **Section 4.3** whenever the application runs the collection classes need to be populated. After successfully populating data, end user should get a Menu with a welcome message and to choose the type of the role they belongs (Employee, Admin, Manager, HR, Admin Manager) and also an option if the user is first time visitor to the system. If the User selects the last option (New user) then **RegisterUser()** method of the Employee class need to be called with all logics required to insert a new Employee to the User collection.

**Section 5.1.1**

After choosing the role user will be prompted to prove the identity by providing the Id and Password.

You have to write the code to perform the login check in the Login () method of Employee class.

**Note.** User credentials should be validated with the **User** collection class’s dummy data **using generic delegates** with proper validation messages (E.g. Message should be prompted if the user failed to enter any one of the credentials, if the credentials doesn’t match proper error message has to be displayed)

On successful authentication user will be displayed with Menu based on the Role

**Section 5.2**

If the User chooses the **Employee** role then the following Menu will be displayed

1. Apply Leave

2. View Leave Application Status

3. Cancel Leave Application

4. View Leave History

5. View Leave Availability

6. Logout

Now you have to implement the corresponding methods in the Employee class to perform the above operations.

**Note:** You can refer the **Employee.cs** given in the **Section 5.1** to understand the signature of the corresponding methods

**Section 5.3**

If the User chooses the **Manager** role then the following Menu will be displayed along with the options available for Employees

1. View Leave Requests
2. Approval Status

Now you have to implement the corresponding methods in the Manager class to perform the above operations.

**Note:** You can refer the **Manager.cs** given in the **Section 5.1** to understand the signature of the corresponding methods. Manager can perform Approval/Rejecting or viewing requests only those who are reporting to him/her

**Section 5.4**

If the User chooses the **Admin** role then the following Menu will be displayed along with the options available for Employees

1. Add New Employee
2. View All Employees
3. Add New Leave Type
4. View All Leave Types
5. Add New Portfolio
6. Add New Project

Now you have to implement the corresponding methods in the Manager class to perform the above operations.

**Note:** You can refer the **Admin.cs** given in the **Section 5.1** to understand the signature of the corresponding methods.

**Section 5.5**

If the User chooses the **AdminManager** role then the Menu will be displayed will be having all the options available for Manager and Admin along with the options available for Employees

**Day 6**

**Section 6.1**

You have completed the Employee Leave Management Module, but now management of Odessa Technologies want few changes to be implemented.

After the successful login user’s should to get a personalized greeting message with the Name and greeting message based on the time of Login. The Message format given below

****

So the greetings will change based on the time of Login. You need to perform these operation using **delegates and events** and this message should trigger automatically**.**

**Section 6.2**

As per the requirements, applying for a leave is permitted within the office timings only (9-6).If any Employee login to the Employee Leave Management System other than the above specified time there should be an alert message displayed to user informing the same and applying for leave should not be carried out, but all other operations should function normally

**Note:** You should use **events and delegates** to perform the above operations

**Section 6.3**

You have to modify the Employee leave application module in the following manner.

When the user applying for leave there need to perform some basic validation like, start date should be less than end date, you can’t apply leave for the date more than 1 month advance etc.

When you check for these basic validation you need to throw a **user defined exception** if the start date is greater than end date.

**Section 6.4**

In to the main menu add one option “**View Operations**” and if the user selects this option it should prompt the user to enter the role for which you need to view the operation. Now based on the role selected by the user you should display all the methods/operations that role can perform

**Note:** Use **refection** techniques to carry out this requirement