

Encapsulation Assignment

By

Narala Praveen

28-jan-2022

Amazon Employee Class:

Program: `class Employee`

```
{
    private int Empid;
    private string Name;
    private string Email;
    private string Designation;
    private int Phone;
    public void AddEmployedata()
    {
        //To do
    }
    public void PrintEmployeeData()
    {
        //To do
    }
    public void EditEmployeeData()
    {
        //To do
    }
    public void DeleteEmployeeData()
    {
        //To do
    }
}
```

UML Diagram:

Class Employee

- Name: String
- EmpID: String
- Email: String
- Phoneno: Int
- Designation: String

+AddEmployedata():void
+PrintEmployedata():void
+EditEmployedata():void
+DeleteEmployedata():void

Amazon Customer Class:

Program class: `class Customer`

```
{
    private int CustomerID;
    private string Name;
    private string Email;
    private string Address;
    private int Phoneno;
    public void AddCustomerdata()
    {
        //To do
    }
    public void PrintCustomerData()
    {
        //To do
    }
    public void EditCustomerData()
    {
        //To do
    }
    public void DeleteCustomerData()
    {
        //To do
    }
}
```

UML Diagram:

Class Customer

- Name: String
- CustomerID: String
- Email: String
- Phoneno: Int
- Address: String

+AddCustomerdata():void
+PrintCustomerdata():void
+EditCustomerdata():void
+DeleteCustomerdata():void

Amazon Product Class Program:

Program: `class Product`

```
{  
    private int price;  
    private string Name;  
    private string Colour;  
    private string Brand;  
    private string IMEI;  
    public void Adddata()  
    {  
        //To do  
    }  
    public void PrintData()  
    {  
        //To do  
    }  
    public void EditData()  
    {  
        //To do  
    }  
    public void DeleteData()  
    {  
        //To do  
    }  
}
```

UML Diagram:

Class Product

- Name: String
- Colour: String
- Price: Int
- IMEI: Int
- Brand: String

+Addproductdata():void
+PrintProductdata():void
+EditProductdata():void
+DeleteProductdata():void

Amazon Class Order:

Program: `class Order`

```
{
    private int orderID;
    private int Price;
    private string TrackID;
    private string Specifications;
    private string DeliveryDate;
    public void Addorderdate()
    {
        //To do
    }
    public void PrintOrderData()
    {
        //To do
    }
    public void EditOrderData()
    {
        //To do
    }
    public void DeleteOrderData()
    {
        //To do
    }
}
```

UML Daigram:

Class Order

- TrackID: String
- Specifications: String
- OrderID: Int
- Price: Int
- Deliverdate: String

+AddOrderdata():void
+PrintOrderdata():void
+EditOrderdata():void
+DeleteOrderdata():void

Amazon Class Seller:

Program: `class Seller`

```
{  
    private string SellerId;  
    private int Phoneno;  
    private string name;  
    private string Adress;  
    private string Product;  
    public void Adddata()  
    {  
        //To do  
    }  
    public void PrintData()  
    {  
        //To do  
    }  
    public void EditData()  
    {  
        //To do  
    }  
    public void DeleteData()  
    {  
        //To do  
    }  
}
```

UMD Daigram:

Class Seller

- Phoneno: String
- Name: String
- SellerID: Int
- Address: Int
- Product: String

- +AddSellerdata():void
- +PrintSellerdata():void
- +EditSellerdata():void
- +DeleteSellerdata():void

Apollo class Doctors:

Program: `class Doctors`

```
{
    private string name;
    private int Phoneno;
    private string Email;
    private string Patientcount;
    private string Specialization;
    public void AddDoctorsdata()
    {
        //To do
    }
    public void PrintDoctorsData()
    {
        //To do
    }
    public void EditDoctorsData()
    {
        //To do
    }
    public void DeleteDoctorsData()
    {
        //To do
    }
}
```

UML Daigram:

Class Doctors

- Phoneno: String
- Name: String
- Email: String
- Specialization:String
- Patientcount:Int

+AddDoctorsdata():void
+PrintDoctorsdata():void
+EditDoctorsdata():void
+DeleteDoctorsdata():void

Apollo Class Patient:

Program: `class Patients`

```
{
    private string name;
    private int Phoneno;
    private string Email;
    private string Disease;
    private string Gender;
    public void Addpatientdata()
    {
        //To do
    }
    public void PrintPatientData()
    {
        //To do
    }
    public void EditPatientData()
    {
        //To do
    }
    public void DeletePatientData()
    {
        //To do
    }
}
```

UML Diagram:

Class Patient

- Phoneno: Int
- Name: String
- Email: String
- Diseases:String
- Gender:String

+AddPatientdata():void
+PrintPatientdata():void
+EditPatientdata():void
+DeletePatientdata():void

Apollo Non-Medical Staff:

Program: `class Nonmedicalstaff`

```
{
    private string name;
    private int Phoneno;
    private string Email;
    private string Designation;
    private string Address;
    public void Addnonmedicalstaffdata()
    {
        //To do
    }
    public void PrintnonmedicalstaffData()
    {
        //To do
    }
    public void EditnonmedicalData()
    {
        //To do
    }
    public void DeletenonmedicalstaffData()
    {
        //To do
    }
}
```

UML Diagram:

Class nonmedicalstaff

- Phoneno: Int
- Name: String
- Email: String
- Designation:String
- Address:String

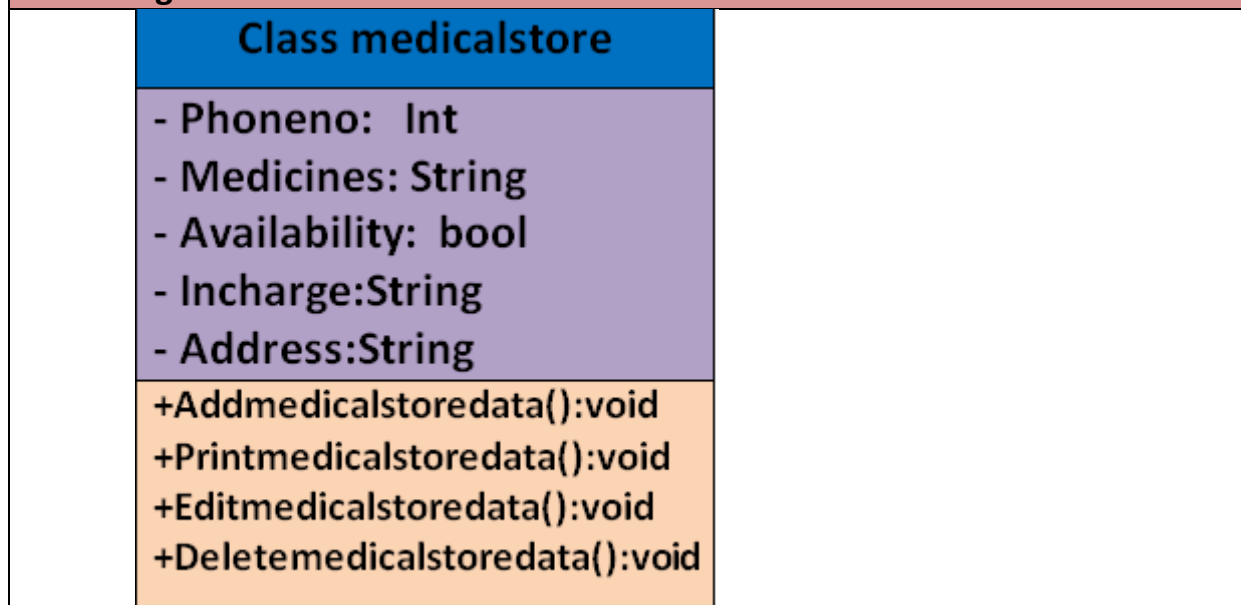
- +Addnonmedicalstaffdata():void
- +Printnonmedicalstaffdata():void
- +Editnonmedicalstaffdata():void
- +Deletenonmedicalstaffdata():void

Apollo Class Medical store:

Program: `class medicalstore`

```
{
    private string medicines;
    private int Phoneno;
    private bool Availability;
    private string incharge;
    private string Address;
    public void Addmedicalstoredata()
    {
        //To do
    }
    public void PrintmedicalstoreData()
    {
        //To do
    }
    public void EditmedicalstoreData()
    {
        //To do
    }
    public void DeletemedicalstoreData()
    {
        //To do
    }
}
```

UML Diagram:



Apollo Class Diagnostics:

Program: `class Diagnostics`

```
{  
    private string type;  
    private int testsno;  
    private string Equipment;  
    private string incharge;  
    private string location;  
    public void AddDiagnosticsdata()  
    {  
        //To do  
    }  
    public void PrintDiagnosticsData()  
    {  
        //To do  
    }  
    public void EditDiagnosticsData()  
    {  
        //To do  
    }  
    public void DeleteDiagnosticsData()  
    {  
        //To do  
    }  
}
```

UML Diagram:

Class Diagnostics

- Testno: Int
- Incharge: String
- Type: String
- Equipment:String
- Location:String

+AddDiagnosticsdata():void
+PrintDiagnosticsdata():void
+EditDiagnosticsdata():void
+DeleteDiagnosticsdata():void

Police station Class

Program: `class Station`

```
{
    private string name;
    private int stationno;
    private string type;
    private string Address;
    private string Commissionarate;
    private string zone;
    public void AddSationdata()
    {
        //To do
    }
    public void PrintSationData()
    {
        //To do
    }
    public void EditstaionData()
    {
        //To do
    }
    public void DeleteStationData()
    {
        //To do
    }
}
```

UML Diagram:

Class Station

- Name: String
- Stationno: Int
- Type: String
-Address:String
- Zone:String

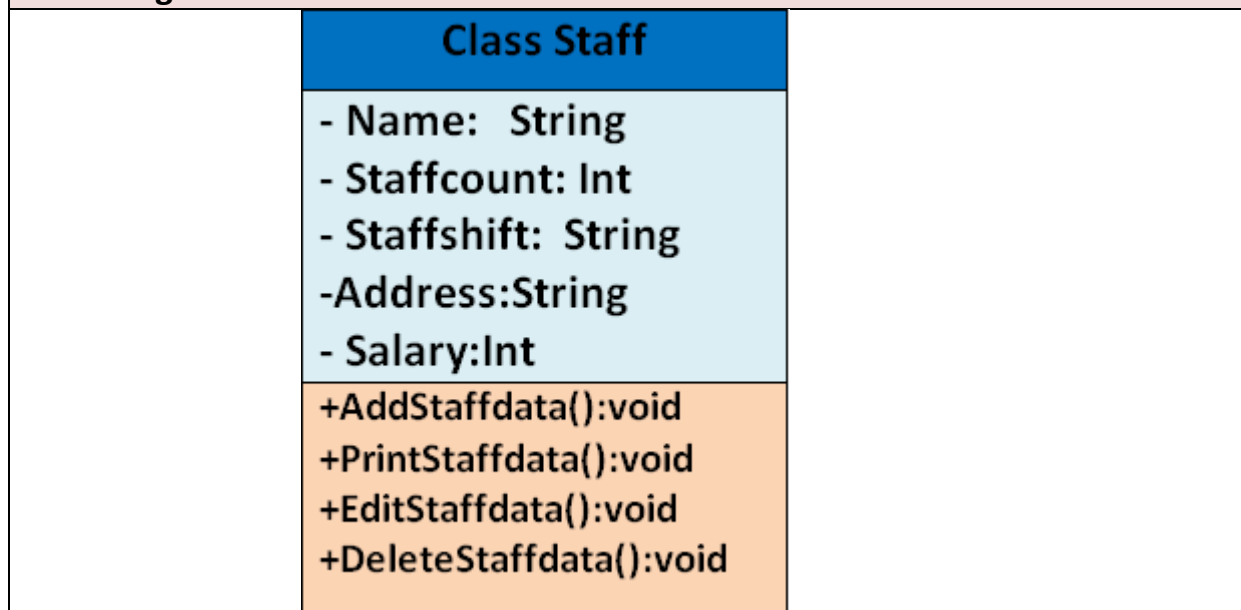
+AddStationdata():void
+PrintStationdata():void
+EditStationdata():void
+DeleteStationdata():void

Police Station Class Staff:

Program: class Staff

```
{
    private string name;
    private int staffcount;
    private string staffshift;
    private string Address;
    private string Designation;
    private int salary;
    public void AddStaffdata()
    {
        //To do
    }
    public void PrintStaffData()
    {
        //To do
    }
    public void EditStaffData()
    {
        //To do
    }
    public void DeleteStaffData()
    {
        //To do
    }
}
```

UML Daigram:



Police Station Class case:

Program: `class Case`

```
{
    private string name;
    private int caseno;
    private string casetype;
    private string caseassigned;
    private string casestatus;

    public void AddCasedata()
    {
        //To do
    }
    public void PrintCaseData()
    {
        //To do
    }
    public void EditCaseData()
    {
        //To do
    }
    public void DeleteCaseData()
    {
        //To do
    }
}
```

UML Diagram:

Class Case

- Name: String
- Caseno: Int
- Casetype: String
- Caseassigned:String
- Casestatus:String

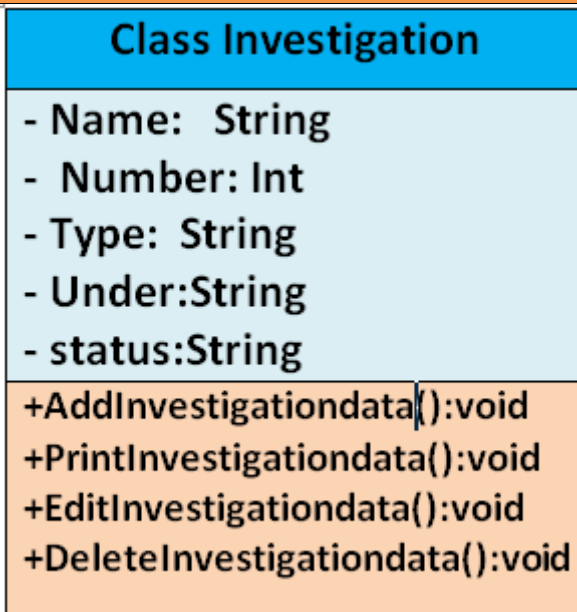
- +AddCasedata():void
- +PrintCasedata():void
- +EditCasedata():void
- +DeleteCasedata():void

Police Station Class Investigation:

Program: `class Investigation`

```
{  
    private string name;  
    private int number;  
    private string type;  
    private string under;  
    private string status;  
  
    public void AddInvestigationdata()  
    {  
        //To do  
    }  
    public void PrintInvestigationData()  
    {  
        //To do  
    }  
    public void EditInvestigationData()  
    {  
        //To do  
    }  
    public void DeleteInvestigationData()  
    {  
        //To do  
    }  
}
```

UML Diagram:



Police Station Class accused:

Program: `class Accused`

```
{
    private string name;
    private int number;
    private string type;
    private int section;
    private string status;

    public void AddAccuseddata()
    {
        //To do
    }
    public void PrintAccusedData()
    {
        //To do
    }
    public void EditAccusedData()
    {
        //To do
    }
    public void DeleteAccusedData()
    {
        //To do
    }
}
```

UML Diagram:

