Notepad Content:

Day 2:
C# Programming:

-> Classes, Objects and Methods
-> Constructors
-> Static keyword
-> Encapsulation

-> Constructors
Product p = new Product();
LHS Reference = RHS Object

-> Allocate memory to the object
-> Initialize the member variables
-> three kinds: Default constructor, Parameterless constructor, Parameterized constructor

-> Short cut for creating parameterless constructor => ctor + TAB

-> Static keyword
job of static ctor -> to initialize static variables of the class
static variables are shared by all objects of the class -> common value.
static members belong to the class itself, not to any object.

static ctor is called only when the first object is created for a class.

current value of counter - initial value of counter = number of objects created

when a method works only with static variables of the class, the method then belongs to the whole class, and not to any one object. Hence this method must also be marked as static.
Static methods must be called with the class name, not the object name.

In a non-static method, you can use static variables and instance variables.

In a static method, you can only use static variables, but not instance variables.

-> Encapsulation

-> Is done using Properties

-> Abstraction means hiding the data which you NEED NOT know.

-> Encapsulation means hiding the data which you CANNOT know.

-> Properties:

1. Create a private variable
2. Create a public property with the same name as the variable, but in pascal case.

```
private int age;

public int GetAge()
{
	return this.age;
}

public void SetAge(int age)
{
	this.age = age;
}
```

If you want validation, go for ordinary property with get set blocks.

If you dont want any validation, go for auto-implemented properties.

Shortcut to create auto implemented properties -> prop + TAB

1. Is it compulsory to have both get and set blocks?

-> In ordinary properties

-> In auto-implemented properties

2. Can get/set block be private? -> Yes

3. Can both get and set be private for a property?

Assignments:
1. Classes, Methods, Constructors
2. Encapsulation using Properties

Visual Studio Demos:

Product.cs:

```
namespace A2ZSalesBusinessLayer
{
   public class Product
   {
      public int productId;
      public string productName;
      public string productDescription;
      public double price;
      public string category;
      public int quantityAvailable;

      public static int counter;  //static variable

      //static ctor
      static Product()
      {
         Console.WriteLine("Static ctor");
         counter = 101;
      }

      public Product(string productName, string description, double price,
```

```
        string category, int qtyAvailable)
    {
        Console.WriteLine("Instance ctor");
        this.productName = productName;
        this.productDescription = description;
        this.price = price;
        this.category = category;
        this.quantityAvailable = qtyAvailable;
        //Autogenerate product Id with the help of counter
        this.productId = counter++;
    }

    public static int CountOfProducts()
    {
        //this.productId = -1;

        int count = counter - 101;
        return count;
    }

    public Product()
    {
        Console.WriteLine("Parameterless ctor");
        productId = -1;
        productName = "";
        productDescription = "N.A.";
        category = "Out of stock";
        quantityAvailable = 0;
        price = 0;
    }

    public Product(int productId, string productName, string description,
double price,
        string category, int qtyAvailable)
    {
        this.productId = productId;
        this.productName = productName;
        productDescription = description;
```

```
            this.price = price;
            this.category = category;
            quantityAvailable = qtyAvailable;
        }

        public Product(int productId, string productName, string description,
double price,
            string category)
        {
            this.productId = productId;
            this.productName = productName;
            productDescription = description;
            this.price = price;
            this.category = category;
            //quantityAvailable = 10;
        }



        public string DisplayDetails()
        {
            string details = "";
            details = productId + " " + productName + " " +
productDescription;
            return details;
        }

        public void UpdateProductStock(int addOnQuantity)
        {
            quantityAvailable = quantityAvailable + addOnQuantity;
        }


    }
}
```

Customer.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace A2ZSalesBusinessLayer
{
    public class Customer
    {
        //public int customerId;
        //public string name;
        //public long contactNumber;      //string because no calculations
needed
        //public string email;
        //public bool isPrivilegedCustomer;
        //public int age;

        public int CustomerId { get; private set; }
        public string Name { get; set; }
        public long ContactNumber { get; set; }
        public string Email { get; set; }
        public bool IsPrivilegedCustomer { get; set; }

        private int age;

        public int Age
        {
            get
            {
                return this.age;
            }
            set
            {
                if(value > 0)
                {
                    this.age = value;
```

```
            }
            else
            {
                this.age = 0;
                Console.WriteLine("Age is set to default value 0");
            }
        }
    }

    public Customer(int custId, string name, long contact, string email,
        bool isPriv, int age)
    {
        this.customerId = custId;
        this.name = name;
        this.contactNumber = contact;
        this.email = email;
        this.isPrivilegedCustomer = isPriv;
        this.age = age;
    }
  }
}
```

Program.cs

```
using A2ZSalesBusinessLayer;

namespace A2ZSalesConsoleApp
{
  internal class Program
  {
    static void Main(string[] args)
    {
      #region Default Constructor

      //Product p = new Product();

      //Console.WriteLine("Product Id : " + p.productId);
```

```
        //Console.WriteLine("Product Name : " + p.productName);
        //Console.WriteLine("Product Desc : " + p.productDescription);
        //Console.WriteLine("Product Category : " + p.category);
        //Console.WriteLine("Product Price : " + p.price);
        //Console.WriteLine("Product Quantity : " + p.quantityAvailable);

        #endregion

        #region Parameterless Constructor

        //Product p = new Product();

        //Console.WriteLine("Product Id : " + p.productId);
        //Console.WriteLine("Product Name : " + p.productName);
        //Console.WriteLine("Product Desc : " + p.productDescription);
        //Console.WriteLine("Product Category : " + p.category);
        //Console.WriteLine("Product Price : " + p.price);
        //Console.WriteLine("Product Quantity : " + p.quantityAvailable);

        #endregion

        #region Parameterized Constructor

        //Product p = new Product(1, "Death on the Nile", "Murder Mystery
by Agatha Christie",
        //    299, "Book", 25);

        //Console.WriteLine("Product Id : " + p.productId);
        //Console.WriteLine("Product Name : " + p.productName);
        //Console.WriteLine("Product Desc : " + p.productDescription);
        //Console.WriteLine("Product Category : " + p.category);
        //Console.WriteLine("Product Price : " + p.price);
        //Console.WriteLine("Product Quantity : " + p.quantityAvailable);

        //Product p2 = new Product(1, "Death on the Nile", "Murder Mystery
by Agatha Christie",
        //    299, "Book");
```

```
        //Console.WriteLine("Product Id : " + p2.productId);
        //Console.WriteLine("Product Name : " + p2.productName);
        //Console.WriteLine("Product Desc : " + p2.productDescription);
        //Console.WriteLine("Product Category : " + p2.category);
        //Console.WriteLine("Product Price : " + p2.price);
        //Console.WriteLine("Product Quantity : " + p2.quantityAvailable);

        #endregion

        #region Static keyword

        ////Create 3 products
        //Product p1 = new Product("Death on the Nile", "Book on murder
mystery", 299,
        //    "Book", 25);
        //Product p2 = new Product("The Big Four", "Book on murder
mystery", 399,
        //    "Book", 15);
        //Product p3 = new Product("Harry Potter and the prisoner of
Azkaban",
        //    "Book on fantasy", 499,
        //    "Book", 20);

        //Console.WriteLine("Product ID of P1 : " + p1.productId);
        //Console.WriteLine("Product ID of P2 : " + p2.productId);
        //Console.WriteLine("Product ID of P3 : " + p3.productId);

        //int c = Product.CountOfProducts();

        //Console.WriteLine("Count = " + c);

        ////Product p;


        #endregion

        #region Encapsulation
```

```
            //Customer customer = new Customer(1, "John", 9999999999,
"john@gmail.com", true, -1);

            ////customer.age = 10;
            //customer.Age = -10;
            ////customer.CustomerId = 1;      //private set
            //customer.ContactNumber = 9999999999;
            //customer.Name = "James";
            //customer.IsPrivilegedCustomer = true;

            //Console.WriteLine("Customer ID : " + customer.CustomerId);
            //Console.WriteLine("Customer Name : " + customer.Name);
            //Console.WriteLine("Customer Contact : " +
customer.ContactNumber);
            //Console.WriteLine("Customer Email : " + customer.Email);
            //Console.WriteLine("Customer Privilege : " +
customer.IsPrivilegedCustomer);
            ////Console.WriteLine("Customer Age : " + customer.age);
            //Console.WriteLine("Customer Age : " + customer.Age);

            #endregion
        }
    }
}
```

Assignments:

Assignment 1:

Springboard:
https://infyspringboard.onwingspan.com/web/en/viewer/web-module/lex_3304360677113107300 0_shared?collectionId=lex_33493780988125323000_s

hared&collectionType=Course&pathId=lex_auth_012923704538013696134 1_shared

Assignment 2: Healthcare Management System

Requirements:

Create a healthcare system **GetWellApp** that manages doctors and patients using the following concepts:

- Classes (`Doctor`, `Patient`, and `Hospital`)
- Instance and Static Constructors
- Static Variables and Methods
- Properties (Auto-implemented and Full properties)

1. Class: `Hospital` (Static Class)

- **Static Field:** `TotalHospitals` (int) – Keeps track of the number of hospitals created.
- **Static Constructor:** Initializes `TotalHospitals`.
- **Static Method:** `DisplayTotalHospitals()` – Prints the total number of hospitals.

2. Class: `Doctor`

- **Fields (Private with Properties):**
  - `DoctorID` (int, read-only)
  - `Name` (string)
  - `Specialization` (string)
  - `HospitalName` (string)
- **Constructor:** Assigns values to fields.
- **Method:** `DisplayDoctorInfo()` – Prints the doctor's details.
- **Static Field:** `TotalDoctors` (int) – Tracks the number of doctors.
- **Static Constructor:** Initializes `TotalDoctors`.

3. Class: `Patient`

- **Properties (Auto-implemented):**
  - `PatientID` (int, read-only)
  - `Name` (string)
  - `Disease` (string)
  - `Age` (int)
- **Constructor:** Assigns values to properties.
- **Method:** `DisplayPatientInfo()` – Prints patient details.

Instructions:

1. Create at least **two doctors** and **two patients** in the `Main` method.
2. Call the methods to display details of doctors and patients.
3. Display the total number of doctors and hospitals.

Assignment 3: Banking System

Requirements:

Develop a banking system that manages accounts using:

- Classes (Bank, BankAccount)
- Static Constructors, Static Variables
- Properties (Auto-implemented and Full properties)
- Methods

1. Class: Bank (Static Class)

- **Static Field:** TotalBanks (int) – Tracks the total number of banks created.
- **Static Constructor:** Initializes TotalBanks.
- **Static Method:** DisplayTotalBanks() – Prints the total number of banks.

2. Class: BankAccount

- **Fields (Private with Properties):**
  - AccountNumber (int, read-only)

    - AccountHolderName (string)
    - Balance (decimal, with validation)
- **Constructor:** Assigns values to fields.
- **Methods:**
    - Deposit(decimal amount) – Adds amount to Balance.
    - Withdraw(decimal amount) – Deducts amount if funds are available.
    - DisplayAccountDetails() – Prints account details.
- **Static Field:** TotalAccounts (int) – Tracks the total number of accounts.
- **Static Constructor:** Initializes TotalAccounts.

Instructions:

1. Create at least **two bank accounts** in the Main method.
2. Perform deposit and withdrawal operations.
3. Display the account details and total number of accounts and banks.