PT- Tutorial 1

Date
Page

Bishal Poudel
191318

1. How could you differentiate between structural and object oriented programming. Explain the main features of object oriented programming (OOP)?

⇒

SOP (structural)	OOP (object)
→ Function oriented with step-by-step interaction between functions.	→ object oriented with step-by-step interaction between objects.
→ Function is given more importance.	→ Data is given more importance.
→ suitable for simple program.	→ suitable for complex program.
→ Less data security	→ more data security.
→ code reusability is not possible.	→ code reusability
→ Less abstraction	→ more abstractions.
→ Follow Top-down approach	→ Follow Button-up approach.

Features of OOP:-

→ object : Entities of a real world.
→ class : Blueprint of object.
→ Abstraction : Representing essential features by hiding the background details.
→ Encapsulation and data hiding : Wrapping of data and function in a single unit.

1/1

→ Inheritance : process of deriving char. of one class by another class.

→ polymorphism : ability to take one than more form.

→ Message passing : communication between objects.

→ Dynamic binding : Runtime linkage of function call with function definition.

2. Write about event oriented programming and subject oriented programming. Write the advantage of using IDE for software development.

⇒ Event oriented programming (EOP) :

It means that the program everything as a response to an event, instead of a top down program where the ordering of the code determines the order of execution. Program code based on what happens when the user does something. Events can be mouse move, clicks etc.
EOL implies that an application waits for an event to occur before taking any action.

subject oriented programming (SOP) :

It is an enhancement of OOP that allows decentralized class definition. An application developer who needs new operation associated with classes can implement them, not by editing existing code for the classes, but as a separate collection of class definitions called a subject.

Date
Page

Advantages of using IDE:-

- IDE (integrated development environment) is an software that provides comprehensive facilities to computer programmer for software development.
- provides interfaces for users to write code, organize text groups and automate programming redundancies.
- Syntax highlighting and autocompleting.
- It also provides bed debuggers as well as compiler for specific program.
- Easy to write, execute and debug code.

3. What is aspect oriented programming? How aspect oriented programming solve the limitation of object oriented programming? Explain each core terminologies of AOP.

⇒ AOP is a new technology for separating crosscutting concerns into single units called ~~step~~ aspects. An aspect is a modular unit of crosscutting implementation. It encapsulates behaviours that affect multiple classes into reusable modules.

AOP compliments OOPs in the sense that it also provide modularity and key unit of modularity is aspect ~~with~~ than class. AOP breaks the program logic into distinct parts called concerns. It is used to increase modularity by cross-cutting concerns.

A cross-cutting concern is a concern that can affect the whole application and should be centralized in one code location in code as possible, such as transaction management, authentication, logging, security etc..
It provides dynamic in static OOP.

Terminology:-

(a) Join point

→ It is any point in program such as method execution, exception handling, file access etc.

(b) Pointcut

→ It is an expression language of AOP that matches join point. It allows us to specify join point

(c) Advice

→ It represents an action taken by an aspect at a particular join point.

→ way of expressing a cross-cutting action that needs to occur.

(d) Mixin

→ An instance of a class to be mixed in with the target instance of a class to introduce new behaviour.

(e) Weaving

→ process of combination of core codes with aspects to the final system.

(f) Tangling

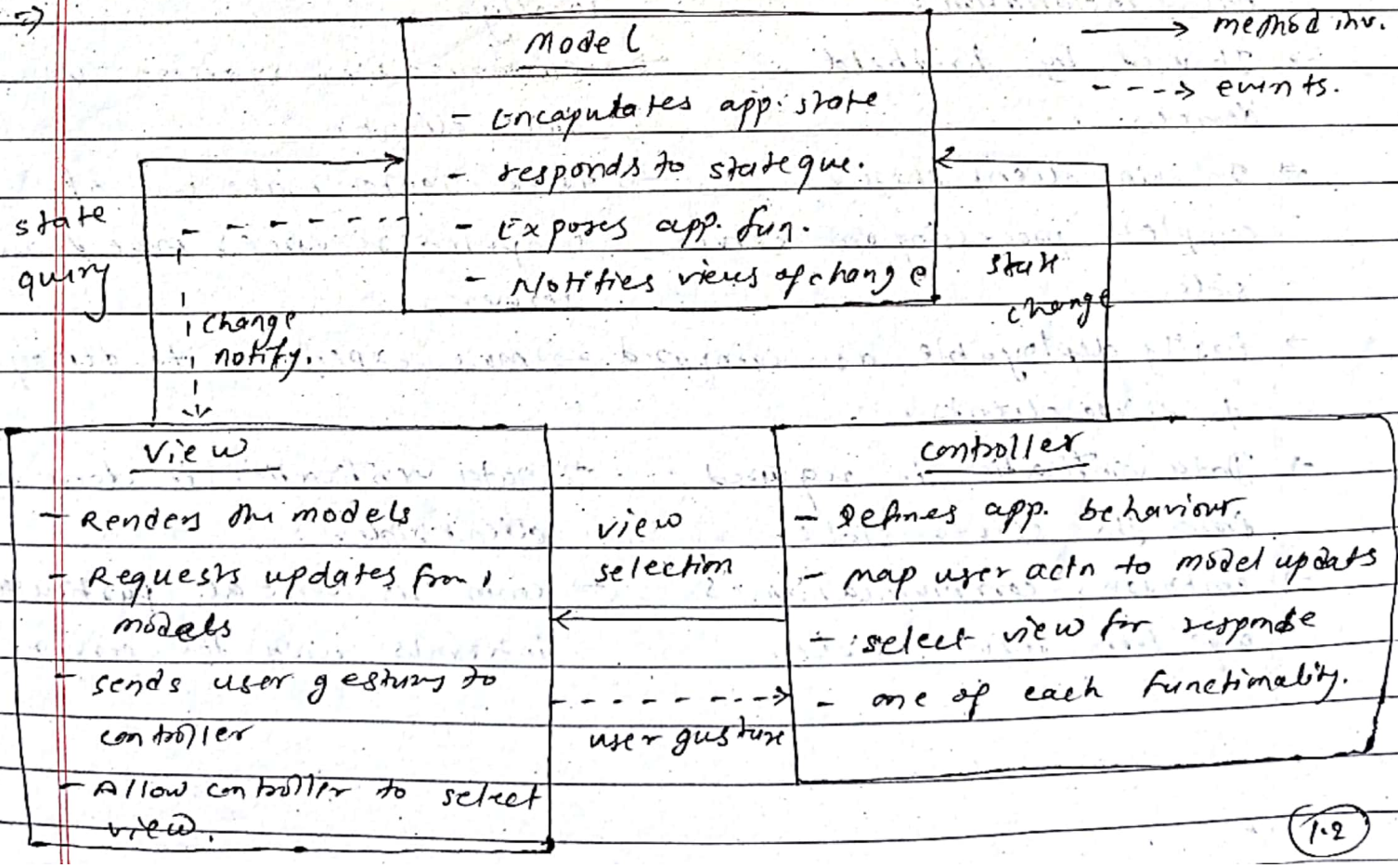
→ When cross-cutting concern code mixed with the core code

(g) Scattering.

→ When cross-cutting concern code used in multiple methods

2. Explain relationship between model, view and controller in MVC architecture. Differentiate between Thin and Thick client.

=>



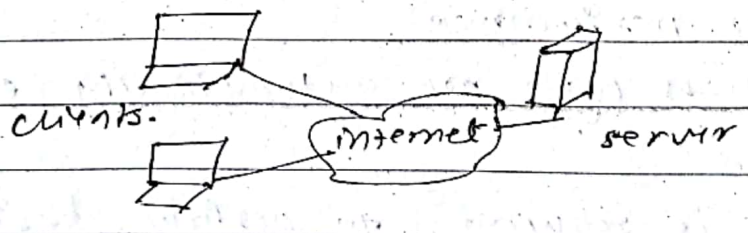
- User interacts with the user interface in some way.
- Controller handles the input event from the user interface, often via a registered handler or callback.
- The controller access the model, posibly updating it in a way appropriate to the user's action. Complex controllers are often structured using the command pattern to encaptulate actions and simplify extension.
- View use the model to generate an appropriate user interface.
- User interface waits for further user interactions, which begins a new cycle.

Thin clients	Thick clients
→ Thin clients have browser based installation.	→ Thick clients are installed locally.
→ It used by handheld devices	→ Customization systems use thick client.
→ In thin client there is complete processing on server side.	→ Thick clients make use of computer resources more than server
→ Easily deployable as compared to thick clients.	→ More expensive to deploy
→ Data verification is required from the server side.	→ Data verification is done by client side.
→ continuous communication is req'd from server side.	→ comm' is done at particular intervals with the server

Date
Page

5. How client side server architecture is related to design the solution of large software problem? Explain each of them with pros and cons.

⇒ To divide the work between several different computers monolithic system changes into client-server system. The client-server model is distributed application structure in computing that partitions tasks between the providers of service called server and service requesters called clients. Clients and servers communicate over a computer network on separate hardware, but both client and server may reside on the same system.
clients initiate communication sessions with servers that await incomming requests.
For the solution of large software, task need to distribute, Business data is the most important assests. Maintaining database on server separate from the client allows for greater security, reliability and performance. So client-server architecture is related to sol of large software problem.

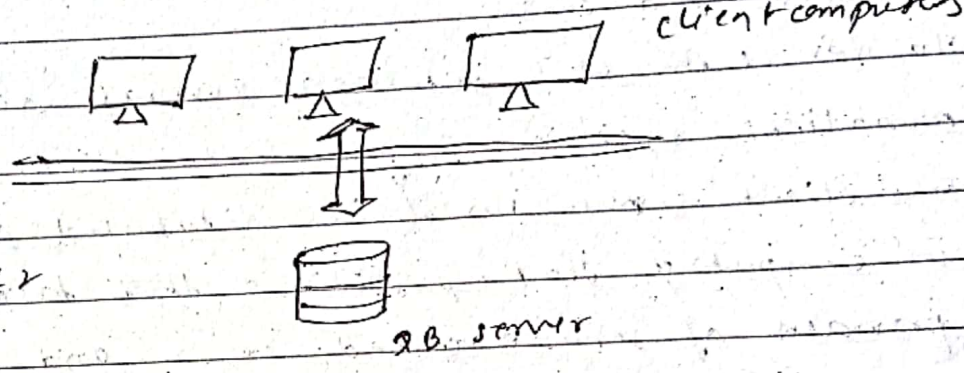


client-server arch.

There are several types of client server architecture:

(10) 2-tire client server architecture.

client directly interacts with the server.



Advantages:

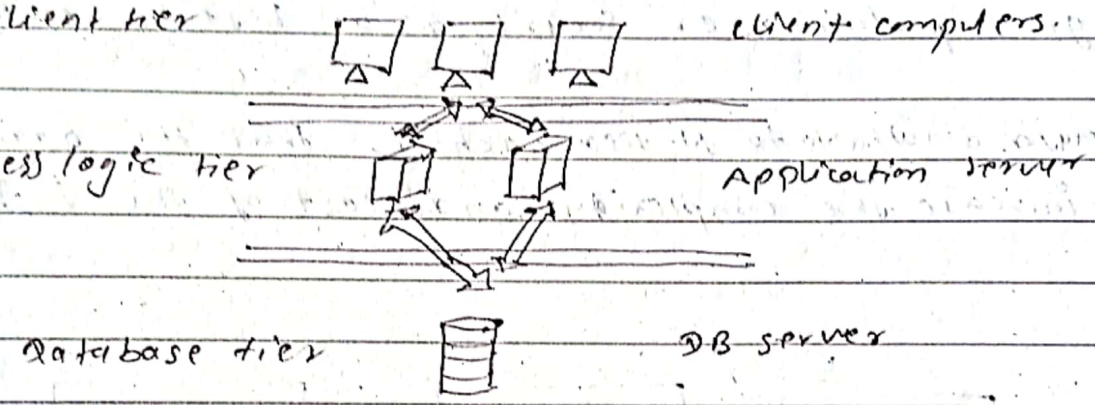
- simple structure
- Easy to setup and maintain
- Adequate performance for low to medium volume environm
- Business logic and database are physically close, which provides higher performance.

Disadvantages:

- Complex application rules difficult to implement in database server - requires more code for the client.
- client have poor performance
- changes to business logic not automatically enforced by a server.
- since db server is required to perform business logic this slow down db operation on db servers.

(b) 3-tier architecture

In this architecture, one more software sits in between client and server. This middle software is called middleware.



Advantages:

- complex app. rules easy to implement in app. server.
- Business logic off-loaded from DB server and client, which improves performance.
- superior performance for medium to high volume environments.

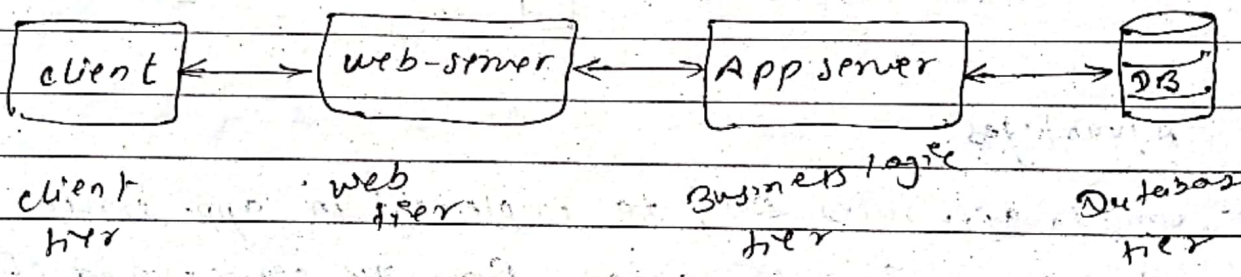
Disadvantages:

- more complex structure
- more difficult to setup and maintain
- The physical separation of application servers containing business logic functions and database server containing databases may moderately affect performance.

(b) N-tier architecture.

As more users access the system a three-tier solution is more scalable than the other solutions because we can add as many middle tiers as needed to ensure good performance. Security also best in multi tyer.

Major drawback to N-tier arch: is that the additional tiers increase the complexity and cost of the installation.



Example of N-tier app arch: is web-based application. It might consists.

Tier 1: client tier, presentation implemented by web browser

Tier 2: middle-tier distribution mechanism implemented by a web server.

Tier 3: Middle-tier service implemented by a set of server side scripts.

Tier 4: data-tier storage mechanism implemented by an database.

Date
Page

6. What is .NET framework? Explain the basic C# program structure with an example.

⇒ .NET framework is a ~~pla~~ platform developed by microsoft with which we can develop i) Windows app ii) web app iii) web services.

A simple Hello world program written in C# is

```
1  using System;
2  namespace HelloWorldApp {
3      class HelloWorld {
4          public static void Main() {
5              Console.WriteLine("Hello, World");
6              Console.ReadLine();
7          }
8      }
9  }
```

- In line 1, we are using a 'using' statement, it is used to include system namespace.
- In line 2, we create a namespace. Namespace are just logical grouping of various concepts and can be used to avoide name collision.
- Then we create a HelloWorld class, inside which we define our Main() function.
- Then we use ReadLine() to hold console and WriteLine() to print.
- The execution of our program begins from Main() method always.

Date
Page

Q. Define the term : class, object, event and delegates in context of c# programming.

=) class :

classes are the blueprint of objects or entities, they are used to express various concepts.

In c# we use class keyword to create class.

```
class Person {
    String name;
    int age;
    public void setName() {
        ...
    }
    public void display() {
    }
}
```

object :

Objects are anything that consume memory and possesses methods and behaviour defined in the class.

eg. person obj; // create object obj of class Person
obj.display() // call methods

Events:

Events are the any user actions through GUI. eg. button click, mouse move etc.

Delegates:

Delegate is a reference type variable that holds reference to method. They are similar in concepts with c/c++ pointers. They are used in events and call back methods.

8. Introduce C# programming with its basic data types, literals, constants, operators and variables.

=) C# is a strongly typed programming language and supports various data types.

eg.

i) int : stores 32 bit integer values.

ii) bool: boolean value ie. True or False

iii) float: floating point value.

iv) byte : 8 bit unsigned integer etc.

Data types are used to identify the type of data, a variable stores.

constants:

constants are fixed values, that can not be changed. constants must be initialized at the time of declaration. We use 'const' keyword to declare a constant.

Date
Page

Literals:

Fixed values are also referred as a literal. C# supports various type of literals:

i) integer literals

513 → int

250U → unsigned int

0123 → octal

ii) floating point

200.05

210E-5L

iii) string literal

"Hello, world"

operators:

operators are used to perform various kind of operations between operands.

some operators support by C# are:

→ Addition (+)

→ subtraction (-)

→ division (/)

→ Modulo (%)

→ multiplication (*)

→ conditional operator (? :)

→ logical AND (&&)

→ logical NOT (!)

→ assignment (=).

etc.

Date
Page

variables:

variables are used to store values.

eg: int x;

x = 5;

Here x is variable that store integer value ie. 5.

- However variable name can not be start with ~~letters~~ numbers, symbol except underscore (_).
- It does not contains any special symbol like $, & etc.
- A keyword (like class) can not be used as a variable name.

9. Explain each of the following with relevant example program. How C# programming provides the basic object oriented features like encapsulation, abstraction, polymorphism and inheritance.

=) Encapsulation:

It is a way of ~~binding~~ bundling member functions with data that they operate on. class are good example of encapsulation.

```
using system;
    class Box {
    private double l, b, h;
    public double getVolume() {
        return (l * b * h);
    }
```

```
Box (double a, double b, double c) {
        l = a;
    this.b = b;
        c = h;
    }
}

class BoxTester {
    public static void main() {
        Box b1 = new Box(5, 5, 5);
        double vol = b1.getVolume();

        Console.WriteLine("vol. of box = " + vol);
    }
}
```

Ans:

vol. of box = 125.0

END

Date
Page

Abstraction:

Data abstraction is the process of hiding implementation with the users.

We use abstract class to obtain abstraction in C#.

```
using System;
abstract class Animal {
    public abstract void makeNoise();
    public void walk() {
        Console.WriteLine("walking");
    }
}
class Dog : Animal {
    public override void makeNoise() {
        Console.WriteLine("Barking...");
    }
}
class Tester {
    static void Main() {
        Dog dog = new Dog();
        dog.walk();
        dog.makeNoise();
    }
}
```

We defined an abstract class, which consists of different methods. Another class inherited the abstract class and was able to invoke walk() function, however it is totally unknown that how the function is being implemented.

Date
Page

Inheritance:

Inheritance enhance code resuability and is a way of inheriting public methods and variables from parent class.

```
using System;
class Dog {
    public void makeNoise() {
        Console.WriteLine("Barking---");
    }
    public void sleep() {
        Console.WriteLine("sleeping---");
    }
}
class Test : Dog {
}

class program {
    static void Main() {
        Test t = new Test();
        t.sleep();
        t.makeNoise();
    }
}
```

Date
Page

Polymorphism :

```
using system;
class vehicle {
    public virtual void intro() {
        Console.WriteLine("I am a vehicle");
    }
}
class car : vehicle {
    public override void intro() {
        Console.WriteLine("I am a car");
    }
}
class Program {
    static void Main() {
        Vehicle v = new vehicle();
        Car c = new Car();
        v.intro();
        c.intro();
    }
}
```

Polymorphism means different form and is can be obtained by using virtual function, method overloading etc. Here we use virtual function in parent class and it is override in child class.

10. What are the basic difference between C# array and C# collections? Explain few of the available collections classes in system.collections namespace.

⇒

Array	Collections.
→ It is the collection of elements of homogeneous type.	→ It is the collection of elements of both homogeneous and heterogeneous type.
→ It is of fixed size	→ Its size can be increase and shrink.
→ Array is strongly typed.	→ They are not strongly typed.

Under System.Collections namespace, there are several collections type. some of which are:

(i) List:

It is a generic collection type.

declaration

```
List<string> names = new List<string>();
```

initialization.

```
List<int> years = new List<int>{2020, 2001, 2022};
```

(ii) Dictionary.

It is a set of key, value with unique keys.

declaration:

```
Dictionary<string, string> phoneBook = new Dictionary
                                        <string, string>();
```

adding elements:

```
phoneBook.Add("Mr. Robot", "Nowhere");
```

(iii) Queue

It is a collection of items implemented in first in first out manner.

declaration:

```
Queue<int> q = new Queue<int>();
```

add: `q.Enqueue(4);`

remove: `q.Dequeue();`

(iv) stack

It is the collection implemented in last in first out mechanism.

declaration:

```
Stack<int> s = new Stack<int>();
```

add: `s.push(2);`

remove: `s.pop();`

Date
Page

11. Why namespace system is important in c# ? Elaborate with suitable example.

=>

Namespace is mainly used for
→ Logical grouping of various entities
→ Avoiding naming conflit.

```
using System;
namespace MyNamespace {
    class MyWorld {
        public SayHello() {
            Console.WriteLine("Hello from MyNamespace");
        }
    }
}

class MyWorld {
    public SayHello() {
        Console.WriteLine("I am out of namespace");
    }
}

class Program {
    public static void main() {
        MyWorld m1 = new MyWorld();
        MyNamespace.MyWorld m2 = new Namespace.
                                 MyWorld();
        m1.SayHello();
        m2.SayHello();
    }
}
```

Here same class and same methods are used.
one class is inside MyNamespace namespace and another class is outside the namespace.
So there is no error or conflict in name or identifier.

the o/p will be.

```
I am out of namespace
Hello from MyNamespace.
```