Velammal College of Engineering and Technology, Madurai-625009
(Autonomous)
Department of Computer Science and Engineering
Assignment I

Degree : B.E/CSE Year/Semester/section : II/III/B
Course Code-Title :21CS203/Object Oriented Programming Batch: 2021-25
Name of the Instructor:J.Shanthalakshmi Revathy
Announcement Date : 18.09.22 Submission Date : 25.09.22
Total Marks : 20 Relevant COs : CO1-K2

Questions

1. Relate OOPs concepts with real time Examples
2. What does the framework mean in Java? Explain in detail any one Java Framework.

Instructions:

- Submit softcopy with neat formatting -On time submission required
- Specify at least five references.
- Everyone should select individual examples and framework

Evaluation Rubrics:

Content	**14**
References : 1.java point 2.tutorials point 3.geeksforgeek 4.stechies 5.gitup	**2**
Presentation	**2**
Timely submission	**2**
Total	**20**

RELATE OOPS'S CONCEPT WITH REAL TIME EXAMPLES

Object Oriented Programming is considered as a design methodology for building non-rigid software. In OOPS, every logic is written to get our work done, but represented in form of Objects. OOP allows us to break our problems into small unit of work that is represented via objects and their functions. We build functions around objects.

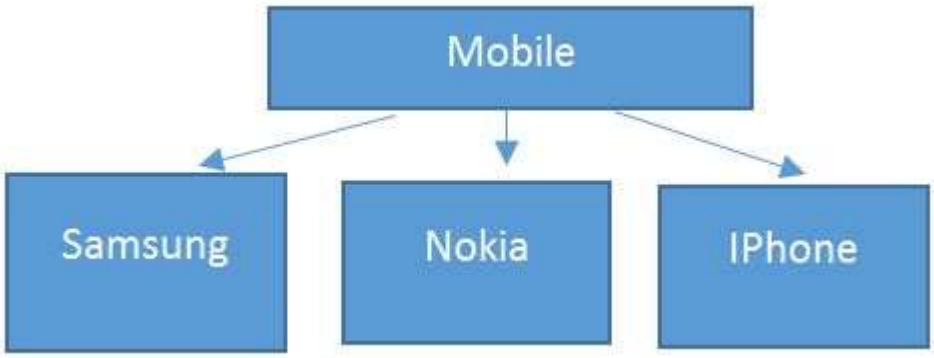
There are mainly four pillars (features) of OOP. If all of these four features are presented in programming, the programming is called perfect Object Oriented Programming.

1. Abstraction
2. Encapsulation
3. Inheritance
4. Polymorphism

Let's consider an example explaining each of these pillars so you can better understand Object
Oriented Programming.

Before that, we need to know something.

When we think of a mobile phone as an object, its basic functionality for which it was invented were Calling & Receiving a call & Messaging. But now a days thousands of new features and models were added and the features and number of models are still growing.



In above diagram, each brand (Samsung, Nokia, IPhone) have their own list of features along with basic functionality of dialing, receiving a call and messaging.

Objects

Any real world entity which can have some characteristics or which can perform some tasks is called as Object. This object is also called an instance i.e. a copy of entity in programming language. If we consider the above example, a mobile manufacturing company can be an object. Each object can be different based on their characteristics. For example, here are two objects.

```
1. Mobile mbl1 = new Mobile ();
2. Mobile mbl2 = new Mobile ();
```

Class

A class in OOP is a plan which describes the object. We call it a blueprint of how the object should be represented. Mainly a class would consist of a name, attributes, and operations. Considering the above example, the Mobile can be a class, which has some attributes like Profile Type, IMEI Number, Processor, and some more. It can have operations like Dial, Receive and SendMessage.

There are some OOPS principle that need to be satisfied while creating a class. This principle is called as SOLID where each letter has some specification. I won't be going into this points deeper. A single line of each explanation may clear you with some points.

1. SRP (The Single Responsibility Principle) - A class should have one, and only one responsibility

2. OCP (The Open Closed Principle) - You should be able to extend a classes behavior, without modifying it. (Inheritance)

3. LSP (The Liskov Substitution Principle) - Derived classes must be substitutable for their base classes. (Polymorphism)

4. ISP (The Interface Segregation Principle) -Make fine chopped interface instead of huge interface as client cannot be forced to implement an interface which they don't use.

5. DIP (The Dependency Inversion Principle) - Depend on abstractions, not on concretions. (Abstraction)

In code, the Mobile class looks like the following:

```
1.  public class Mobile
2.  {
3.      private string IEMICode { get; set; }
4.      public string SIMCard { get; set; }
5.      public string Processor { get; }
6.      public int InternalMemory { get; }
7.      public bool IsSingleSIM { get; set; }
8.
9.      public void GetIEMICode()
10.     {
11.         Console.WriteLine("IEMI Code - IEDF34343435235");
12.     }
13.
14.     public void Dial()
15.     {
```

```
        Console.WriteLine("Dial a number");
    }
    public void Receive()
    {
        Console.WriteLine("Receive a call");
    }
    public virtual void SendMessage()
    {
        Console.WriteLine("Message Sent");
    }
}
```

Abstraction

Abstraction allows us to expose limited data and functionality of objects publicly and hide the actual implementation. It is the most important pillar in OOPS. In our example of Mobile class and objects like Nokia, Samsung, IPhone.

Some features of mobiles,

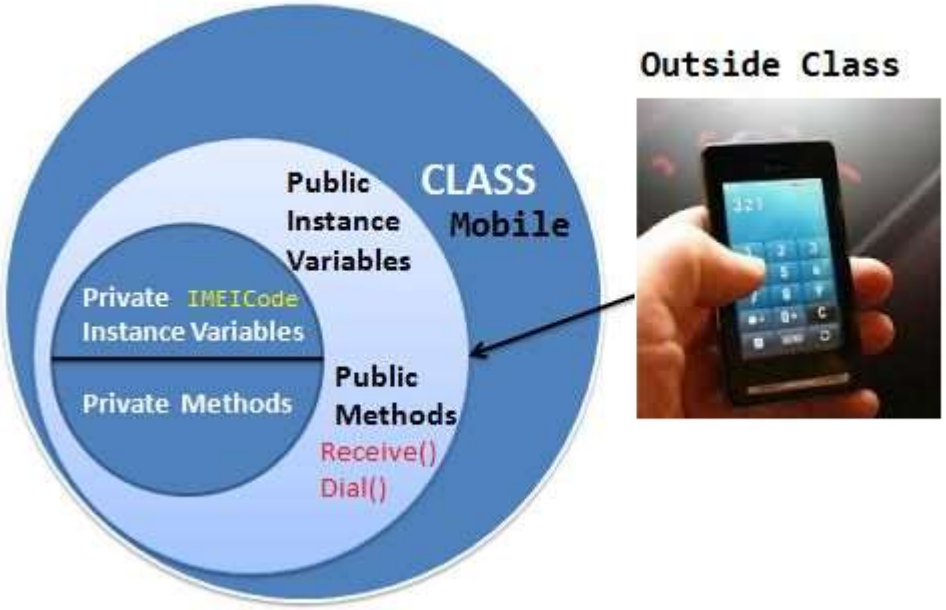
1. Dialing a number call some method internally which concatenate the numbers and displays it on screen but what is it doing we don't know.

2. Clicking on green button actual send signals to calling person's mobile but we are unaware of how it is doing.

This is called abstraction. In classes, we can create methods that can be called and used by the users of the class but users will have no idea what these methods do internally.

```
public void Dial()
{
    //Write the logic
    Console.WriteLine("Dial a number");
}
```

Encapsulation



Encapsulation is defined as the process of enclosing one or more details from outside world through access right. It says how much access should be given to particular details. Both Abstraction & Encapsulation works hand in hand because Abstraction says what details to be made visible and Encapsulation provides the level of access right to that visible details. i.e. – It implements the desired level of abstraction.

Talking about Bluetooth which we usually have it in our mobile. When we switch on a Bluetooth, I am able to connect to another mobile or bluetooth enabled devices but I'm not able to access the other mobile features like dialing a number, accessing inbox etc. This is because, Bluetooth feature is given some level of abstraction.

Another point is when mobile A is connected with mobile B via Bluetooth whereas mobile B is already connected to mobile C then A is not allowed to connect C via B. This is because of accessibility restriction.

Polymorphism

Polymorphism can be defined as the ability of using the same name for doing different things. More precisely we say it as 'many forms of single entity'. This play a vital role in the concept of OOPS.

Let's say Samsung mobile has a 5MP camera available i.e. – it is having a functionality of CameraClick(). Now same mobile is having Panorama mode available in camera, so functionality would be same but with mode. This type is said to be Static polymorphism or Compile time polymorphism. See the example below:

```
1.  public class Samsumg : Mobile
2.  {
3.      public void GetWIFIConnection()
4.      {
5.          Console.WriteLine("WIFI connected");
```

```
    }

    //This is one mwthod which shows camera functionality
    public void CameraClick()
    {
        Console.WriteLine("Camera clicked");
    }

    //This is one overloaded method which shows camera functionality as well but with its camera's different mode(panaroma)
    public void CameraClick(string CameraMode)
    {
        Console.WriteLine("Camera clicked in " + CameraMode + " Mode");

    }
}
```

Compile time polymorphism the compiler knows which overloaded method it is going to call.

Compiler checks the type and number of parameters passed to the method and decides which method to call and it will give an error if there are no methods that matches the method signature of the method that is called at compile time.

Another point where in SendMessage was intended to send message to single person at a time but suppose Nokia had given provision for sending message to a group at once. i.e. - Overriding the functionality to send message to a group. This type is called Dynamic polymorphism or Runtime polymorphism.

For overriding you need to set the method, which can be overridden to virtual & its new implementation should be decorated with override keyword.

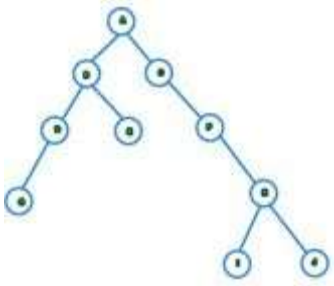
```
public class Nokia : Mobile
{
    public void GetBlueToothConnection()
    {
        Console.WriteLine("Bluetooth connected");
    }

    //New implementation for this method which was available in Mobile Class
    //This is runtime polymorphism
    public override void SendMessage()
    {
        Console.WriteLine("Message Sent to a group");
    }
}
```

By runtime polymorphism, we can point to any derived class from the object of the base class at runtime that shows the ability of runtime binding.

Inheritance

-



Inheritance is the ability to extend the functionality from base entity in new entity belonging to same group. This will help us to reuse the functionality which is already defined before and extend into a new entity.

Considering the example, the above figure 1.1 itself shows what is inheritance. Basic Mobile functionality is to send a message, dial and receive a call. So the brands of mobile is using this basic functionality by extending the mobile class functionality and adding their own new features to their respective brand.

There are mainly 4 types of inheritance,

1. Single level inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Hybrid inheritance
5. Multiple inheritance

example

```
1.  public class Mobile
2.  {
3.        //Properties
4.        //Methods
5.  }
6.
7.  public class Samsumg : Mobile
8.  {
9.        //Properties
10.       //Methods
11.}
12.
13.public class Nokia : Mobile
14.{
15.    //Properties
16.   //Methods
17.}
```

Interface

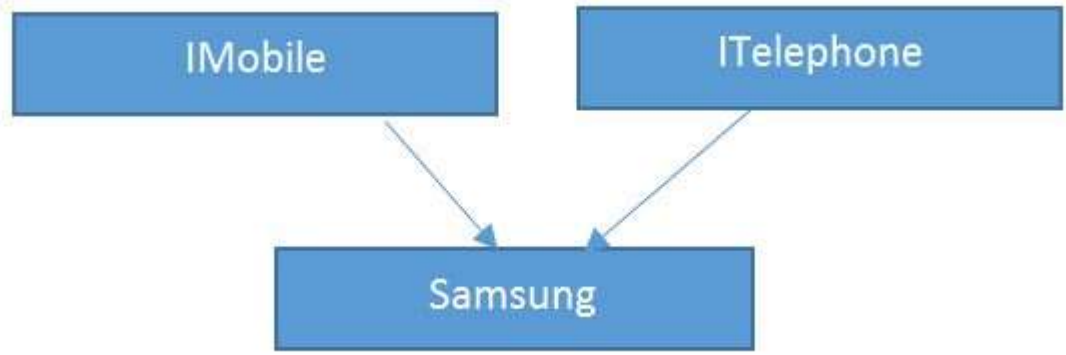
Multiple inheritance where derived class will extend from multiple base classes.

Samsung will use the function of multiple Phone (Mobile & Telephone). This would create a confusion for complier to understand which function to be called when any event in mobile is triggered like Dial () where Dial is available in both the Phone i.e. -(Mobile & Telephone). To avoid this confusion C# came with the concept of interface which is different from multiple inheritance actually.

If we take an interface it is similar to a class but without implementation & only declaration of properties, methods, delegates & events. Interface actually enforces the class to have a standard contract to provide all implementation to the interface members. Then what's is the use of interface when they do not have any implementation? Answer is, they are helpful for having readymade contracts, only we need to implement functionality over this contract.

I mean to say, Dial would remain Dial in case of Mobile or Telephone. It won't be fair if we give different name when its task is to Call the person.

Interface is defined with the keyword 'interface' .All properties & methods with in the interface should be implemented if it is been used. That's the rule of interface.




```
1.      interface IMobile
2.      {
3.         Void Dial();
4.      }
5.
6.      interface ITelephone
7.      {
8.         void Dial();
9.
10.     }
11.
12.     public class Mobile : IMobile, ITelephone
13.
14.     {
15.        public void Dial()
16.        {          Console.WriteLine("Dial a number");  }}
```

2.JAVA FRAMEWORK

Framework are the bodies that contains the pre-written codes (classes and functions) in which we can add our code to overcome the problem. We can also say that frameworks use programmer's code because the framework is in control of the programmer. We can use the framework by calling its methods, inheritance, and supplying "callbacks", listeners, or other implementations of the Observer pattern.

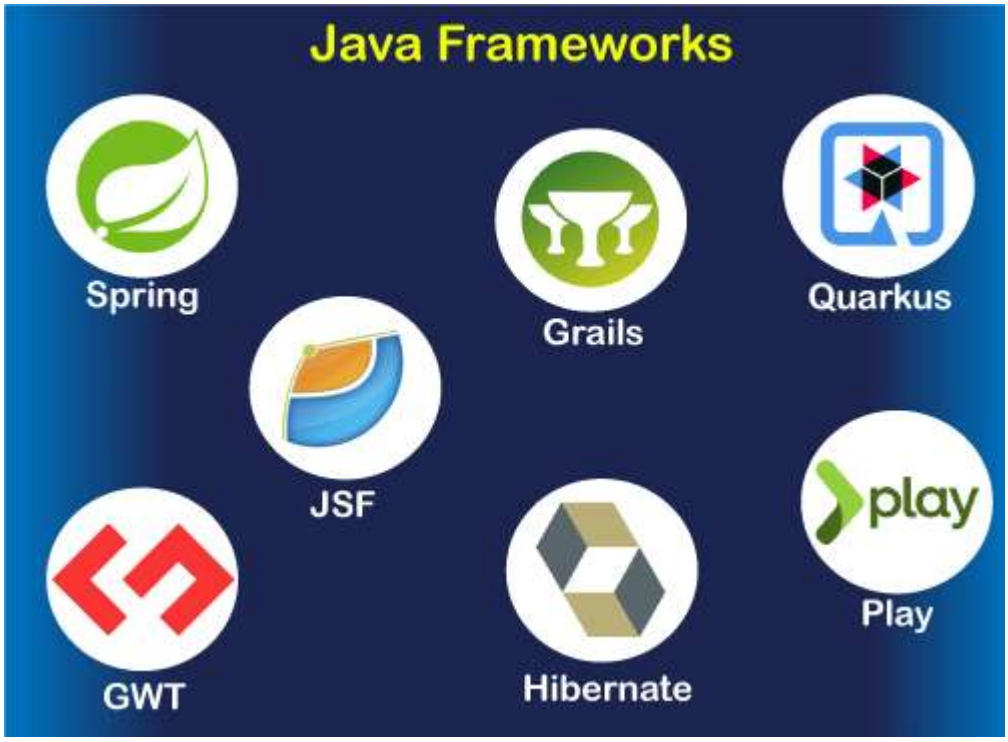
ava is an object-oriented programming (OOP) language that's been used since 1995. Developers use Java to program applications to work within the boundaries of the domain they're in.

Java shouldn't be confused with JavaScript, which is a different programming language developed by another company around the same time. Java development creates apps that can run in browsers or on servers' bare metal machines, virtual machines, or containers. JavaScript only runs in browsers.

Popular Java Frameworks

Some of the most popular Java frameworks are:

- Spring
- Hibernate
- Grails
- Play
- JavaServer Faces (JSF)
- Google Web Toolkit (GWT)
- Quarkus



Java server faces

It is a server side component based user interface framework. It is used to develop web applications. It provides a well-defined programming model and consists of rich API and tag libraries. The latest version JSF 2 uses Facelets as its default templating system. It is written in Java.

The JSF API provides components (inputText, commandButton etc) and helps to manage their states. It also provides server-side validation, data conversion, defining page navigation, provides extensibility, supports for internationalization, accessibility etc.

The JSF Tag libraries are used to add components on the web pages and connect components with objects on the server. It also contains tag handlers that implements the component tag.

With the help of these features and tools, you can easily and effortlessly create server-side user interface.

Java Server Faces Versions History

	Release date	Description
Jsf 2.3	Expected in 2017	It may includes major features: bean validation for complete classes, push communication using enhanced integration with cdi.
Jsf 2.2	21-05-2013	It has introduced new concepts like stateless views, page flow and the ability to create portable resource contracts.
Jsf 2.1	22-11-2010	It was a maintenance release 2 of jsf 2.0. only a very minor number of specification changes.
Jsf 2.0	01-07-2009	It was major release for ease of use, enhanced functionality, and performance. coincides with java ee 6.
Jsf 1.2	11-05-2006	It has many improvements to core systems and apis. coincides with Java ee 5. initial adoption into java ee.
Jsf 1.1	27-05-2004	It was a bug-fix release. no specification changes.
Jsf 1.0	11-03-2004	It was a initial specification released.

Benefits of JavaServer Faces

1) It provides clean and clear separation between behavior and presentation of web application. You can write business logic and user interface separately.

2) JavaServer Faces API?s are layered directly on top of the Servlet API. Which enables several various application use cases, such as using different presentation technologies, creating your own custom components directly from the component classes.

3) Including of Facelets technology in JavaServer Faces 2.0, provides massive advantages to it. Facelets is now the preferred presentation technology for building JavaServer Faces based web applications.

Prerequisites

Java: You must have Java 7 or higher version.

Java IDE: In this tutorial, we have used NetBean IDE 8.2. Although you can also use other Java IDEs.

Server: We did not install server seperatly. All the examples are executed on the default server installed along with NetBeans IDE 8.2.