



COMSATS Institute of Information Technology
Abbottabad
OOP Concepts

1. **Briefly describe the following in the context of object oriented programming:**
 - a) **Class:**
A class is a blueprint from which objects are created.
 - b) **Object:**
Objects consist of state and related behavior. An object stores its state in fields (variables in some programming languages) and exposes its behavior through methods.
2. **Describe two types of class members used in object oriented programming languages.**
Classes contain fields and methods. Fields contain the data that describes the object. Methods describe the operations that may be applied to the object.
3. **Explain how four types of (access modifiers) class member visibility is used in object oriented programming.**

Default:

If a variable is set to default, it will be accessible to the classes which are defined in the same package. Any method in any Class which is defined in the same package can access the variable via **Inheritance** or **Direct access**.

Public:

If a variable is set to public it can be accessible from any class available in the Java world. Any Method in any Class can access the given variable via **Inheritance** or **Direct access**.

Protected:

If a variable is set to protected inside a Class, it will be accessible from its sub classes defined in the same or different package only via **Inheritance**.

Note: The only difference between protected and default is that protected access modifiers respect **class subclass relation** while default does not.

Private:

A variable if defined private will be accessible only from within the Class in which it is defined. Such variables are not accessible from outside the defined Class, not even in its subclass .

Visibility	Public Access Modifier	Private Access Modifier	Protected Access Modifier	Default Access Modifier
Within Same Class	Yes	Yes	Yes	Yes

From Any Class in Same Package	Yes	No	Yes	Yes
From Any Sub Class in Same Package	Yes	No	Yes	Yes
From Any Sub Class from Different Package	Yes	No	Yes(Only By Inheritance)	No
From Any Non Sub Class in Different Package	Yes	No	No	No

Access Modifiers for Methods

Methods are eligible for all the following modifiers.

Default:

When a Method is set to default it will be accessible to the classes which are defined in the same package. Any Method in any Class which is defined in the same package can access the given Method via ***Inheritance or Direct access***.

Public:

When a Method is set to public it will be accessible from any Class available in the Java world. Any Method in any Class can access the given method via ***Inheritance or Direct access*** depending on Class level access.

Protected:

If a Method is set to protected inside a Class, it will be accessible from its sub classes defined in the same or different package.

Note:* The only difference between protected and default is that protected access modifiers respect **class subclass relation** while default does not.

Private:

A Method that is defined private will be accessible only from within the Class in which it is defined. Such Methods are not accessible from outside the defined Class, not even its subclass .

4. **Using an object-oriented language with which you are familiar, give an example of a class definition which illustrates the use of the concepts you described in your answer to parts 2 and 3.**

```
class MyClass {
    public int myPublicVar;
    private int myPrivateVar;
    protected int myProtectedVar;
    private void myPrivateMethod(){ }
    public void myPublicMethod(){ }
    protected void myProtectedMethod(){ }
}
class MySubclass extends MyClass {
```

```

        private void MySubclassMethod() {
            myPublicVar++;
            myProtectedVar++;
        }
    }
}
class AnotherClass{
    public void AnotherClassMethod() {
        MyClass myObject = new MyClass();
        myObject.myPublicVar++;
    }
}

```

5. **You have been invited to give a talk to trainee programmers outlining the reasons for the widespread use of object oriented programming within the software development industry. Summarise the points you would present in your talk.**

There are three main ways in which object oriented programming has been found to improve programmer productivity:

- i. By providing an environment which assists programmers to improve the quality of their code;
- ii. Making it easier for programmers to reuse their own code;
- iii. Providing simple mechanisms to make use of existing code libraries.

The disadvantages of object-oriented programming include the learning curve necessary to become proficient in it and the fact that code produced by an object-oriented language compiler is unlikely to be as efficient as code produced by the best machine code programmers.

6. **Give the meaning of the following terms:**

a) Subclass

A class that is derived from another class is called a subclass.

b) Superclass

The class from which the subclass is derived is called a superclass.

c) Abstract Class

They are classes that cannot be instantiated, and are frequently either partially implemented, or not at all implemented. They are closely related to interfaces.

d) Interface

- An interface in java is a blueprint of a class.
- It has static constants and abstract methods.
- The interface in java is a mechanism to achieve abstraction.
- There can be only abstract methods in the java interface not method body.
- It is used to achieve abstraction and multiple inheritance in Java.
- Java Interface also represents IS-A relationship.
- It cannot be instantiated just like abstract class.

e) Inheritance


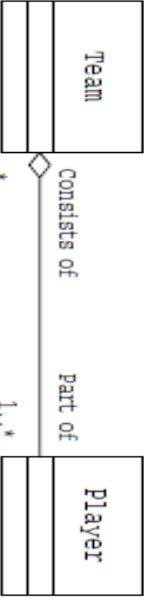

- Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.

- The idea behind inheritance in java is that you can create new classes that are built upon existing classes.
- When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.
- Inheritance represents the IS-A relationship.

f) Association, Aggregation, Composition:

Following is the comparison between all the varieties of HAS-A relationships:

Association	Aggregation	Composition
Class A uses Class B.	Class A owns Class B.	Class A contains Class B.
Example: <ul style="list-style-type: none"> • Employee uses BusService for transportation. • Client-Server model. • Computer uses keyboard as input device. 	Example: <ul style="list-style-type: none"> • Manager has N Employees for a project. • Team has Players. 	Example: <ul style="list-style-type: none"> • Order consists of LineItems. • Body consists of Arm, Head, Legs. • BankAccount consists of Balance and TransactionHistory.
<p>An association is used when one object wants another object to perform a service for it.</p> <p>Eg. Computer uses keyboard as input device.</p>	<p>An aggregation is used when life of object is independent of container object But still container object owns the aggregated object.</p> <p>Eg. Team has players, If team dissolve, Player will still exists.</p>	<p>A composition is used where each part may belong to only one whole at a time.</p> <p>Eg. A line item is part of an order so A line item cannot exist without an order.</p>
Association UML Notation: Associations are represented by just the line (no diamond).	Aggregation UML Notation: Aggregations are represented by the line with diamond.	Composition UML Notation: Compositions are represented by the line with filled diamond.

 <p>UML diagram representing an association.</p>	 <p>The Aggregate Relationship</p>	 <p>The Composite Relationship</p>
<p>Life or existence of the associated objects are independent of each other, They just provide service to each other.</p>	<p>Life or existence of the aggregated objects are independent of each other, But one object is playing the role of Owner of the other object.</p>	<p>Life or existence of the composite object is dependent on the existence of container object, Existence of composite object is not meaningful without its container object.</p>

What If You are Not Sure What Type of Whole-Part Relationship to Use?

- Do not get too confuse if you are unable to decide whether a particular whole-part relationship is best described as an aggregation or composition because the distinction is helpful, it is not critical.
- **If you are not even sure whether the relationship is best described as Association or Composition or Aggregation, then model it as an Association.**
- In fact, there is nothing wrong with modelling all whole-part relationships as simple Associations.
- You lose a little bit of distinction in the model, But it will make no difference to the resulting IT solution.

g) Exception Handling:

The exception handling in java is one of the powerful mechanism to handle the runtime errors so that normal flow of the application can be maintained.

In java, exception is an event that disrupts the normal flow of the program. It is an object which is thrown at runtime.

Exception Handling is a mechanism to handle runtime errors such as ClassNotFoundException, IO, SQL, Remote etc.

The core advantage of exception handling is to maintain the normal flow of the application. Exception normally disrupts the normal flow of the application that is why we use exception handling.

There are mainly two types of exceptions: checked and unchecked where error is considered as unchecked exception. The sun microsystem says there are three types of exceptions: Checked Exception, Unchecked Exception, Error.

Checked Exceptions: The classes that extend Throwable class except RuntimeException and Error are known as checked exceptions e.g. IOException, SQLException etc. Checked exceptions are checked at compile-time.

Runtime Exceptions: The classes that extend RuntimeException are known as unchecked exceptions e.g. ArithmeticException, NullPointerException, ArrayIndexOutOfBoundsException etc. Unchecked exceptions are not checked at compile-time rather they are checked at runtime.

Errors: Error is irrecoverable e.g. OutOfMemoryError, VirtualMachineError, AssertionError etc.

There are 5 keywords used in java exception handling.

try, catch, finally, throw, throws

h) Final:

The final keyword in java is used to restrict the user. The java final keyword can be used in many context. Final can be: variable, method, class

- If you make any variable as final, you cannot change the value of final variable(It will be constant).
 - If you make any method as final, you cannot override it (but can be overloaded). Final method is inherited but you cannot override it.
 - If you make any class as final, you cannot extend it.
-
- a) A final variable that is not initialized at the time of declaration is known as blank final variable. If you want to create a variable that is initialized at the time of creating object and once initialized may not be changed, it is useful. It can be initialized only in constructor.
 - b) A static final variable that is not initialized at the time of declaration is known as static blank final variable. It can be initialized only in static block.
 - c) If you declare any parameter as final, you cannot change the value of it.
 - d) Constructor cannot be final, because constructor is never inherited.

i) Static:

- The static keyword in java is used for memory management mainly.
- We can apply java static keyword with variables, methods, blocks and nested class.
- The static keyword belongs to the class not the instance of the class.

The static can be: variable (also known as class variable), method (also known as class method), block, nested class

- If you declare any variable as static, it is known static variable.
 - a) The static variable can be used to refer the common property of all objects (that is not unique for each object) e.g. company name of employees, college name of students etc.
 - b) The static variable gets memory only once in class area at the time of class loading (memory efficient).
- If you apply static keyword with any method, it is known as static method.
 - a) A static method belongs to the class rather than object of a class.
 - b) A static method can be invoked without the need for creating an instance of a class.
 - c) Static method can access static data member and can change the value of it.
 - d) There are two main restrictions for the static method. They are:
 - The static method cannot use non static data member or call non-static method directly.
 - this and super cannot be used in static context.
- Java main method is static because object is not required to call static method if it were non-static method, jvm create object first then call main() method that will lead the problem of extra memory allocation.
- Static block is used to initialize the static data member. It is executed before main method at the time of class loading.

j) Inner Class (Nested Classes) & Anonymous Class:

- Java inner class or nested class is a class i.e. declared inside the class or interface.
- We use inner classes to logically group classes and interfaces in one place so that it can be more readable and maintainable.
- Additionally, it can access all the members of outer class including private data members and methods.

There are basically three advantages of inner classes in java. They are as follows:

1. Nested classes represent a special type of relationship that is it can access all the members (data members and methods) of outer class including private.
2. Nested classes are used to develop more readable and maintainable code because it logically group classes and interfaces in one place only.
3. Code Optimization: It requires less code to write.

Types of inner classes:

Type	Description
Member Inner Class	A class created within class and outside method.
Anonymous Inner Class	A class created for implementing interface or extending class. Its name is decided by the java compiler.
Local Inner Class	A class created within method.
Static Nested Class	A static class created within class.
Nested Interface	An interface created within class or interface.

Anonymous inner class: A class that have no name is known as anonymous inner class in java. It should be used if you must override method of class or interface. Two ways can create Java Anonymous inner class: Class (may be abstract or concrete), Interface.

```
abstract class Person{
    abstract void eat();
}
class TestAnonymousInner{
    public static void main(String args[]){
        Person p=new Person(){
            void eat(){System.out.println("nice fruits");}
        };
        p.eat();
    }
}
```

k) Package:

A java package is a group of similar types of classes, interfaces and sub-packages. Package in java can be categorized in two form, built-in package and user-defined package. There are many built-in packages such as java, lang, awt, javax, swing, net, io, util, sql etc.

l) Polymorphism:

Polymorphism is the ability for classes to provide different implementations of methods that are called by the same name. Polymorphism allows a method of a class to be called without regard to what specific implementation it provides.

m) Event Handling:

Changing the state of an object is known as an event. For example, click on button, dragging mouse etc. The java.awt.event package provides many event classes and Listener interfaces for event handling.

Following steps are required to perform event handling:

- Register the component with the Listener
- Write event handling code. We can put the event handling code into one of the following places: Within class, Other class, Anonymous class

Anonymous class example:

```
import java.awt.*;
import java.awt.event.*;
class AEvent3 extends Frame{
    TextField tf;
    AEvent3(){
        tf=new TextField();
        tf.setBounds(60,50,170,20);
        Button b=new Button("click me");
        b.setBounds(50,120,80,30);

        b.addActionListener(new ActionListener(){
            public void actionPerformed(){
                tf.setText("hello");
            }
        });
        add(b);add(tf);
        setSize(300,300);
        setLayout(null);
        setVisible(true);
    }
    public static void main(String args[]){
        new AEvent3();
    }
}
```

n) Multi-Threading:

Multithreading in java is a process of executing multiple threads simultaneously.

- Thread is basically a lightweight sub-process, a smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.
- But we use multithreading than multiprocessing because threads share a common memory area.
- They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.
- Java Multithreading is mostly used in games, animation etc.

7. Consider the following class definition:

```
public class date {
    private int day;    // from 1 to 31
    private int month;  // from 1 to 12
    private int year;   // from 2000 upwards
    public void advance(); // move to next day
};
```

- a) Implement a constructor that initializes new objects of date class to be set to the 1st of January 2000.

```

public date() {
    day = 1;
    month = 1;
    year = 2000;
}

```

b) Implement setters for day, month and year.

```

public bool setDay(int newDay) {
    if(newDay>0 && newDay <32)    {
        day = newDay;    return true;
    }
    return false;
}

public bool setMonth(int newMonth) {
    if(newMonth>0 && newMonth <13)    {
        month = newMonth;
        return true;
    }
    return false;
}

public bool setYear(int newYear) {
    if(newYear>1999)    {
        year = newYear;
        return true;
    }
    return false;
}

```

c) Implement the advance method, which moves to the next day, ensuring that all data members are updated appropriately.

```

public void advance() {
    static final int
    daysInMonth[12]={31,28,31,30,31,30,31,31,30,31,30,31};
    day++;
    if(day>daysInMonth(month))    {
        day = 1;
        month++;
        if(month>12)    {
            month = 1;
            year++;    }}}

```