CORE JAVA

MANUAL V8.3

MODULE CODE:

ANUDIP FOUNDATION


ICONS AND THEIR MEANING
HINTS:
Get ready for helpful insites on difficult topics and questions.
STUDENTS:
This icon symbolize important instrcutions and guides for the students.
TEACHERS/TRAINERS:
This icon symbolize important instrcutions and guides for the trainers.

Module 2: Object Oriented Programming and Package

Chapter 3

Objective: After completing this lesson you will be able to : * Gain an understanding of interface in Java * Gain an introduction to aggregation in Java	**Materials Required:** 1. Computer 2. Internet access
Theory Duration: 60 minutes	**Practical Duration:** 60 minutes
Total Duration: 120 minutes	

Chapter 3

3.1 Interface

An interface in Java is an abstract type for defining the behavior of classes. It states what action a class must perform, and how. A Java interface can be considered as the blueprint of a class. It specifies the methods to be implemented by a class. Interface methods are abstract by default as they only contain method signatures.

Interface is utilized for achieving complete abstraction, to selectively show and hide object properties from users. Methods within an interface are public, abstract and contain empty bodies.

An interface is declared with the 'interface' keyword in Java. A class has to implement all methods of an interface to implement it. Interfaces cannot be used to instantiate (incorporate) variables.

Syntax:

```
interface <interface_name> {

}
```

Example of Interface:

```
import java.io.*;

interface in1
{
    final int a = 5;

    void display();
}
```

```
class testClass implements in1
{
    public void display()
    {
        System.out.println('Boy');
    }

    public static void main (String[] args)
    {
        testClass t = new testClass();
        t.display();
        System.out.println(a);

    }
}
```

Output:

```
Boy
5
```

Reasons for using interface in Java –

* To achieve full abstraction
* To achieve loose coupling
* To achieve multiple inheritance, which Java does not support by default
* Preferred over abstract classes as interfaces contain public, static and final variables. Abstract classes have non-final variables.

3.2 Aggregation

In Java, association refers to the relationship between two classes. A class can be associated with another through its

objects. Through association, one object can establish a connection to another and use the latter's features.

'Aggregation' is a type of association that represents a Has-A relationship. It is a one-way association. An example of this relationship is the one between 'bag' and 'coins'. The relationship is one-way because the bag has coins but the coins do not require the bag to exist. So, the coins class can exist even if the bag class ceases to exist.

An aggregate class possesses a reference to another class and gains ownership of the class. Aggregation is mainly used for enabling code reusability.

Example of the aggregation program:

```
class Table{
    public void show(){
        System.out.println('exhibit table details.');
    }
}

public class Room {
    public static void main(String args[]){
        Table obj = new Table();
        obj.show();
    }
}
```

Output:

Exhibit table details

Practical (60 minutes)

See the example programme of Java interface below. Write a similar programme for the object Boy and value of int b = 7. Write it again for the object Girl and the value of int c=9. Show the resulting outputs.

```
import java.io.*;

interface in1
{
    final int a = 5;

    void display();
}

class testClass implements in1
{
   public void display()
   {
      System.out.println('Boy');
   }

   public static void main (String[] args)
   {
      testClass t = new testClass();
      t.display();
      System.out.println(a);

   }
}
```

Instructions: The progress of students will be assessed with the exercises mentioned below.

MCQ

1. An interface is an _____ type in Java.

a) absolute

b) abstract

c) aggregate

d) None of the mentioned

2. A Java specifies the _____ to be implemented by a class

a) methods

b) codes

c) data types

d) None of the mentioned

3. What is the default nature of Interface methods?

a) abstract

b) absolute

c) assimilated

d) None of the mentioned

4. Methods within an interface contain _____ bodies.

a) empty

b) semi-full

c) full

d) None of the mentioned

5. Which keyword specifies an interface in Java?

a) int

b) interface

c) intersect

d) None of the mentioned

6. Interfaces cannot be used to _____ variables.

a) re-process

b) insinuate

c) instantiate

d) None of the mentioned

7. Interface is utilized for achieving _____ abstraction.

a) complete

b) semi-full

c) incomplete

d) None of the mentioned

8. Interfaces contain _____, static and final variables.

a) private

b) public

c) hybrid

d) None of the mentioned

9. What type of relationship does aggregation represent?

a) Had-A

b) Has-A

c) Have-A

d) None of the mentioned

10. Aggregation is mainly used for enabling code _____.

a) reusability

b) rejection

c) redevelopment

d) None of the mentioned