Day 7 :- Assignment } 22-04-2023

Abstraction ::

-> Hiding the implementation part.

-> Data abstraction is the process of hiding certain details and showing only essential information to the user.

Two types :.

Abstract class (partially abstraction)
Interface (fully abstraction)

Abstract class ::

- An abstract class and methods should be declared with an abstract keyword.
- It can have abstract and non-abstract methods.
- we can't create object.
- It can have constructors and static methods also.
- contains keyword extends.

Example program ::

```
package org.abstraction.interface;
public abstract void placementReport();
public void demo()
{ System.out.println("Demo Method");
}
public static void main (String
                          [] args)
{ // AiitAcademy a = new AiitAcademy
                          ();
  // cannot instantiate the
```

Abstract class.

```
}}
```

```
package org.abstractionInterface;
    public class AiitoJavaTrainer extends
                        AiitoAcademy {
    @Override
        public void salaryCalculation()
        { System.out.println("Java
          Trainer Salary - 25000");
        }
    @Override
        public void placementRecords()
        { System.out.println("placement
                    Percentage - 97%");
        }
    public static void main (String [] args)
        { AiitoJavaTrainer a = new AiitoJava
                                Trainer();
          a.placementRecords();
          a.salaryCalculation();
          a.demo();
        }
}
```

Points to Remember:

- An abstract class can have abstract and / or non-abstract methods.

- The abstract may or may not contain the final variables.
- An abstract class have final, static or non-static or non-final variables.
- An abstract class may provide interface implementation.
- An abstract class is is inherited using extends keyword.
- An abstract class can extend other classe or implement multiple interfales.
- An abstract class can have private or protected data members apart from public members.

INTERFACE :- (fully Abstraction)

A Java interface contain static constants can abstract methods.

- It contain abstract methods.
- It contain keyword implements.
- fields are public static and final by default.
- we can't create object
- only specifier can be used in public and abstract.
- we can have default and static method.

Example Program:-

```
package org.abstractionInterface;
public interface Smartphone{
        int num =5; // public static final;
    public abstract void internet();
    public abstract void touchscreen
    void display(); // by default
        it will take public abstract
    public default void message()
     { System.out.println("Interface
          default method");
    }
}
```

```
Package org.abstractioninterface;
Public class AiiteJavaTrainer extends
    Aiitakademy {
        @Override
        public void salaryCalculation()
        { System.out.Println("placement
            percentage - 97.1.");
        }
// Defaut method can Override
        public static void main
(string []args)
    { Samsung s = new Samsung();
      s.display();
      s.faicerecognition();
```

```
s.internet();
s.touchscreen();
s.message();
}}
```

Multiple Inheritance achieved using interface:.

```
Package org.abstractionInterface;
public interface Smartphone {
        int ram = 5;
        public abstract void touchscreen();
        public abstract void internet();
        void display();
      public default void message()
      { System.out.println("Interface default
  void hotspot(); method"); }
}
```

```
Package org.abstractionInterface;
public interface SmartTv {
        void wifi();
        void gaming();
        void app();
        void hotspot();
}
```

```
Package org.abstractioninterface;
public class Samsung implements
  Smartphone, SmartTv {
      @override
        public void touchscreen()
  } { System.out.println("Samsung Touchsc
```

```
@Override
public void internet()
  { System.out.println("Samsung internet");
  }
@Override
 public void display()
  { System.out.println("Samsung display");
  }
 public void facerecognition()
  { System.out.println("Samsung face
    recognition");
  }
 public void message()
    { // Default method can override
    }
public static void main(String []
    args)
  { Samsung s = new Samsung();
    s.display();
    s.facerecognition();
    s.internet();
    s.touchScreen();
    s.message();
    s.app();
    s.internet();
    s.gaming();
  }
@Override
      public void wifi()
      { System.out.println("Samsung
          TV wifi");
        }
```

```
@Override
    public void gaming()
    { System.out.println("Samsung Tv
                        gaming");
    }
@Override
    public void app()
    { System.out.println("samsung Tv
                        app");
    }
@Override
    public void hotspot()
    { System.out.println("common Method");
    }
}
```

Day-7 - NOTES TAKING:. -

Abstraction - hiding data (or) implementation part

→ Hiding certain details
→ showing essential detail to User.

```
Public abstract class aiit
{
    Salary_computing();
}
public abstract class java_trainers
{
    public void Salary_computing()
    {
        // business_logic
    }
}
```

two types of Abstraction

Abstract class - partially hide
Interface - fully hide.

non-Abstract class properties allowed.

There is no object created because unimplemented methods are there.

Interface - set of rules.

public class Samsung implements Smartphone

{

void methodname();

}

(*) variable in interface is public static final

int num = 5;

default method syntax: (only in interface)

public default void massage()

{ // business logics.

(:-) }

default method can override