**Declarations and Access Modifiers – Part-11- Interfaces-01**

* Agenda:
  + Introduction.
  + Interface declaration and Implementation.
  + Extends vs Implements.
  + Interface methods.
  + Interface variables.
  + Interface naming conflicts.
    - Method naming conflicts.
    - Variable naming conflicts.
  + Marker interface.
  + Adapter classes.
  + Interface vs abstract class vs concrete class.
  + Difference between interface and abstract classes.
  + Conclusions.
* Introduction:
  + Definition\_01:

Any service requirement specification (SRS) is considered as an interface.

Example\_01:

JDBC API acts as requirement specification to develop database driver.

Database vendor is responsible to implement this JDBC API.

SUN

JDBC API

Example\_02:

Servlet API acts as requirement specification to develop webserver.

Webserver vendor is responsible to implement Servlet API.

SUN

Servlet API

* + Definition\_02:

From client point of view an interface defines the set of services what he is expecting.

From service provider point of view an interface defines, the set of services what he is offering. Hence any contract between client and service provider is considered as an interface.

Example\_01:

Through Bank ATM GUI screen bank people are highlighting the set of services what they are offering. At the same time the same GUI screen represents the set of services what customer is expecting. Hence this GUI screen acts as contract between the customer and bank people.

Withdraw

Get Mini Statement

Balance Enquiry

Customer ATM GUI Screen Bank

* + Definition\_03:

Inside interface every method is always abstract whether we are declaring or not. Hence interface is considered as 100% pure abstract class.

* + Summary:

Any service requirement specification or any contract between client and service provider or 100% pure abstract class is nothing but an interface.

* Interface declaration and Implementation:
  + Whenever we are implementing an interface for each and every method of that interface we have to provide implementation, otherwise we have to declare class as abstract. Then next level child is responsible to provide implementation.
  + Every interface method is always public and abstract whether we are declaring or not. Hence whenever we are implementing an interface method, compulsory we should declare as public otherwise we will get compile time error.
  + Example:

interface Interf{

void m1();

void m2();

}

abstract class ServiceProvider implements Interf{

public void m1(){

}

}

Class SubServiceProvider extends ServiceProvider{

public void m2(){

}

}

* Extends vs Implements:
  + A class can extend only one class at a time.
  + An Interface can extend any number of interfaces simultaneously.
  + Example:

interface A{

}

interface B{

}

interface C extends A, B{

}

* + A class can implements any number of interfaces simultaneously.
  + A class can extend another class and can implement any number of interfaces simultaneously.
  + Example:

class A extends B implements C, D, E{

}

* + Which of the following is valid?
    - A class can extend any number of classes at a time. – Invalid
    - A class can implement only one interface at a time. - Invalid
    - An Interface can extend only one interface at a time. – Invalid
    - An Interface can implement any number of interfaces simultaneously. – Invalid
    - A class extend another class or can implement an interface but not both simultaneously. – Invalid
    - None of the above. // This is the answer.
  + Consider the following expression
    - x extends y

For which of the following possibilities of x and y the above expression is valid?

1. Both x and y should be classes.
2. Both x and y should be interfaces.
3. Both x and y can be either classes or interfaces. // Valid
4. No restrictions.
   * + X extends Y, Z

X, Y, Z should be an interface.

* + - X implements Y, Z

X 🡪 Class

Y, Z 🡪 Interface

* + - X extends Y implements Z

X, Y 🡪 classes

Z 🡪 Interface

* + - X implements Y extends Z

CE: Because we have to take extends first followed by interface.