Mahindra
École Centrale
COLLEGE OF ENGINEERING

Object Oriented Programming
JAVA

Dr. Prafulla Kalapatapu
Computer Science Engineering
Mahindra Ecole Centrale
prafulla.kalapatapu@mechyd.ac.in

LEADER ■ ENTREPRENEUR ■ INNOVATOR

Mahindra
École Centrale
COLLEGE OF ENGINEERING

Interface

Interface

- What is an interface

  - An interface is a specification of method prototype.

  - All methods of the interfaces are public and abstract.

- Why the methods of interface are public and abstract by default.

  - Interface methods are public since they should be available to third party vendors to provide implementation.

  - They are abstract because their implementation is left for third party vendor.

Declaration & Implementation of an interface

Mahindra École Centrale
COLLEGE OF ENGINEERING

- We can declare an interface by using “interface” keyword.
- We can implement an interface by using “implements” keyword

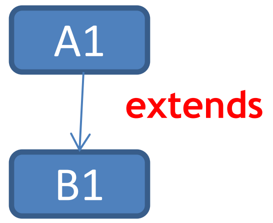
Ex:
```
interface ABC
{
    void m1();    // public,abstract
    void m2();
}
abstract class Sample implements ABC
{
    public void m1() { … }
}
```

- If a class implements an interface, compulsory we should provide implementation for every method of that interface. Otherwise we have to declare class as abstract. Violation leads to “compile-time error”.

- Whenever we are implementing an interface method, compulsory it should be declared as public otherwise we get compile time error

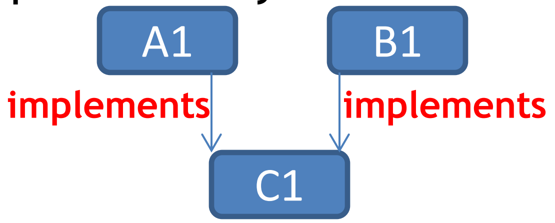
extends Vs implements

- A class extends only one class at a time.



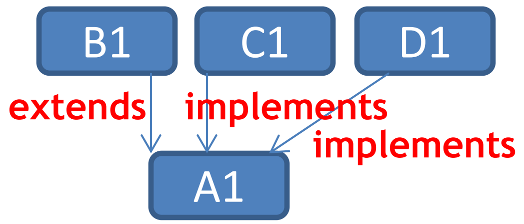
A1, B1 are classes

- A class can implement any no. of interfaces.



A1, B1 are interfaces

- A class can extend a class and can implement any no. of interfaces simultaneously.



A1, B1 are classes
C1, D1 are interfaces

- An interface can extend any no. of interfaces at a time.

Ex:

```
interface A1
{
}
interface B1
{
}
interface C1 extends A1, B1
{
}
```

Interface Methods

- Whether we are declaring or not every interface method is by default public and abstract.

Ex:

```
interface ABC
{
        void m1();    // by default, it is public, abstract
}
```

public: To make this method available for every implementation class

abstract: Because interface methods specifies requirements but not implementation

Valid method declarations inside interfaces

- void m1();
- public void m1();
- abstract void m1();
- public abstract void m1();

Interface variables

- An interface can contain variables, the main purpose of these variables is to specify constants at requirement level.

Ex:

```
interface ABC
{
        int x=10; // by default , public, static, final
}
```

public : To make this variable available for every implementation class

static : without existing object also, implementation class can access this variable.

final : implementation can access this variable but cant modify

Valid variable declarations inside interfaces

Mahindra École Centrale
COLLEGE OF ENGINEERING

- int x=10;
- public int x=10;
- public static int x=10;
- static public int x=10;
- public static final int x=10;
- final int x=10;
- public final int x=10;
- static final int x=10;

Interface Naming conflicts

1. Method naming conflicts:

(i) Case 1 : if two interfaces contains a method with same signature and same return type in the implementation class we can provide implementation for only one method.

```
interface I
{
        void m1();
}
interface U
{
        void m1();
}
class Sample implements I, U
{
        public void m1() { … }
}
```

(ii) Case 2 : if two interfaces contains a method with same name but different args then, in the implementation class we have to provide implementation for both methods and these methods are considered as overload methods.

```
interface I
{
        void m1();
}
interface U
{
        void m1(int x);
}
class Sample implements I, U
{
        public void m1() { … }
        public void m1(int x) { … }
}
```

(iii) Case 3 : if two interfaces contains a method with same signature but different return types, then it is impossible to implement both interfaces at a time.

```
interface I
{
        void m1();
}
interface U
{
        int m1();
}
```

- A java class can implement any no. of interfaces simultaneously, Is it possible ?

yes, except two interfaces contains a method with same signature but different return types.

2. Variable naming conflicts :

```
interface I
{
        int x=10;
}
interface J
{
        int x=20;
}
class Test implements I, J
{
        public static void main(String… a)
        {
                System.out.println(x);      // CE : reference to x is ambigious
        }
}
```

- There may be a chance of 2 interfaces contains a variable with same name and may rise variable naming conflicts. But we can resolve these naming conflicts by using interface names.

- In previous example

```
System.out.println(I.x);  // 10
System.out.println(J.x);  // 20
```

abstract class Vs interface

Mahindra École Centrale
COLLEGE OF ENGINEERING

abstract class	interface
It is declared using “abstract” keyword	It is declared using “interface” keyword
It can have combination of instance variables and static variables	It can contain only static final variables
It can contain abstract methods and concrete methods	It contains only abstract methods
When an abstract class is written, it is the duty of programmer to provide sub class to it	An interface is written, when the programmer wants to leave the implementation to the third party vendors
abstract class can have constructors	interface cant have constructors
Inside abstract class we can write static block and instance block	In interface we cant write static block and instance block