BITS Pilani
Pilani Campus

Object Oriented Programming CS F213

J. Jennifer Ranjani
email: jennifer.ranjani@pilani.bits-pilani.ac.in
Chamber: 6121 B, NAB
Consultation: Appointment by e-mail

-Interfaces
-Nested Interfaces

BITS Pilani
Pilani Campus

innovate achieve lead

BITS Pilani
Pilani Campus

Interfaces

Default Methods in Interface (defender or virtual extension)

- Before Java 8, interfaces could have only abstract methods. Implementation is provided in a separate class

- If a new method is to be added in an interface, implementation code has to be provided in all the classes implementing the interface.

- To overcome this, default methods are introduced which allow the interfaces to have methods with implementation without affecting the classes.

Default Methods

```
interface Printable{
void print();
default void show()
{
System.out.println("Within Show");
}
}

class trial implements Printable {

public void print()
{
System.out.println("Within Print");
}
}
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.print();
t.show();
}


}
```

Static Methods in Interfaces

```
interface Printable{
void print();
static void show()
{
System.out.println("Within
    Printable Show");
}
}

class trial implements Printable {
public void print()
{
System.out.println("Within Print");
}
}
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.print();
Printable.show();
}
}
```

Question:
Can we replace Printable.show() with t.show()?

Static Methods in Interfaces

```
interface Printable{
void print();
static void show()
{
System.out.println("Within
    Printable Show");
}
}
class trial implements Printable {
public void print() {
System.out.println("Within Print");}
static void display()
{
System.out.println("Within
    Display");
}}
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.print();
t.display();  //Warning
t.show();     //Error
 }
}
```

Default Methods & Multiple Inheritance

```
interface Printable{
void print();
default void show()
{
System.out.println("Within
    Printable Show");
}
}
interface Showable{
default void show()
{
System.out.println("Within
    Showable Show");
}
void print();
}
```

```
class trial implements Printable,Showable{
public void show()  {
Printable.super.show();
Showable.super.show(); }

public void print() {
System.out.println("Within Print"); }}

public class test {
public static void main(String[] args) {
trial t = new trial();
t.print();
t.show();
}
}
```

Question:
What happens if super keyword is omitted?

Nested Interfaces

- Interface can be declared within another interface or class
- Nested interface cant be accessed directly, it is referred by the outer interface or class
- Nested interface must be public if it is declared inside the interface but it can have any access modifier if declared within the class.
- Nested interfaces are declared static implicitly.

Class Implementing Outer Interface

```
interface Printable{
void print();
interface Showable{
void show(); }
}

class trial implements Printable {
public void print()
{
System.out.println("Within Print");
}
public void show() {
System.out.println("Within Show");
}
}
```

```
public class test {
public static void main(String[]
     args) {
trial t = new trial();
t.print();
t.show();
}
}
```

Question:
What happens when implementation of show() is removed from class trial?

Class Implementing Outer Interface

```
interface Printable{
void print();
interface Showable{
void show(); }
}

class trial implements Printable {
public void print()
{
System.out.println("Within Print");
}
}
```

```
public class test {
public static void main(String[]
      args) {
trial t = new trial();
t.print();
}
}
```

Answer: Nothing happens. Outer interface does not have access to inner interface.

Class Implementing Inner Interface

```
interface Printable{
void print();
interface Showable{
void show();}
}
class trial implements
    Printable.Showable {
public void print1()
{
System.out.println("Within Print");
}
public void show()
{
System.out.println("Within Show");
}
}
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.print1();
t.show();
}

}
```

Note: If we omit the implementation of show() method, we get compilation error

Interface within the Class

```
class Printable{
public void print()
{
System.out.println("Within Print");
}
interface Showable{
void show();}
}

class trial implements
    Printable.Showable {
public void show()
{
System.out.println("Within Show");
} }
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.show();
t.print();   //print undefined for the type
    trail
}
}
```

Interface within the Class

```
class Printable{
public void print()
{
System.out.println("Within Print");
}
interface Showable{
void show();}
}

class trial extends Printable
    implements Printable.Showable
    {
public void show()
{
System.out.println("Within Show");
} }
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.show();
t.print();
}
}
```

Output:

```
Within Show
Within Print
```

Class within the Interface

```
interface Showable{
class Printable{
public void print()
{
System.out.println("Within Print");
}}
void show();
}

class trial extends
    Showable.Printable {
public void show()
{
System.out.println("Within Show");
} }
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.show();
t.print();
}
}
```

Output:

```
Within Show
Within Print
```

innovate achieve lead

Class within the Interface

```
interface Showable{
class Printable{
public void print()
{
System.out.println("Within Print");
}}
void show1();
}
class trial extends
    Showable.Printable implements
    Showable {
public void show()
{
System.out.println("Within Show");
} }
```

```
public class test {
public static void main(String[]
    args) {
trial t = new trial();
t.show();
t.print();
}
}
```

Error:
Class trial should implement the method show1()

What happens if the class does not implement all members of the Interface?

```
interface Printable{
void print();
void show();
}
abstract class trial implements Printable {
public void print() {
System.out.println("Within Print");
}}

public class test {
public static void main(String[] args) {
trial t = new trial();
t.print();}
}
```

Error:
Cannot Instantiate trial

(Because trail is an abstract class)

What happens if the class does not implement all members of the Interface?

```
interface Printable{
void print();
void show(); }

abstract class trial implements Printable {
public void print() {
System.out.println("Within Print");}
}

public class test extends trial {
public void show() {
System.out.println("Within Show");}
public static void main(String[] args) {
test t = new test();
t.print();
t.show();}}
```

Output:

```
Within Show
Within Print
```