Lecture 24 Default Methods Inside Interfaces:

Until Java 1.7:

* Interface had only public and abstract method present inside interface.
* Variables where always public static and final

From Java 1.8 versions:

* We can define concrete methods inside Interface. These methods which we define inside interface is called default methods.

How to declare default methods.

* default keyword is defined as follows.

**interface** Interf1{

**default** **void** m1() {

System.***out***.println("Default method");

}

}

* The implementation class gets this default implementation automatically. If the implementation class wants it can override the default method.

**package** com.durgaSoft.section2.lecture24;

**public** **class** Example1 {

**public** **static** **void** main(String[] args) {

Test test = **new** Test();

test.m1();

}

}

**class** Test **implements** Interf1 {

}

**interface** Interf1 {

**default** **void** m1() {

System.***out***.println("Default method");

}

}

* The implementation class can override the default method.

**package** com.durgaSoft.section2.lecture24;

**public** **class** Example2 {

**public** **static** **void** main(String[] args) {

Test2 test = **new** Test2();

test.m1();

}

}

**class** Test2 **implements** Interf2 {

@Override

**public** **void** m1() {

// **TODO** Auto-generated method stub

System.***out***.println("We are in Test2 m1 method");

}

}

**interface** Interf2 {

**default** **void** m1() {

System.***out***.println("Default method");

}

}

Lecture 25 Differences between Interface with Default Methods and Abstract Class:

Default methods with Multiple inheritance: Two Interfaces are having the same methods and when the class Test is implementing these two interfaces it will throw compilation errors. Due to ambiguity issue.

**interface** Left3{

**default** **void** m1() {

System.***out***.println("Left Default method");

}

}

**interface** Right3{

**default** **void** m1() {

System.***out***.println("Right Default method");

}

}

**class** Test **implements** Left3, Right3{

}

C.E: Class Test inherits unrelated defaults for Left and Right,

This ambiguity problem can be overcome by overriding the default method as shown below:

**package** com.durgaSoft.section2.lecture24;

**public** **class** Example4 {

**public** **static** **void** main(String[] args) {

Test4 test4 = **new** Test4();

test4.m1();

}

}

**interface** Left4{

**default** **void** m1() {

System.***out***.println("Left Default method");

}

}

**interface** Right4{

**default** **void** m1() {

System.***out***.println("Right Default method");

}

}

**class** Test4 **implements** Left4 ,Right4{

**public** **void** m1() {

System.***out***.println("My own Implementation");

}

}

When you want to just use one of the default method:

**package** com.durgaSoft.section2.lecture24;

**public** **class** Example5 {

**public** **static** **void** main(String[] args) {

Test5 test5 = **new** Test5();

test5.m1();

}

}

**interface** Left5{

**default** **void** m1() {

System.***out***.println("Left Default method");

}

}

**interface** Right5{

**default** **void** m1() {

System.***out***.println("Right Default method");

}

}

**class** Test5 **implements** Left5 ,Right5{

**public** **void** m1() {

System.***out***.println("My own Implementation");

Right5.**super**.m1();

}

}

Difference between interface with default methods and abstract methods

|  |  |  |
| --- | --- | --- |
| SI | Interface with default method | Abstract Class |
|  |  |  |
| 1 | Inside interface every variable is always public static final we cannot declare instance variables | Inside Abstract class we can declare instance variable, which are required to the child class |
|  |  |  |
| 2 | Interface never talks about state of Objects | Abstract Class can talk about state of Object |
|  |  |  |
| 3 | Inside Interface we can’t declre constructors | Inside Abstract Class we can declare Constructors |
|  |  |  |
| 4 | Inside Interface we can’t declare instance and static blocks | Inside Abstract Class we can declare instance and static blocks |
|  |  |  |
| 5 | Functional interface with default method can refer lambda expression | Abstract class can’t refer Lambda expression. As referring lambda expression means implementing an interface. |
|  |  |  |
| 6 | Inside interface we can’t override Object class methods as default method | Inside Abstract class we can override Object class methods. |

Lecture 26: Static methods inside Interfaces:

Static Methods inside Interface:

* Static methods in interface are used to define Utility methods.
* They can be called only using Interface name only.
* If these methods are called using Implementation class name or Implementation Object reference then you will get Compilation Error.

Compilation Error: Cannot find Symbol.

Interface Static methods with respect to Overriding.

Interface static methods are not available to the implementation class so there is no overriding at all

Case1: Here we are overriding the method as static method only.

**interface** Interf261{

**public** **static** **void** m1() {

System.***out***.println("Static method of Interf261");

}

}

**class** Test **implements** Interf261{

**public** **static** **void** m1() {

System.***out***.println("Static method inside Test class ");

}

}

Case2: Here we are overriding the method as non static method only.

**interface** Interf261{

**public** **static** **void** m1() {

System.***out***.println("Static method of Interf261");

}

}

**class** Test **implements** Interf261{

**public** **void** m1() {

System.***out***.println("Non Static method inside Test class");

}

}

Case 3: We are overriding the method as private:

**interface** Interf261{

**public** **static** **void** m1() {

System.***out***.println("Static method of Interf261");

}

}

**class** Test **implements** Interf261{

**private** **static** **void** m1() {

System.***out***.println("Static method of Interf261");

}

}

We can declare main Method inside interface and Run the class too..

**package** com.durgaSoft.section2.lecture26;

**public** **interface** CheckInterf {

**public** **static** **void** main(String[] args) {

System.***out***.println("We are here In Interface Static method");

}

}