CourseCube®
(Formerly Java Learning Center)

Java 8
New Features

Author
Srinivas Dande

Java Learning Center

CourseCube®
(Formerly Java Learning Center)

CourseCube®
(Formerly Java Learning Center)

1. Introduction

- Java 8 is a revolutionary release of the **World's #1 development platform**.
- It includes a huge upgrade to the Java programming model and a coordinated evolution of the JVM, Java language, and libraries.
- Java 8 includes features for productivity, ease of use, security and improved performance.
- Welcome to the latest iteration of the largest, open, standards-based, community-driven platform.

- Java8 Release comes with Many new features as Listed

1) Default Methods in Interface
2) Static Methods in Interface
3) Lambda Expressions
4) Method References
5) Functional Interfaces
6) Functional Programming
7) Streams API
8) Joda Date API
9) Optional Class
10) Miscellaneous Features

1. Default Methods in Interface

- **Concrete methods (Methods with Body)** Defined in the Interface with **default keyword** are called as **Default Methods**.
- Default methods are also known as **defender methods** or **virtual extension methods**
- Default Methods are public by default.
- Default Methods will be inherited to Sub classes.
- Sub class can override the Interface Default Methods.
- We can't write default methods inside a class. Even when we are overriding the default method in sub class, we should not use default keyword for sub class method.
- We can't override Object class methods as default methods inside Interface

Why Default Methods:

- We can define new functionality in the interfaces without breaking down the implementing classes
- We can avoid writing separate utility classes

Demo1: Files Required:

1. Animal.java	2. Dog.java
3. Cat.java	4. Demo1.java

1)Animal.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface Animal {
	public abstract void eating();
	public abstract void sleeping();

	default void running() {
		System.out.println("Animal is Running");
	}

	default void thinking() {
		System.out.println("Animal is Thinking");
	}
}
```

2)Dog.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Dog  implements Animal{

	@Override
	public void eating() {
		System.out.println("Dog is eating");
	}
	@Override
	public void sleeping() {
		System.out.println("Dog is sleeping");
	}
	@Override
	public void running() {
		System.out.println("Dog is running");
	}
}
```

3)Cat.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Cat  implements Animal{

	@Override
	public void eating() {
		System.out.println("Cat is eating");
	}
	@Override
	public void sleeping() {
		System.out.println("Cat is sleeping");
	}
	@Override
	public void thinking() {
		System.out.println("Cat is thinking");
	}
}
```

4)Demo1.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo1 {
        public static void main(String[] args) {

                Dog mydog= new Dog();
                mydog.eating();  //Overriden method
                mydog.sleeping();  //Overriden method
                mydog.running();  //Overriden method
                mydog.thinking();  //Inherited default method

                Cat mycat= new Cat();
                mycat.eating();  //Overriden method
                mycat.sleeping();  //Overriden method
                mycat.running();  //Inherited default method
                mycat.thinking();   //Overriden method
        }
}
```

Demo2: Files Required:

1. A.java	2. B.java
3. Hello.java	4. Demo2.java

1)A.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface A {

        default void show() {
                System.out.println("A- show() ");
        }
}
```

2)B.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface B {

	default void show() {
		System.out.println("B- show()");
	}
}
```

3)Hello.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Hello  implements A,B{

	@Override
	public  void show() {
		System.out.println("Hello- show() ");
	}
	public void test() {
		System.out.println("Hello- test() ");
		show();
		A.super.show();
		B.super.show();
	}
}
```

4)Demo1.java

```
package com.jlcindia.demo2;

public class Demo2 {
	public static void main(String[] args) {
	Hello hello=new Hello();
	hello.test();
	}
}
```

Demo3: Files Required:

1. A.java	2. Hello.java
3. Demo3.java	

1)A.java

```
package com.jlcindia.demo3;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface A {

        default void m1() {
                System.out.println("A- m1() ");
        }

        default void m2() {
                System.out.println("A- m2() ");
                m1();
        }

        /*
        default boolean equals(Object obj) {
                System.out.println("A- equuals() ");
        }
        */
}
```

2)Hello.java

```
package com.jlcindia.demo3;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Hello  implements A{

}
```

3)Demo3.java

```
package com.jlcindia.demo3;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo3 {
        public static void main(String[] args) {
                Hello hello=new Hello();
                hello.m1();
                hello.m2();
        }
}
```

Demo4: Files Required:

1. A.java	2. B.java
3. Hello.java	4. Demo4.java

1)A.java

```
package com.jlcindia.demo4;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface A {
        default void m1() {
                System.out.println("A- m1() ");
        }
}
```

2)B.java

```
package com.jlcindia.demo4;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface B extends A {
        default void m2() {
                System.out.println("B- m2() ");
                m1();
        }
}
```

3)Hello.java

```
package com.jlcindia.demo4;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Hello  implements B{

}
```

4)Demo4.java

```
package com.jlcindia.demo4;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo4 {
        public static void main(String[] args) {
                Hello hello=new Hello();
                hello.m1();
                hello.m2();
        }
}
```

Demo5: Files Required:

1. A.java	2. B.java
3. Hello.java	4. Demo4.java

1)A.java

```
package com.jlcindia.demo5;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface A {
        default void m1() {
                System.out.println("A- m1() ");
        }
}
```

2)B.java

```
package com.jlcindia.demo5;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface B extends A {

	default void m1() {
		System.out.println("B- m1() ");
	}

	default void m2() {
		System.out.println("B- m2() ");
		m1();
	}
}
```

3)Hello.java

```
package com.jlcindia.demo5;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Hello  implements B{

}
```

4)Demo4.java

```
package com.jlcindia.demo5;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo4 {
	public static void main(String[] args) {
		Hello hello=new Hello();
		hello.m1();
		hello.m2();
	}
}
```

Interview Questions:

Q1) What are Interface Default Methods?

Ans:

Q2) Will Default Methods be inherited to Sub Class?

Ans:

Q3) Can I Override the Default Methods in Sub Class?

Ans:

Q4) Can I mark the Default Methods as Protected?

Ans:

Q5) Can I mark the Regular Java Class Methods as Default?

Ans:

Q6) Can I mark the Overriden Default Methods as Default in the Sub Class?

Ans:

Q7) Can I Override Object class methods as Default Methods inside Interface?

Ans:

Q8) Why we need Default Methods inside Interface?

Ans:

Q9) What happens when Sub Class is implementing two interfaces which are having same default method?

Ans:

Q10) How Can I access Default Methods inside Sub Class?

Ans:

Q11) Can I have multiple Default Methods in interface?

Ans:

Q12) Can I call one Default Method from another Default Methods of same Interface?

Ans:

Q13) Can I Override Interface Default Method extended from other Interface?

Ans:

2. Static Methods in Interface

- **Concrete methods (Methods with Body)** Defined in the Interface with **static keyword** are called as **Static Methods.**
- Static Methods are public by default.
- Static Methods will not be inherited to Sub classes.
- Sub class can not override the Interface Static Methods.
- If we write the Static Method of Interface in Sub Class then That will be treated as New Method in Sub Class.

Why Static Methods:

- We can avoid writing separate utility classes

Demo1: Files Required:

1. A.java	2. Hello.java
3. Demo1.java	

1)A.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface A {

        int P=101;
        public final static int Q=102;

        void m1();
        public abstract void m2();

         default void m3() {
                System.out.println("A - m3()");
        }

         default void m4() {
                System.out.println("A - m4()");
        }

         static void m5() {
                System.out.println("A - m5()");
        }
```

```
          static void m6() {
                    System.out.println("A - m6()");
          }
}
```

2)Hello.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Hello implements A {

          public void test(){

                    System.out.println(P);  //Inherited
                    System.out.println(Q); //Inherited
                    m1();    //Overriden
                    m2();   //Overriden

                    m3();     //Overriden
                    A.super.m3();

                    m4();   //Inherited
                    A.super.m4();

                    A.m5();
                    A.m6();
                    //A.super.m6();
          }

          @Override
          public void m1() {
                    System.out.println("Hello -m1");
          }

          @Override
          public void m2() {
                    System.out.println("Hello -m2");
          }
          @Override
          public void m3() {
                    System.out.println("Hello -m3");

          }
          /*
          @Override
          public static void m5() {
                    System.out.println("Hello -m5");
          }
          */
}
```

3)Demo1.java

```
package com.jlcindia.demo1;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo1 {
          public static void main(String[] args) {
                    Hello hello=new Hello();
                    hello.test();
          }
}
```

Demo2: Files Required:

1. A.java	2. B.java
3. Hello.java	4. Demo2.java

1)A.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface A {
           static void m1() {
                    System.out.println("A - m1()");
          }
}
```

2)B.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface B {

        static void m1() {
                System.out.println("B - m1()");
        }

}
```

3)Hello.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Hello implements A,B {

        public void test(){

                m1();
                A.m1();
                B.m1();
        }

         static void m1() {
                        System.out.println("Hello- m1()");
        }

         static void show() {
                        System.out.println("Hello- show()");
        }
}
```

4)Demo2.java

```
package com.jlcindia.demo2;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public class Demo2 {

            public static void main(String[] args) {

            Hello hello=new Hello();
            hello.test();

            //1. Calling Static Method with Ref. Variable having Null
           // A aobj = null;
           //aobj.m1();

            Hello hello1=null;
            hello1.show();


            //2. Calling Static Method with Ref. Variable having Object address
             //A aobj = new Hello();
            //aobj.m1();

            Hello hello2=new Hello();
            hello2.show();

            //3. Calling Static Method with Class Name
            A.m1();
            Hello.show();

          // Interface Static Methods must called with Interface name always
            }

}
```

Demo3: Files Required:

1. A.java	

1)A.java

```
package com.jlcindia.demo3;
/*
* @Author : Srinivas Dande
* @Company: Java Learning Center
* */
public interface A {

        static void m1() {
                System.out.println("A - m1()");
                //m2();   // Can not call Instance Method in Static
        }

        default void m2() {
                        System.out.println("A - m2()");
        }

        public static void main(String[] args) { //Standard Main Method

                System.out.println("main method");
                m1();
                //m2();   // Can not call Instance Method in Static

        }

}
```

Interview Questions:

Q1) What are Interface Static Methods?

Ans:

Q2) Will Static Methods be inherited to Sub Class?

Ans:

Q3) Can I Override the Static Methods in Sub Class?

Ans:

Q4) Can I mark the Static Methods as Protected?

Ans:

Q5) Can I mark the Regular Java Class Methods as Static?

Ans:

Q6) Why we need Static Methods inside Interface?

Ans:

Q7) What happens when Sub Class is implementing two interfaces which are having same Static method?

Ans:

Q8) How Can I access Static Methods inside Sub Class?

Ans:

Q9) Can I have multiple Static Methods in Interface?

Ans:

Q10) Can I call one Static Method from another Static Methods of same Interface?

Ans:

Q11) Can I call one Static Method from another Default Methods of same Interface?

Ans:

Q12) Can I call one Default Method from another Static Methods of same Interface?

Ans:

Q13) Can I call Write Standard Main Method in Interface?

Ans:

Q14) Can I mark Default Method as Static?

Ans:

Q15) Can I mark Default Method as Abstract?

Ans:

Q16) Can I mark Static Method as Abstract?

Ans: