Key points

- Prior to java 8,interfae in java can only have abstract methods.
- All the methods of interfaces are public & abstract by default.
- Java 8 allows the interfaces to have default and static methods. The reason we have default methods in interfaces is to allow the developers to add new methods to the interfaces without affecting the classes that implements these interfaces.
- **Static** methods in interfaces are similar to the **default** methods except that we cannot override these methods in the classes that implements these interfaces.
- Java 8 interface are **Backward compatibility.**[Backward compatibility is adding new features without breaking the old code.]

Java 8 Interface Changes – default method and static method

Why these changes are required??

If classes implements an interface then if we add a new method to the interface, we have to change the code in all the classes that implements this interface.

Example : Consider interface X , class A,B and C

```
public interface X {
   //by default interface methods are public
   void print();
   void save();
}
```

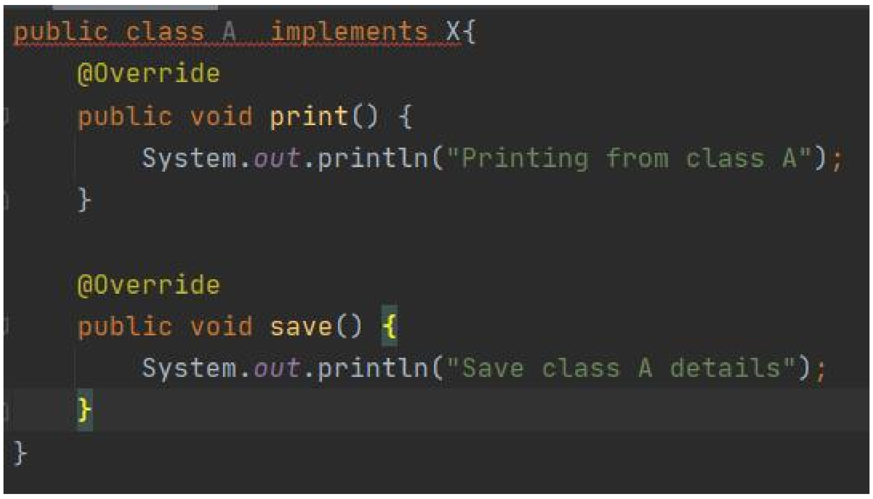
```
public class A  implements X{
   @Override
   public void print() {
      System.out.println("Printing from class A");
   }

   @Override
   public void save() {
      System.out.println("Save class A details");
   }
}
```

```
public class B implements X{
   @Override
   public void print() {
      System.out.println("Printing from class B");
   }

   @Override
   public void save() {
      System.out.println("Save class B");
   }
}
```

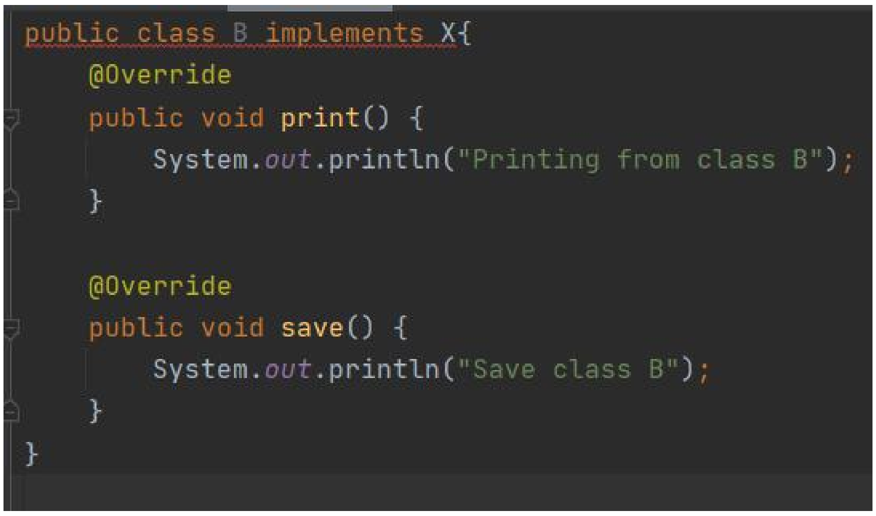
If we try to add one more method to interface X , we need to make sure all inherited classes/implemented classes should also implement those methods

Add view method to interface X

```
public interface X {
   //by default interface methods are public
   void print();
   void save();
   void view();
}
```



```
public class A  implements X{
    @Override
    public void print() {
        System.out.println("Printing from class A");
    }

    @Override
    public void save() {
        System.out.println("Save class A details");
    }
}
```



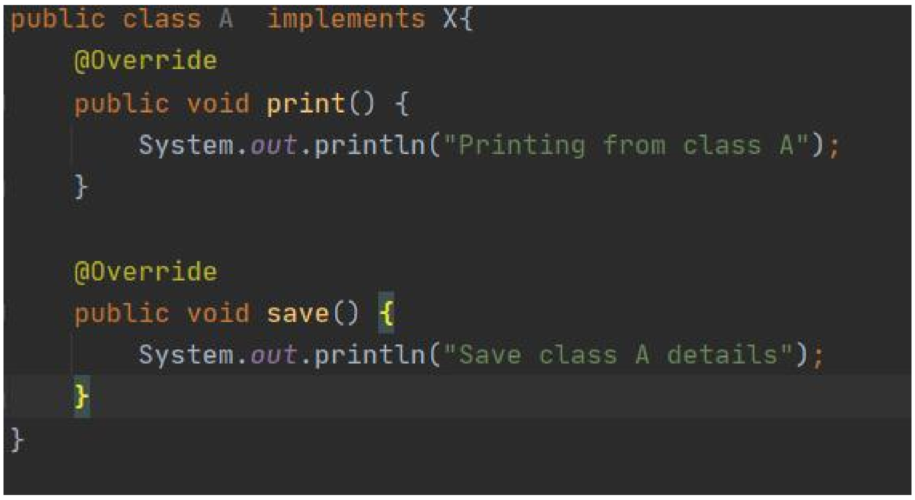
```
public class B implements X{
    @Override
    public void print() {
        System.out.println("Printing from class B");
    }

    @Override
    public void save() {
        System.out.println("Save class B");
    }
}
```

In above example class A and class B throws compile time exception because view methods is not implemented as view is abstract in scope.
This may create a big problem when we are working with "N" number of classes which inherits or implements interface where we keep on changing the methods based on new requirements.
So to over come from this problem java 8 features updated with interface changes like **default method** and **static** method **.** These **static** and **default** methods must have method body /method definition, So that classes which are implementing interface can implement those methods if and only if it is required else those newly added changes will not impact on inherited classes.

"implementation is not required for **static** and **default** methods"

Let’s take above example and try to add default method

```java
public interface X {
   //by default interface methods are public
   void print();
   void save();
   default void view()
   {
      System.out.println("Hai I am default for this interface");
   }
}
```



```java
public class A  implements X{
    @Override
    public void print() {
        System.out.println("Printing from class A");
    }

    @Override
    public void save() {
        System.out.println("Save class A details");
    }
}
```

```java
public class B implements X{
   @Override
   public void print() {
      System.out.println("Printing from class B");
   }

   @Override
   public void save() {
      System.out.println("Save class B");
   }
}
```

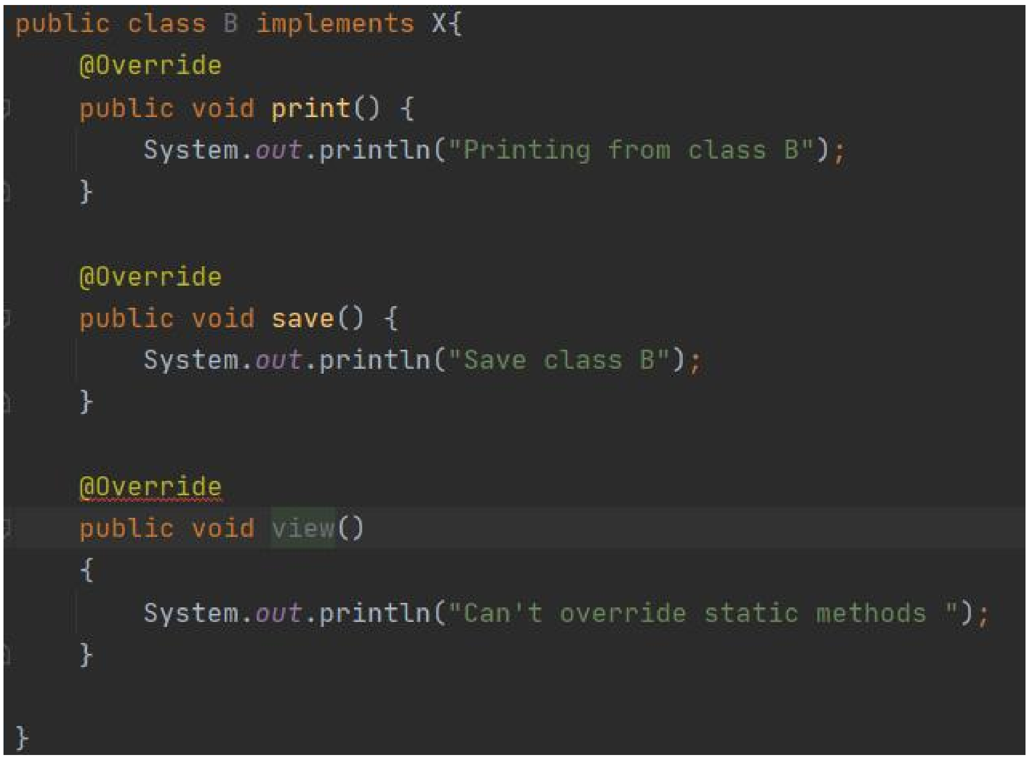
```java
public class B implements X{
   @Override
   public void print() {
      System.out.println("Printing from class B");
   }

   @Override
   public void save() {
      System.out.println("Save class B");
   }

   @Override
   public void view() {
      System.out.println("can be override default methods ");
   }
}
```

```java
public class A  implements X{
   @Override
   public void print() {
      System.out.println("Printing from class A");
   }

   @Override
   public void save() {
      System.out.println("Save class A details");
   }

   public void callDefaultMethod()
   {
      X.super.view();
   }
}
```

Let’s take above example and try to add static method

```java
public interface X {
   //by default interface methods are public
   void print();
   void save();
   static void view()
   {
      System.out.println("Hai I am default for this interface");
   }
}
```

```java
public class A  implements X{
   @Override
   public void print() {
      System.out.println("Printing from class A");
   }

   @Override
   public void save() {
      X.view();
      System.out.println("Save class A details");
   }
}
```



```java
public class B implements X{
    @Override
    public void print() {
        System.out.println("Printing from class B");
    }

    @Override
    public void save() {
        System.out.println("Save class B");
    }

    @Override
    public void view()
    {
        System.out.println("Can't override static methods ");
    }

}
```