Functional Programming (Default methods)

B.Tech. (IT), Sem-6,
Applied Design Patterns and Application Frameworks (ADPAF)

Dharmsinh Desai University
Prof. H B Prajapati

Topics

- Lambda Expression
- Default methods
- Using lambdas in design patterns

Traditional Java interfaces

- Interfaces in Java always had method declaration (only signature, no implementation)
- Java did not allow multiple inheritance of classes
  - Due to diamond problem

Default methods

- In Java 8, it is now possible to add method bodies into interfaces
- Example

```
public interface MathOperation{
    int add (int a,int b);
    default int multiply(int a, int b){
        return a*b;
    }
}
```


Why default methods

- Before Java 8, once we declare a method in interface, all its implementing classes must define method body of this newly declared method.
- A new feature called Lambda is introduced in Java 8
  - It is not possible to use this feature in existing Java libraries such as java.util.List without adding methods in existing List
  - But, adding a single method in List can break the existing code.
- Therefore, for backward compatibility, Java 8 introduced concept of default methods.

Example-Multilevel inheritance

- Interface1

```
package ambiguity.multilevel;
public interface Interface1 {
   default void hello(){
      System.out.println("Interface1's hello()");
   }
}
```

- Interface2 inherits Interface1

```
package ambiguity.multilevel;
public interface Interface2 extends Interface1{
   @Override
   default void hello(){
      System.out.println("Interface2's hello()");
   }
}
```

Example-Multilevel inheritance

- InterfaceImpl (which hello() method it gets)

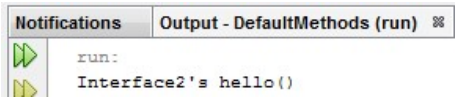
```
package ambiguity.multilevel;
public class InterfaceImpl implements Interface2{
}
```

- Call hello() method on object of the class

```
package ambiguity.multilevel;
public class Test {
   public static void main(String[] args) {
      InterfaceImpl implObj=new InterfaceImpl();
      implObj.hello();
   }
}
```

7

Running the Example-Multilevel inheritance



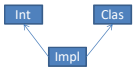
- If a class inherits the same method from different interfaces?
  - sub-interfaces shadow super-interfaces

8

Example: Default methods with interface inheritance and class inheritance

- BaseInterface having hello() method

```
package ambiguity.dominance;
public interface BaseInterface {
   default void hello(){
      System.out.println("BaseInterface's hello() method");
   }
}
```



- BaseClass having hello() method

```
package ambiguity.dominance;
public class BaseClass {
   public void hello(){
      System.out.println("BaseClass's hello() method");
   }
}
```

9

Example: Default methods with interface inheritance and class inheritance

- A class inherits from interface as well as class

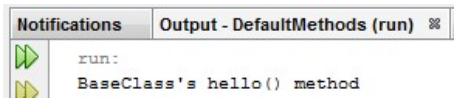
```
package ambiguity.dominance;
public class DerivedClass extends BaseClass implements BaseInterface{
}
```

- Which hello() method gets called?

```
package ambiguity.dominance;
public class Test {
   public static void main(String[] args) {
      DerivedClass dcObj=new DerivedClass();
      dcObj.hello();
   }
}
```

10

Running the Example: Default methods with interface inheritance and class inheritance

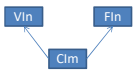


- If a class inherits the same method from a class and an interface?
  - extends dominates implements
  - sub-class inherits super-class's method (not interface's method)

11

Example: Default methods with multiple inheritance

- Interface Vehicle




```
package defaultmethods;
public interface Vehicle {
   default void drive(){
      System.out.println("Drive me: I am a vehicle!");
   }
   static void blowHorn(){
      System.out.println("Blowing horn!!!");
   }
}
```

12

Example: Default methods with multiple inheritance

- Interface FourWheeler

```
package defaultmethods;
public interface FourWheeler {
   default void drive(){
      System.out.println("Drive me: I am a four wheeler!");
   }
}
```


Example: Default methods with multiple inheritance

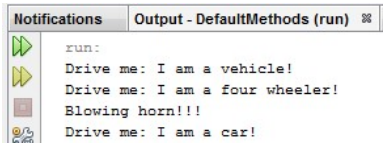
- Car class implementing two interfaces

```
package defaultmethods;
public class Car implements Vehicle, FourWheeler {
   @Override
   public void drive(){
      Vehicle.super.drive(); // Call drive() of Vehicle interface
      FourWheeler.super.drive(); // Call drive() of FourWheeler interface
      Vehicle.blowHorn();
      System.out.println("Drive me: I am a car!");
   }
}
```


Example: Default methods with multiple inheritance

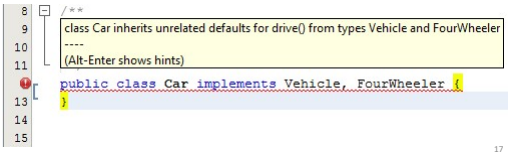
```
package defaultmethods;
public class Test {
   public static void main(String[] args) {
      Vehicle vehicle = new Car();
      vehicle.drive();
   }
}
```


Running the Example: Default methods with multiple inheritance



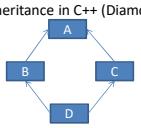
Running the Example: Default methods with multiple inheritance

- If a class inherits the same method from different interfaces?
  - if the interfaces are unrelated
    - results in a compile error



Multiple inheritance in C++ versus Java 8

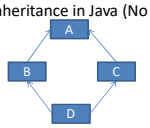
- Multiple inheritance in C++ (Diamond problem)



- How is A's state (i.e., its data members) inherited to D ?
  - once, or
  - twice (via B and via C)
- There is no right answer to this question
  - C++ gives you the option: virtual / non-virtual inheritance
  - But, it makes it more complicated

Multiple inheritance in C++ versus Java 8

- Multiple inheritance in Java (No diamond problem)



- A can only be an interface (not a class)
  - A can have an implementation, but
  - no state (no instance fields)
- No state means no (diamond) problem !
  - No issue regarding "how many A parts does D have ?

19

References

- Java 8 in Action, Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft, Manning Publications
- Java 8 Lambda Expressions & Streams, by Adib Saikali, Video, The San Francisco Java User Group

20