Object Oriented Programming
CS F213
Amit Dua

BITS Pilani
Pilani Campus

Slides Taken from the slides prepared by Dr. Jennifer

Questions from prev. class

- What is the actual use of dynamic method dispatch?
- Child c1 = new parent(); //allowed ??
- How can we access the hidden instance variables from the reference of parent class storing the child object?
- Can we access the parent variables from child class object?
- Can we do the same with the private parent instance variables/methods?

Private methods

```
class Base {
  private void fun() {
     System.out.println("Base
    fun");
  }
}

class Derived extends Base {
  private void fun() {
     System.out.println("Derived
    fun");
  }  }
```

```
Class Main {
public static void main(String[ ]
args) {
      Base obj = new Derived();
      obj.fun();
  }
}
```

error: fun() has private access in Base

With static methods?

```
class Base {
  static void fun() {
     System.out.println("Base
    fun");
  }
}

class Derived extends Base {
  static void fun() {
     System.out.println("Derived
    fun");
  }  }
```

```
Class Main {
public static void main(String[ ]
args) {
      Base obj = new Derived();
      obj.fun();
  }
}
```

Base fun

Example abstract class

```
abstract class Base
{
    final void fun()
{System.out.println("Der
   ived fun() called"); }
}
class Derived extends
   Base { }
```

```
class Main {
    public static void
    main(String args[])
    {
        Base b = new
    Derived();
        b.fun();
    }
}
```

innovate achieve lead

BITS Pilani

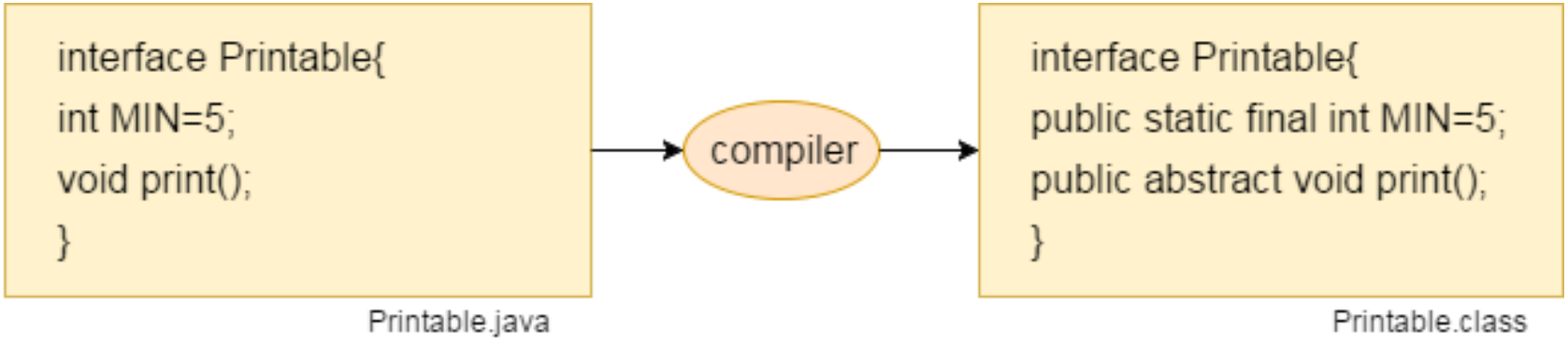
Pilani Campus

Interfaces

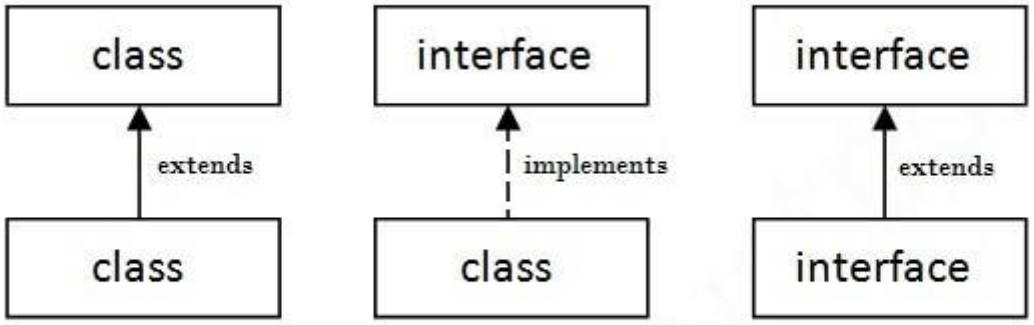
Interface

- Interface is a blueprint of a class containing static constants and abstract methods. It cannot have a method body.

- It is a mechanism to achieve abstraction.



Relationship between Classes and Interfaces



Interfaces - Example

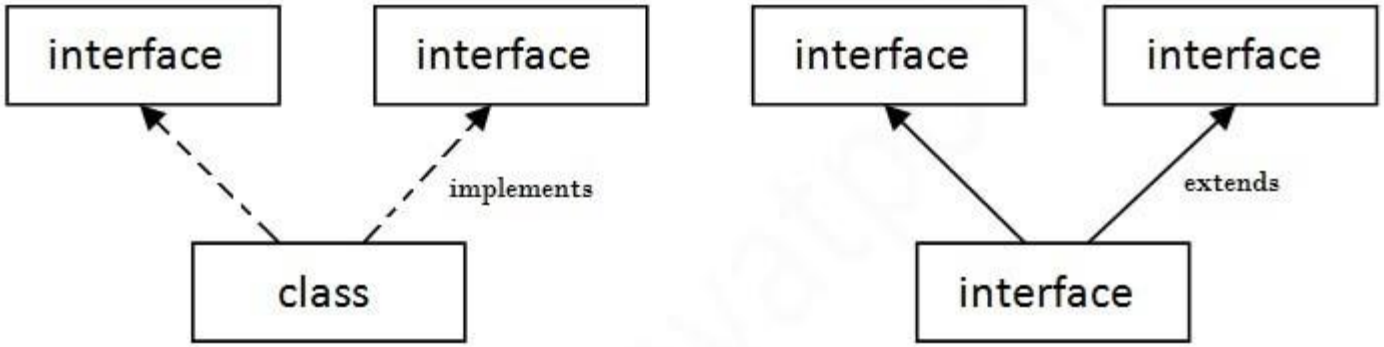
innovate achieve lead

```
Interface Bank {
 void deductFee();

class BankAccount implements Bank{
.
.
public void deductFee();{}
}

class CheckingAccount extends BankAccount implements Bank
{
.
.
.
}
```

Multiple Inheritance in Interface



Multiple Inheritance in Java

Why is Multiple Inheritance not a problem in Interface?


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

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

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();
}

}
```

Key points

- We can’t create instance(interface can’t be instantiated) of interface but we can make reference of it that refers to the Object of its implementing class.
- A class can implement more than one interface.
- An interface can extends another interface or interfaces (more than one interface) .
- A class that implements interface must implements all the methods in interface.
- All the methods are public and abstract. And all the fields are public, static, and final.
- It is used to achieve multiple inheritance.

Default Methods & Multiple Inheritance

innovate achieve lead

```
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();
}
}
```

Added features

New features added in interfaces in JDK 9

From Java 9 onwards, interfaces can contain following also

Static methods

Private methods

Private Static methods

Ques

- Why do we need interfaces when we already have abstract class?

abstract classes may contain non-final variables, whereas variables in interface are final, public and static.

- How can we use a static method from an interface?

- Why do we need private method in an Interface? How do we use it?
- Can we override the default method?

Default Methods & Multiple Inheritance

innovate achieve lead

```
public interface TempI {
  public abstract void mul(int a, int b);
  public default void add(int a, int b)
    {  sub(a, b);
 div(a, b);
 System.out.print(“Default method ");
System.out.println(a + b); }
  public static void mod(int a, int b)  {
        div(a, b);
System.out.print(" Static method”);
System.out.println(a % b); }
```

```
    private void sub(int a, int b)  {
      System.out.print(“Private method");
      System.out.println(a - b);  }
  private static void div(int a, int b){
      System.out.print(“Private static ");
      System.out.println(a / b);  } }
class Temp implements TempI {
   public void mul(int a, int b)  {
      System.out.print("Abstract method = ");
      System.out.println(a * b); }
   public static void main(String[] args) {
      TempI in = new Temp();
      in.mul(2, 3);
      in.add(6, 2);
      TempI.mod(5, 3);     } }
```