

FAST

National University of Computer and Emerging Sciences Peshawar

Lecture # 15

Software Construction and Development (Java Programming)

Instructor: Muhammad Abdullah Orakzai

DEPARTMENT OF COMPUTER SCIENCE



الذى علم بالقلم. علم الانسان ما لم يعلم.



Abstraction in Java

(Abstract classes and Abstract methods)

Contents

- 1) Abstraction in java
- 2) Abstract methods in java
- 3) Abstract classes in java

Abstraction in Java

- ❖ **Abstraction** is a process of hiding the implementation details and showing only functionality to the user.
- ❖ Hiding internal details and showing functionality.
- ❖ Another way, it shows only essential things to the user and hides the internal details.

Abstraction in Java...

- ❖ **for example**, sending SMS where you type the text and send the message. You don't know the internal processing about the message delivery.
- ❖ Abstraction lets you focus on what the object does instead of how it does it.

Ways to achieve Abstraction

There are two ways to achieve abstraction in java:

1. Abstract class (0 to 100%)
2. Interface (100%)

abstract keyword

- ❖ Used with methods called abstract method.
- ❖ Used with classes called abstract classes.

Abstract methods

- ❖ A method having no body is called abstract method.
- ❖ A method that is declared as abstract and does not have implementation is known abstract method.
- ❖ Keyword **abstract** is used to declare abstract methods.

Example of abstract methods

```
abstract void printStatus();    //no method body
```

Abstract class in Java

- ❖ A class having at least one abstract method is called abstract class.
- ❖ It also contains concrete methods.
- ❖ A class which is declared with the abstract keyword is known as an abstract class in Java.

Abstract class in Java...

- ❖ It can have abstract and non-abstract methods (method with the body).
- ❖ It needs to be extended and its method to be implemented.
- ❖ It cannot be instantiated.
- ❖ Usually used as a super class, so that other classes extend it.

Abstract class in Java...

Note:

The class which extend or inherits the abstract class will must implement the abstract method of abstract class. And if that class does not implement its method then it is also written as abstract class.

Abstract class in Java...

Points to Remember

1. An abstract class must be declared with an abstract keyword.
2. It can have abstract and non-abstract methods.
3. It cannot be instantiated.
4. It can have constructors and static methods also.
5. It can have final methods which will force the subclass not to change the body of the method.

Example of abstract class

abstract class A

{

Code to be written

}

Example of Abstract class that has an abstract method

In this example, Bike is an abstract class that contains only one abstract method run. Its implementation is provided by the Honda class.

```
abstract class Bike{    // abstract class

    abstract void run();  // abstract method

}
```

Example of Abstract class that has an abstract method...

```
class Honda4 extends Bike{  
    void run(){System.out.println("running safely");}  
    public static void main(String args[]){  
        Bike obj = new Honda4();  
        obj.run();  
    }  
}
```

Output: running safely

Abstract class with concrete method

```
abstract class Abs{  
    public abstract void met();  
    public void met2()  
    {  
        System.out.println("met2 called");  
    }  
    class Test extends Abs{  
        -----  
    }
```

Note

Abs obj; // obj is reference variable

Test t = new Test();

obj = t;

OR

Abs obj = new Test();

Abstract methods and abstract class example

```
abstract class Animal {    // abstract class
    public void eat() {
        System.out.println("Eating.....");
    }
    public void sleep() {
        System.out.println("Sleeping.....");
    }
    public abstract void sound();    // abstract method
}
```

Abstract methods and abstract class example...

```
class Cat extends Animal {  
    public void sound() {  
        System.out.println("Mew Mew.....");  
    }  
}  
  
class Horse extends Animal  
{  
    public void sound() {  
        System.out.println("Hehehehe.....");  
    } }  
}
```

Abstract methods and abstract class example...

```
class Cow extends Animal
{
    public void sound()
    {
        System.out.println("Baw Baw.....");
    }
}
```

Abstract methods and abstract class example...

```
public class AbstractionDemo {           // main class  
    public static void main(String[] args) {
```

```
        Animal a;    // animal class reference variable
```

```
        Cat mano = new Cat();    // Cat class object
```

```
        Horse tony = new Horse(); // Horse class object
```

```
        Cow bano = new Cow();    // Cow class object
```

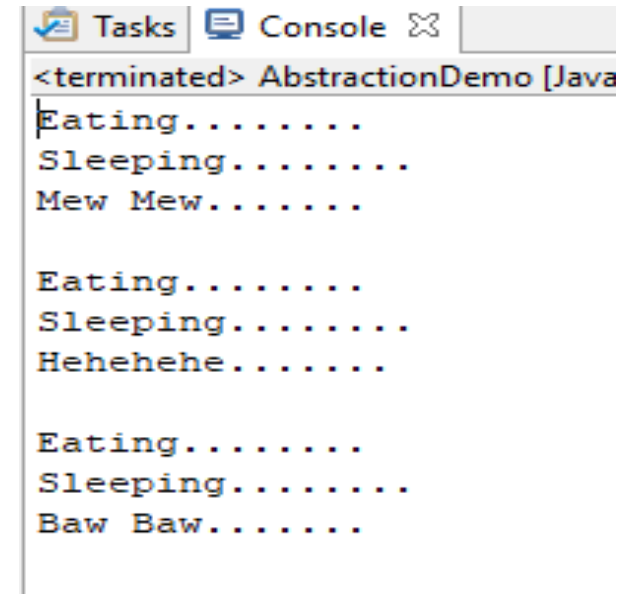
Abstract methods and abstract class example...

```
a = mano;
```

```
a.eat();
```

```
a.sleep();
```

```
a.sound();
```



The screenshot shows a Java IDE console window with the title "<terminated> AbstractionDemo [Java]". The console output displays three sets of actions performed by an object, each set separated by a blank line. The first set shows "Eating.....", "Sleeping.....", and "Mew Mew.....". The second set shows "Eating.....", "Sleeping.....", and "Hehehehe.....". The third set shows "Eating.....", "Sleeping.....", and "Baw Baw.....".

```
<terminated> AbstractionDemo [Java]
Eating.....
Sleeping.....
Mew Mew.....

Eating.....
Sleeping.....
Hehehehe.....

Eating.....
Sleeping.....
Baw Baw.....
```

Abstract methods and abstract class example...

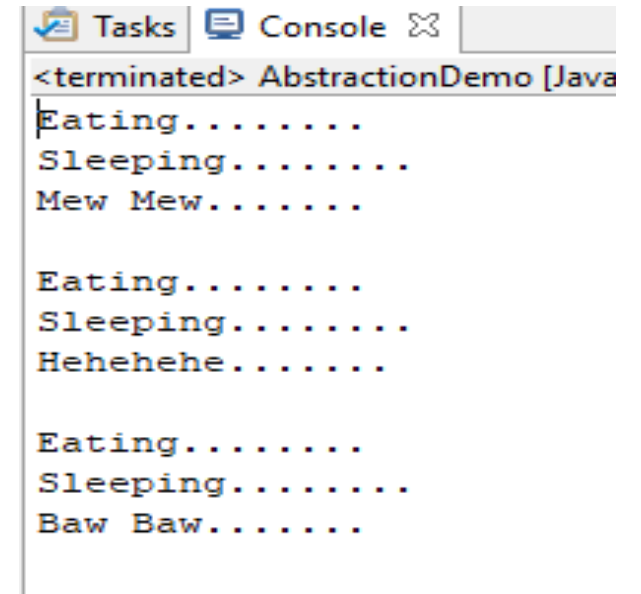
```
System.out.println("");    // for new line
```

```
a = tony;
```

```
a.eat();
```

```
a.sleep();
```

```
a.sound();
```

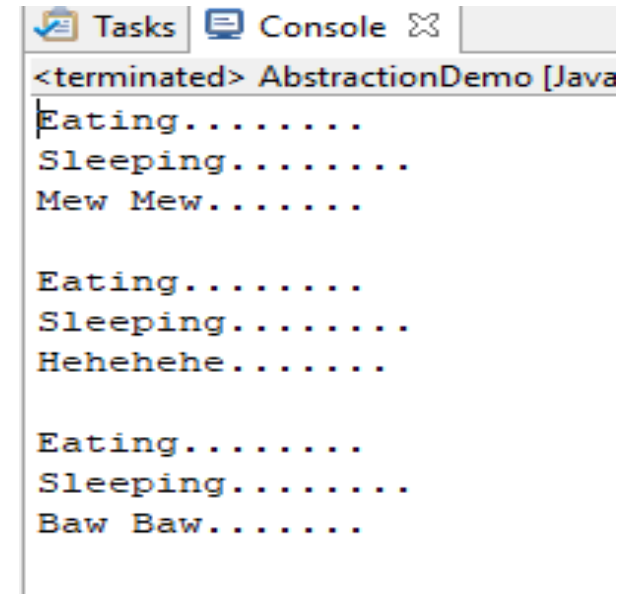


The screenshot shows a Java IDE console window with the title "<terminated> AbstractionDemo [Java]". The console output displays three sets of actions performed by an object named 'a':

```
Eating.....  
Sleeping.....  
Mew Mew.....  
  
Eating.....  
Sleeping.....  
Hehehehe.....  
  
Eating.....  
Sleeping.....  
Baw Baw.....
```

Abstract methods and abstract class example...

```
System.out.println("");  
a = bano;  
a.eat();  
a.sleep();  
a.sound();  
}  
} // main class body end
```



The screenshot shows a Java IDE window titled "Tasks Console" with a sub-window titled "<terminated> AbstractionDemo [Java]". The console output displays three sets of actions: "Eating.....", "Sleeping.....", and "Mew Mew....." repeated three times, followed by "Eating.....", "Sleeping.....", and "Hehehehe....." repeated three times, and finally "Eating.....", "Sleeping.....", and "Baw Baw....." repeated three times.

```
<terminated> AbstractionDemo [Java]  
Eating.....  
Sleeping.....  
Mew Mew.....  
  
Eating.....  
Sleeping.....  
Hehehehe.....  
  
Eating.....  
Sleeping.....  
Baw Baw.....
```


Abstract methods and abstract class

- ❖ Same method call producing different results (polymorphism)
- ❖ An object having different shapes (polymorphism)

Note:

- ❖ If there is any abstract method in a class that class must be abstract.
- ❖ If you extending any abstract class that have abstract method you must either provide implementation of the method or make this class abstract.

THANK YOU

