# Object Oriented Programming using Java

Prepared By:
Suyel, PhD
Assistant Professor
Dept. of CSE, NIT Patna

## Outline

1. final Keywords
2. Uses of final Keyword
3. abstract Keyword
4. Rules of abstract Keyword
5. Uses of abstract Keyword

## final Keyword

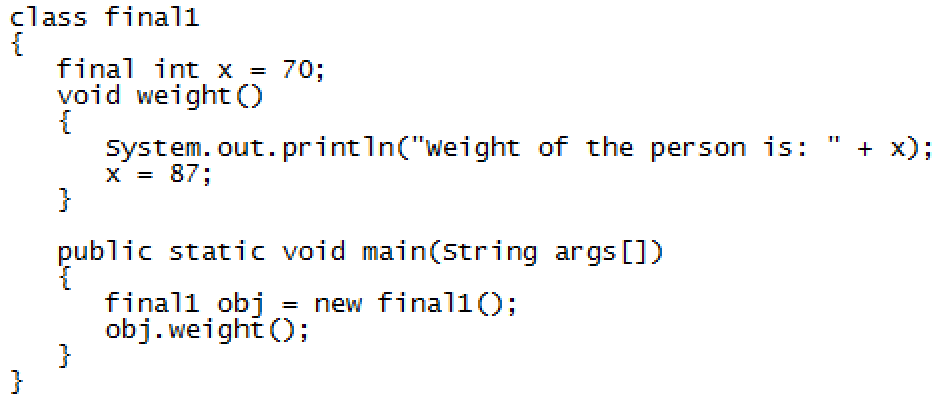
- The **final** keyword in Java is used to restrict the user.
- Here, the final keyword is used to denote constants.
- The final is a non-access modifier.
- In Java, the final keyword can be used while declaring an entity.
- The final keyword can be used along with variables, methods and classes.
- Using the final keyword means that the value cannot be modified in the future.

## Uses of final Keyword

- **final variable**
  - Final variables are nothing but constants.
  - We cannot change the value of a final variable once it is initialized.
  - A final variable that is not initialized at the time of declaration is known as **blank final variable**. We must initialize the blank final variable in constructor of the class.
  - A static final variable that is not initialized during declaration is called as **static blank final variable**. It is initialized only in static block.
  - The final keyword with a parameter variable means that the value of this variable cannot be changed in the function.
  - If we have a final reference variable, we cannot reassign it. But, this doesn’t mean that the object it refers to is immutable. We can change the properties of this object freely.

## Uses of final Keyword (Cont…)

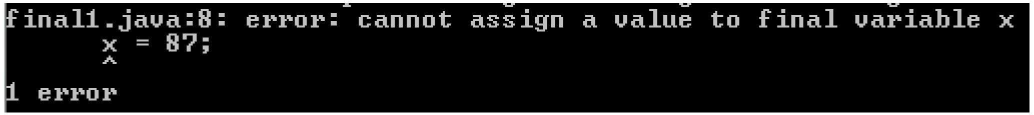
- **final variable (Cont…)**



```
class final1
{
    final int x = 70;
    void weight()
    {
        System.out.println("Weight of the person is: " + x);
        x = 87;
    }

    public static void main(String args[])
    {
        final1 obj = new final1();
        obj.weight();
    }
}
```

## Uses of final Keyword (Cont…)

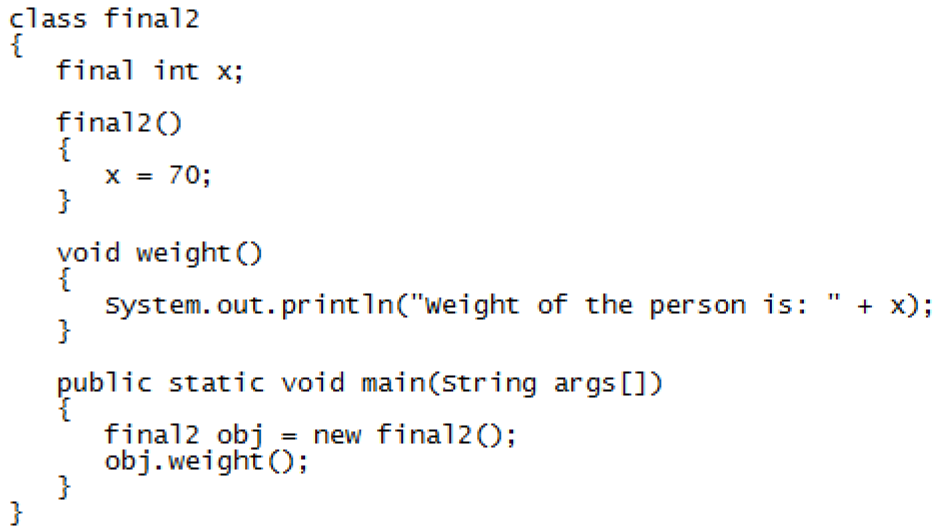
- **final variable (Cont…)**
  - **Output**



```
final1.java:8: error: cannot assign a value to final variable x
        x = 87;
        ^
1 error
```

## Uses of final Keyword (Cont…)

- **final variable (Cont…)**



```
class final2
{
    final int x;

    final2()
    {
        x = 70;
    }

    void weight()
    {
        System.out.println("Weight of the person is: " + x);
    }

    public static void main(String args[])
    {
        final2 obj = new final2();
        obj.weight();
    }
}
```

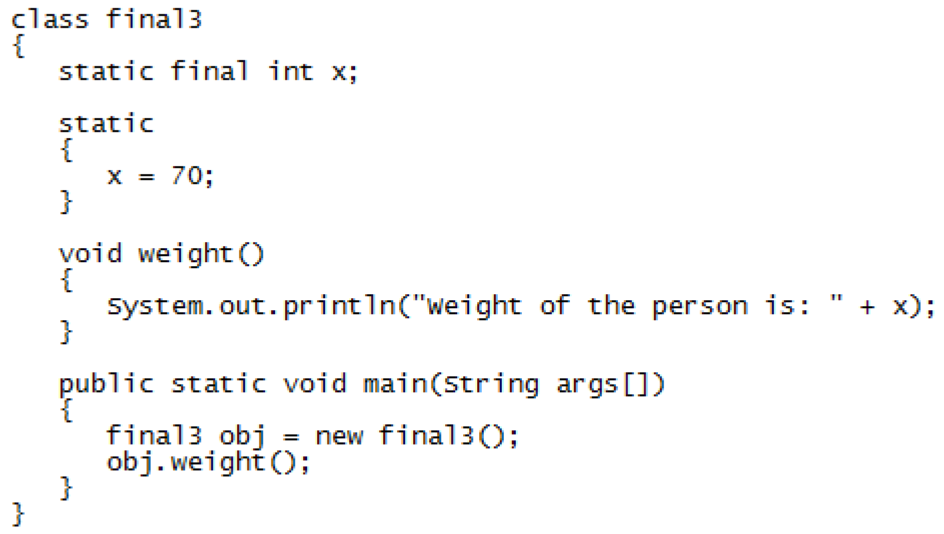
## Uses of final Keyword (Cont…)

- **final variable (Cont…)**
  - **Output**

```
Weight of the person is: 70
```

## Uses of final Keyword (Cont…)

- **final variable (Cont…)**



```
class final3
{
    static final int x;

    static
    {
        x = 70;
    }

    void weight()
    {
        System.out.println("Weight of the person is: " + x);
    }

    public static void main(String args[])
    {
        final3 obj = new final3();
        obj.weight();
    }
}
```

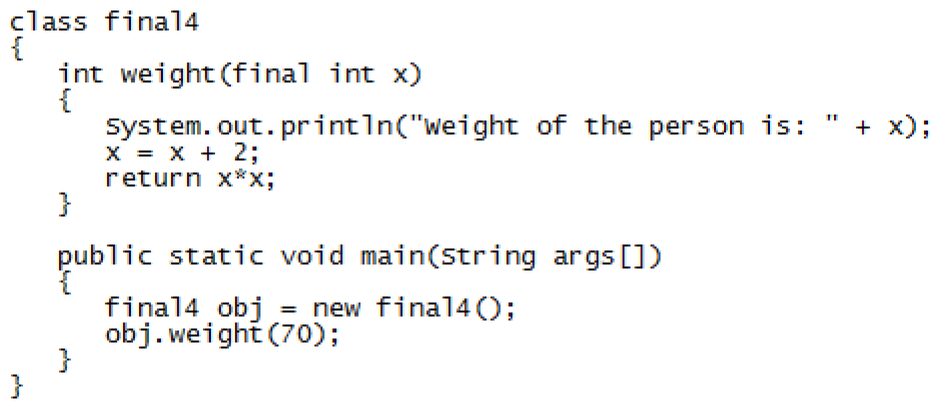
## Uses of final Keyword (Cont…)

- **final variable (Cont…)**
  - **Output**

```
Weight of the person is: 70
```

## Uses of final Keyword (Cont…)

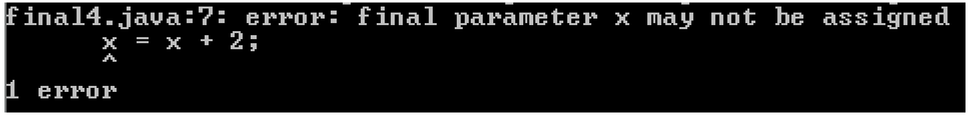
- **final variable (Cont…)**



```
class final4
{
    int weight(final int x)
    {
        System.out.println("Weight of the person is: " + x);
        x = x + 2;
        return x*x;
    }

    public static void main(String args[])
    {
        final4 obj = new final4();
        obj.weight(70);
    }
}
```

## Uses of final Keyword (Cont…)

- **final variable (Cont…)**
  - **Output**



```
final4.java:7: error: final parameter x may not be assigned
      x = x + 2;
      ^
1 error
```

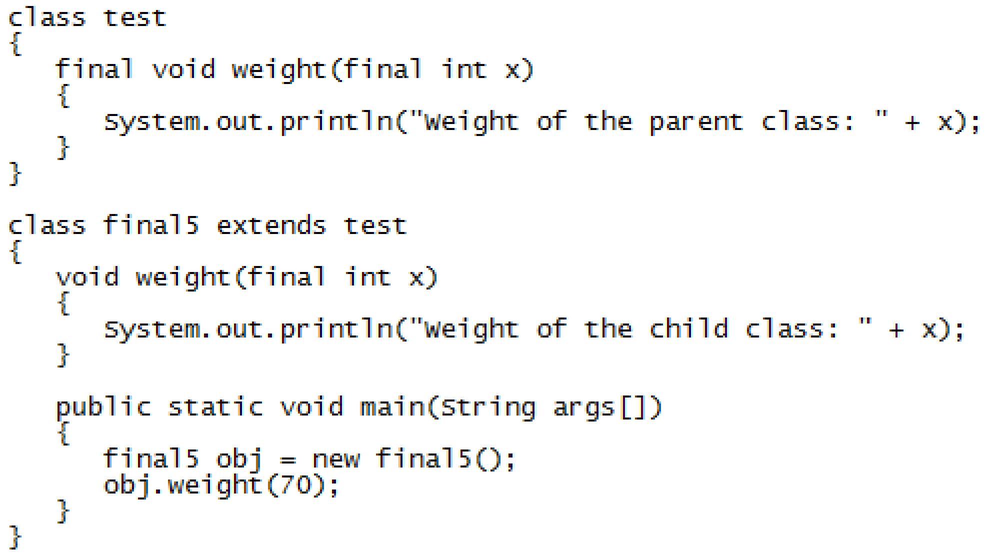
## Uses of final Keyword (Cont…)

- **final Method**

  - In Java, the final method cannot be overridden by the child class.
  - A child class can call the final method of parent class without any issues, but it cannot override it.

# Uses of final Keyword (Cont…)

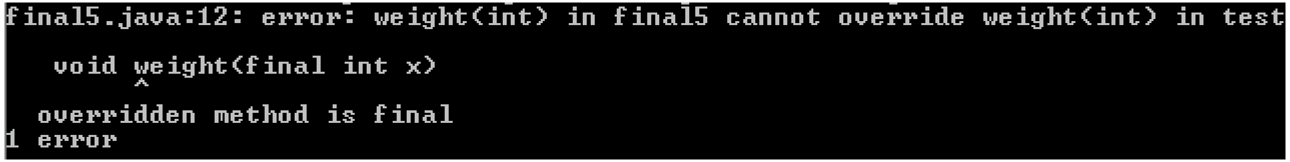
## ❑ final Method (Cont…)



```
class test
{
    final void weight(final int x)
    {
        System.out.println("Weight of the parent class: " + x);
    }
}

class final5 extends test
{
    void weight(final int x)
    {
        System.out.println("Weight of the child class: " + x);
    }

    public static void main(String args[])
    {
        final5 obj = new final5();
        obj.weight(70);
    }
}
```

# Uses of final Keyword (Cont…)

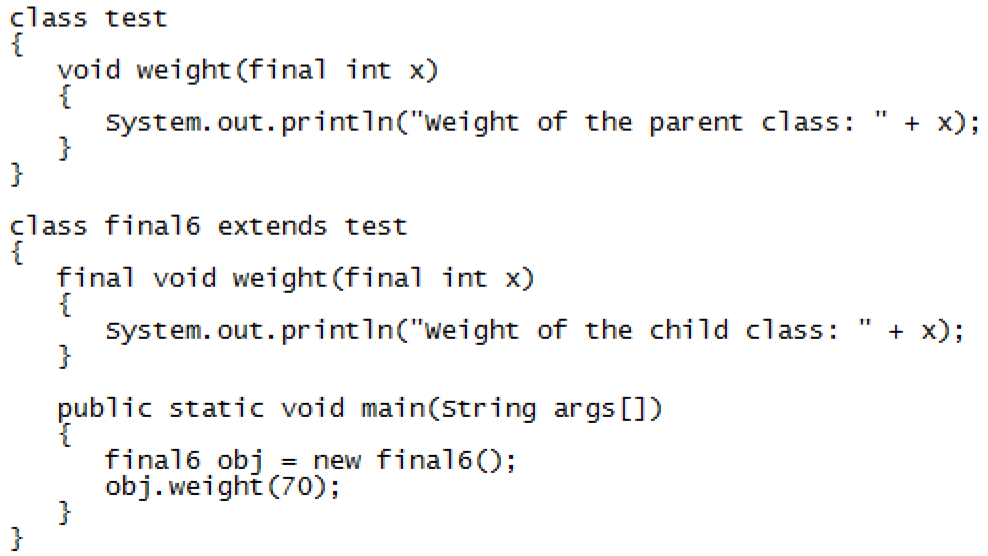
- final Method (Cont…)
  - Output



```
final5.java:12: error: weight(int) in final5 cannot override weight(int) in test

   void weight(final int x)
        ^
  overridden method is final
1 error
```

## Uses of final Keyword (Cont…)

- final Method (Cont…)



```
class test
{
   void weight(final int x)
   {
      System.out.println("Weight of the parent class: " + x);
   }
}

class final6 extends test
{
   final void weight(final int x)
   {
      System.out.println("Weight of the child class: " + x);
   }

   public static void main(String args[])
   {
      final6 obj = new final6();
      obj.weight(70);
   }
}
```

## Uses of final Keyword (Cont…)

- **final Method (Cont…)**
  - **Output**

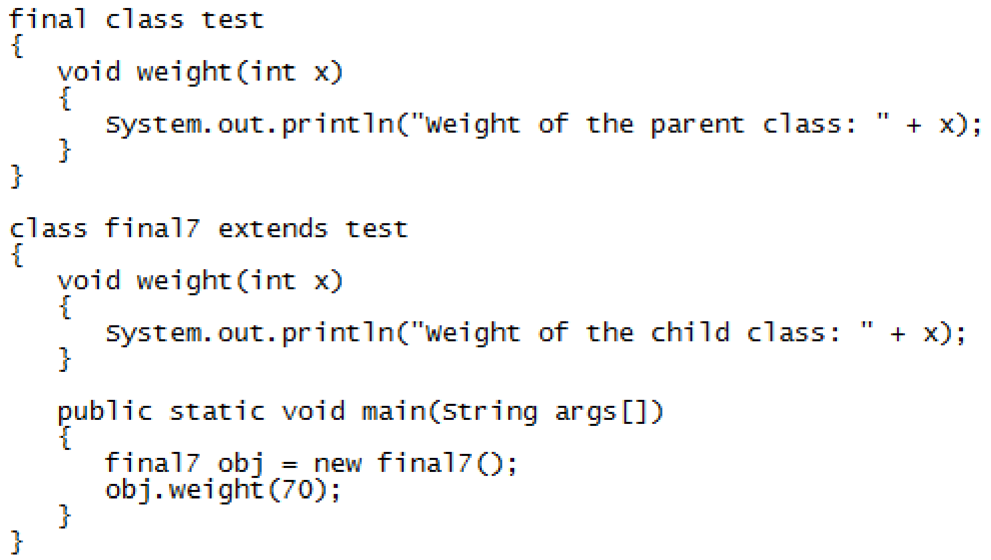
```
Weight of the child class: 70
```

## Uses of final Keyword (Cont…)

- **final Class**
  - When a class is declared with the final keyword, it is called a final class.
  - A final class cannot be extended (inherited).
  - Any attempt to inherit from a final class causes a compiler error.
  - There are two uses of a final class:
    1. To prevent inheritance as final classes cannot be extended. For example, all Wrapper Classes like Integer, Float, etc. are final classes. We cannot extend them.
    2. To create an immutable class like the predefined String class. We cannot make a class immutable without making it final.

## Uses of final Keyword (Cont…)

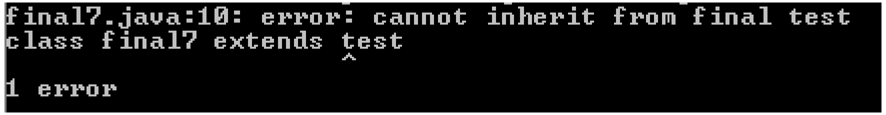
❑ **final Class (Cont…)**



```
final class test
{
   void weight(int x)
   {
      System.out.println("Weight of the parent class: " + x);
   }
}

class final7 extends test
{
   void weight(int x)
   {
      System.out.println("Weight of the child class: " + x);
   }

   public static void main(String args[])
   {
      final7 obj = new final7();
      obj.weight(70);
   }
}
```

## Uses of final Keyword (Cont…)

- **final Class (Cont…)**
  - **Output**



```
final7.java:10: error: cannot inherit from final test
class final7 extends test
                     ^
1 error
```

## abstract Keyword

- The **abstract** keyword is used to achieve abstraction in Java.
- It is a non-access modifier.
- It is used to create **abstract class** and **abstract method**.
- **Abstract method**
  - A method that is declared with abstract keyword and doesn't have any implementation is known as an **abstract method**.
  - These methods are sometimes referred to as **subclasser responsibility** because they have no implementation specified in the superclass.
  - Thus, a subclass must override them to provide method body.
  - Any class that contains one or more abstract methods must also be declared with abstract.

## abstract Keyword (Cont…)

- **Abstract class**
  - A class which contains the abstract keyword in its declaration is known as an **abstract class**.
  - Abstract classes may or may not contain abstract methods.
  - If a class has at least one abstract method, then the class must be declared abstract.
  - Due to their partial implementation, we cannot instantiate abstract classes.
  - To use an abstract class, we have to inherit it from another class, provide implementations for the abstract methods in it.
  - If we inherit an abstract class, we have to provide implementations to all the abstract methods in it.

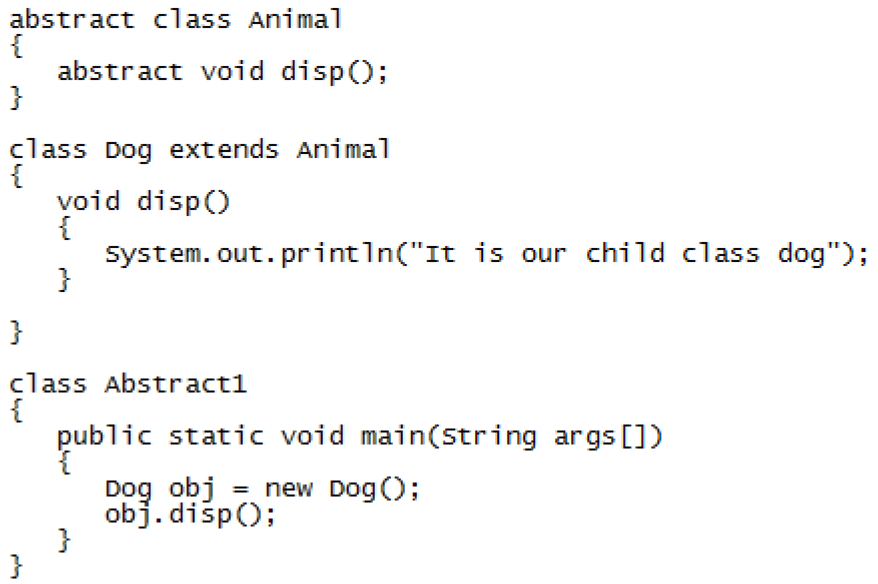
## Rules of abstract Keyword

- **Allowed**
  - The abstract keyword can only be used with class and method.
  - An abstract class can contain constructors and static methods.
  - If a class extends the abstract class, it must also implement at least one of the abstract method.
  - An abstract class can contain the main method and the final method.
  - An abstract class can contain overloaded abstract methods.
  - We can declare the local inner class as abstract.
  - We can declare the abstract method with a throw clause.

## Rules of abstract Keyword (Cont…)

- **Not Allowed**
  - The abstract keyword cannot be used with variables and constructors.
  - If a class is abstract, it cannot be instantiated.
  - If a method is abstract, it doesn’t contain the body.
  - We cannot use the abstract keyword with the final.
  - We cannot declare abstract methods as private.
  - We cannot declare abstract methods as static.
  - An abstract method cannot be synchronized.

# Uses of abstract Keyword



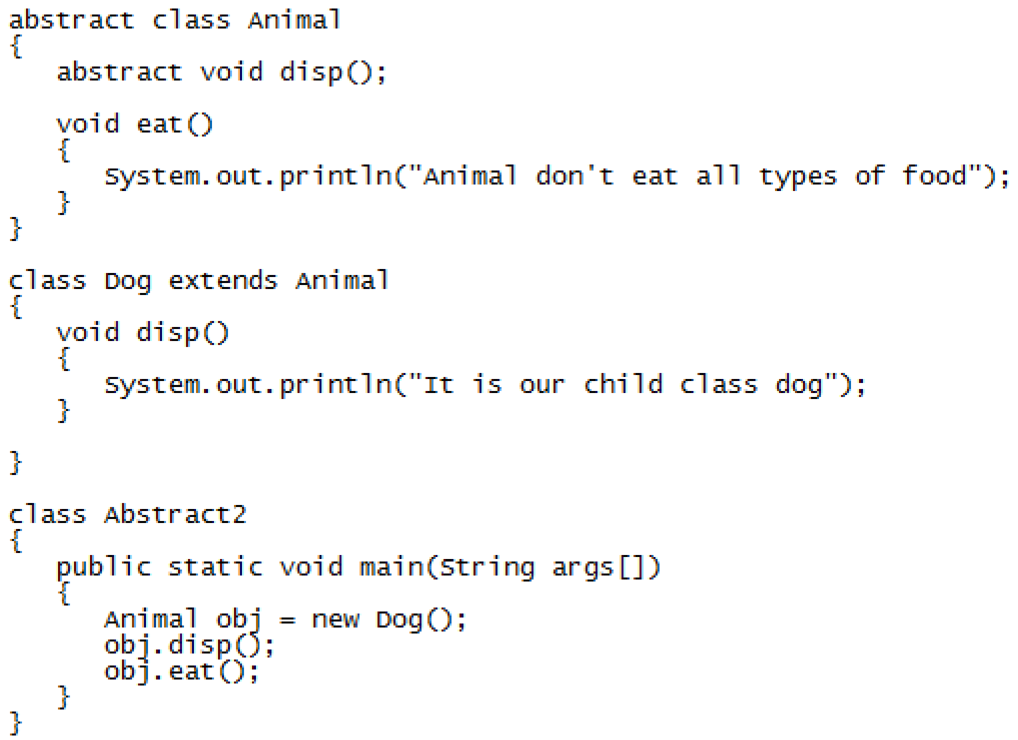
```
abstract class Animal
{
   abstract void disp();
}

class Dog extends Animal
{
   void disp()
   {
      System.out.println("It is our child class dog");
   }

}

class Abstract1
{
   public static void main(String args[])
   {
      Dog obj = new Dog();
      obj.disp();
   }
}
```

## Uses of abstract Keyword (Cont…)

- **Output**

```
It is our child class dog
```

# Uses of abstract Keyword (Cont…)



```
abstract class Animal
{
   abstract void disp();

   void eat()
   {
      System.out.println("Animal don't eat all types of food");
   }
}

class Dog extends Animal
{
   void disp()
   {
      System.out.println("It is our child class dog");
   }

}

class Abstract2
{
   public static void main(String args[])
   {
      Animal obj = new Dog();
      obj.disp();
      obj.eat();
   }
}
```

## Uses of abstract Keyword (Cont…)

- **Output**

```
It is our child class dog
Animal don't eat all types of food
```

# Uses of abstract Keyword (Cont…)

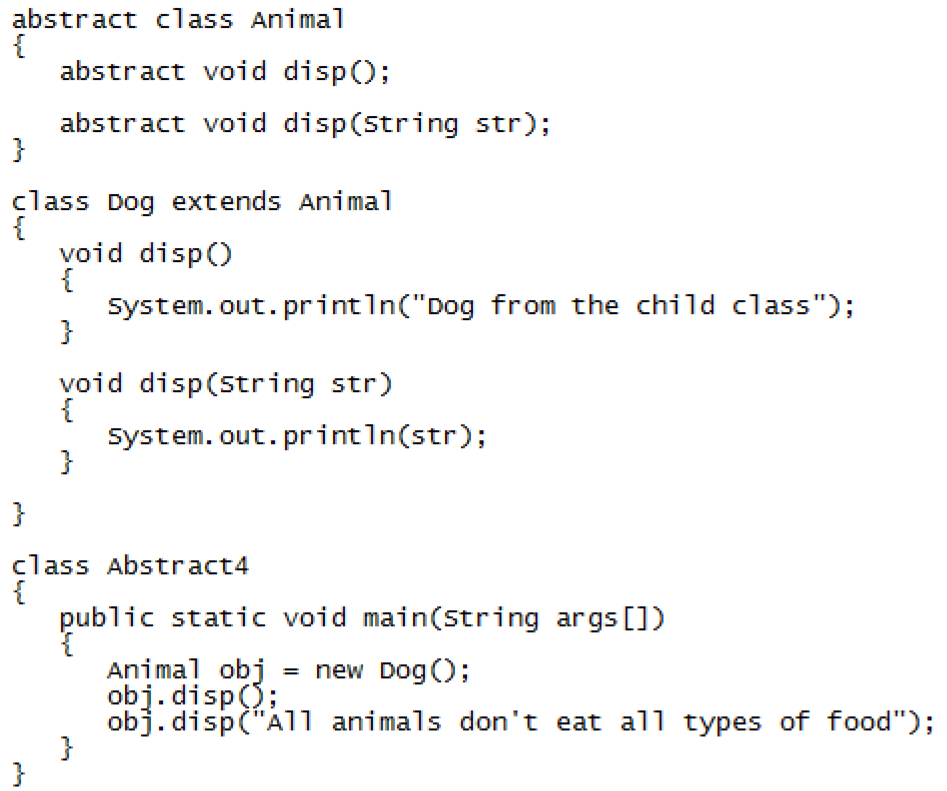
```
abstract class Animal
{
   String str;

   Animal(String str)
   {
      this.str = str;
   }

   abstract void disp();

   void eat()
   {
      System.out.println("Animal don't eat all types of food");
   }
}

class Dog extends Animal
{
   Dog(String str)
   {
      super(str);
   }

   void disp()
   {
      System.out.println("Name of the Dog is: " + str);
   }

}

class Abstract3
{
   public static void main(String args[])
   {
      Dog obj = new Dog("Max");
      obj.disp();
      obj.eat();
   }
}
```

## Uses of abstract Keyword (Cont…)

- **Output**

```
Name of the Dog is: Max
Animal don’t eat all types of food
```

## Uses of abstract Keyword (Cont…)



```
abstract class Animal
{
    abstract void disp();

    abstract void disp(String str);
}

class Dog extends Animal
{
    void disp()
    {
        System.out.println("Dog from the child class");
    }

    void disp(String str)
    {
        System.out.println(str);
    }

}

class Abstract4
{
    public static void main(String args[])
    {
        Animal obj = new Dog();
        obj.disp();
        obj.disp("All animals don't eat all types of food");
    }
}
```

## Uses of abstract Keyword (Cont…)

- **Output**

```
Dog from the child class
All animals don’t eat all types of food
```

thank
you . . .

**Slides are prepared from various sources, such as Book, Internet Links and many more.**