BITS Pilani

Object-Oriented Programming (CS F213)

Module I: Object-Oriented and Java Basics

CS F213 RL 6.4: Object class in Java

Dr. Pankaj Vyas
Department of Computer Science, BITS-Pilani, Pilani Campus

CS F213 RL 6.4 : Topics

- Object class in Java

Object class in Java

- Supermost class in Java. [java.lang.Object]
- If a class does not extend any super class then that class is a direct sub class of Object class.

```
class X                 class Y                 class Z
{                       {                       {
}// End of class X      }// End of class Y      }// End of class Z
```

- Classes X, Y and Z as shown above are the direct sub-classes of Object class.
- Every class in Java, directly or in-directly is a sub-class of Object.

Important Methods of Object class

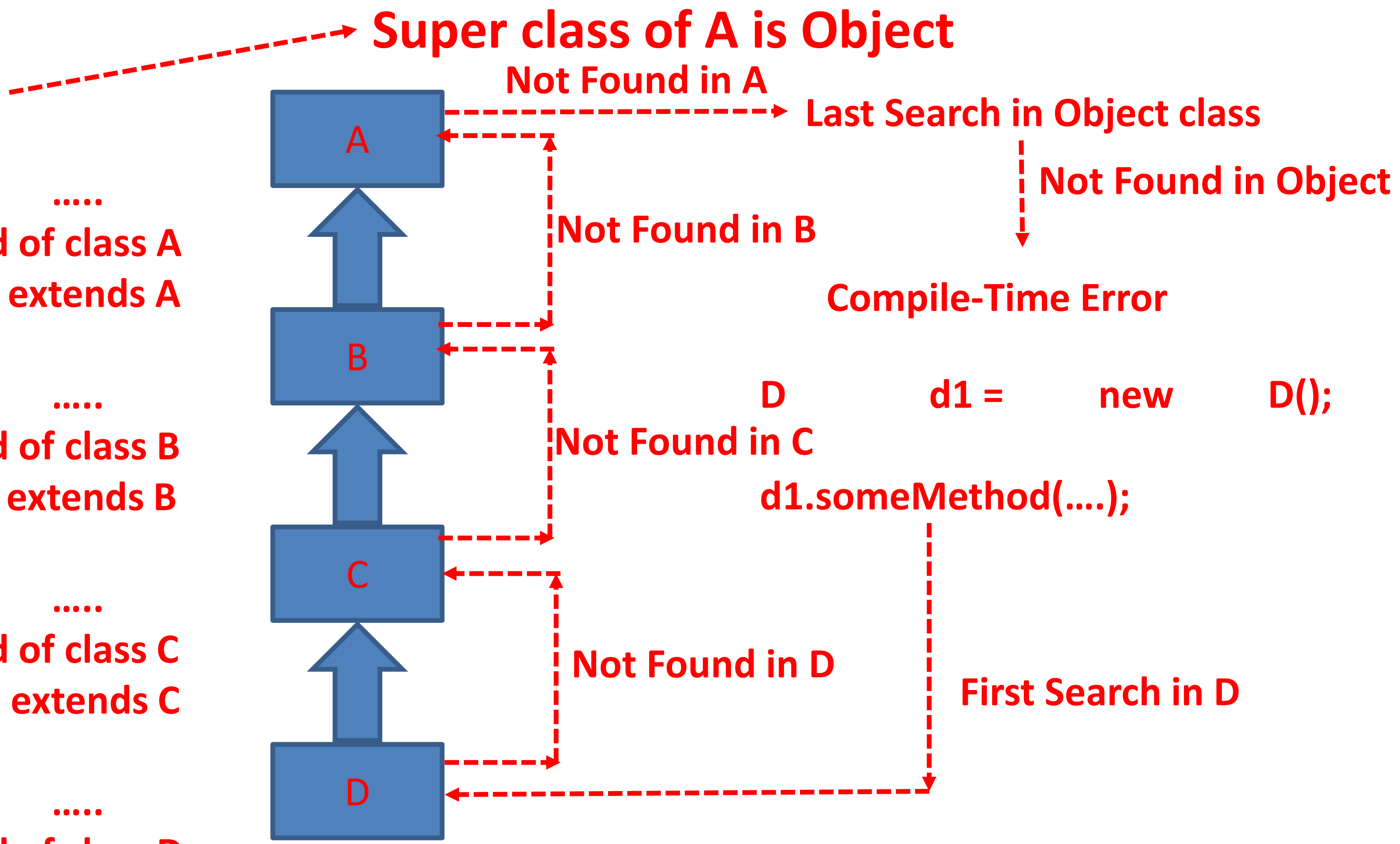
- public int hashCode()
- public boolean equals(Object obj)
- public String toString()

Discussed in this Lecture

- protected void finalize() throws Throwable

How Java Searches For Called Methods

```
class A
{
        .....
}// End of class A
class B extends A
{
        .....
}// End of class B
class C extends B
{
        .....
}// End of class C
class D extends C
{
        .....
}// End of class D
```



```
D d1 = new D();
d1.someMethod(….);
```

public int hashCode()

- Returns the hash-code of the Object. Hashcode → an Integer Representation of an Object.
- hashCode() Method of Object class considers the memory address of the object as the hash-code
- So, the method returns the memory address of the object in hexadecimal form as the hash-code value of that object

public int hashCode() : Example

```
// File Name : Test.java
class A     {           }           // End of class A
class Test
{
          public static void main(String args[])
          {
                    A         a1        =         new       A();
                    A         a2        =         new       A();
                    A         a3        =         a1;

                    System.out.println("Hash Code a1 :" + a1.hashCode());
                    System.out.println("Hash Code a2 :" + a2.hashCode());
                    System.out.println("Hash Code a3 :" + a3.hashCode());

          }// End of Method
}// End of class Test
```

hashCode() Method Invoked from Object class

<<OUTPUT>>

```
F:\>java Test
Hash Code a1 :1284693
Hash Code a2 :31168322
Hash Code a3 :1284693
```

public booelan equals()

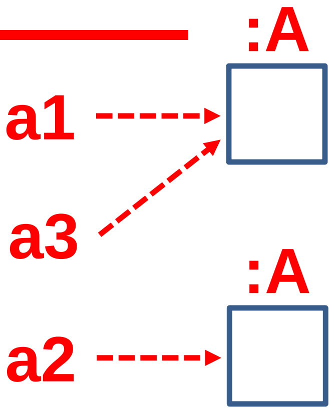
- Compares this object-reference and ‘obj’ for equality and returns true if equal otherwise false
- The equals() method of Object class tests whether the hash-codes of the object are equal or not.
- Hash-codes (as per the implementation hashCode() Method in Object class) of two objects are equal if and only if they have same memory address

equals() Method : Example

```
// File Name : Test.java
class A { } // End of class A
class Test
{
    public static void main(String args[])
    {
        A       a1      =       new     A();
        A       a2      =       new     A();
        A       a3      =       a1;


        System.out.println(a1.equals(a2));
        System.out.println(a1.equals(a3));

    }// End of Method
}// End of Test class
```



<<OUTPUT>>

```
F:\>java Test
false
true
```

equals() Method is called from Object class

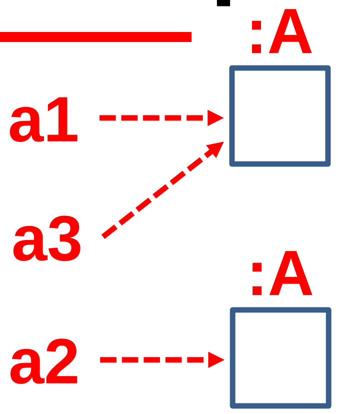
== (Equality Operator) for Object-Reference Equality : Example

```
// File Name : Test.java
class A { } // End of class A
class Test
{
    public static void main(String args[])
    {
        A       a1      =       new     A();
        A       a2      =       new     A();
        A       a3      =       a1;

        if(a1 == a2)
                System.out.println("Hello");
        else
                System.out.println("Hi");

        if(a1 == a3)
                System.out.println("Thanks");
        else
                System.out.println("Welcome");

    }// End of Method
}// End of Test class
```



<<OUTPUT>>

```
F:\>java Test
Hi
Thanks
```

'==' equality operator when used for object-references returns true only if object-references points to the same object

Supplying equals() Method in class

- You can overload the equals() Method as follows

```
public          boolean          equals(<T> obj)
{
                .....
} // End of Method
```

<T> is class type in which the method is supplied

- You can override the equals() Method as follows

```
public          boolean          equals(Object obj)
{
                T var   = (T) obj;
                .....

} // End of Method
```

Parameter is Object Type

Type cast the parameter first to local class and then code equals Method

Supplying equals() Method in a class: Example

innovate achieve lead

```
// File Name : Test.java
class           Circle
{
      private  double        radius;        // Instance Field : radius
      // Constructor Method
      Circle(double radius)                 {  this.radius = radius;   }
      // Accessor Method
      public   double        getRadius()  {  return this.radius;  }
      // Method to compute area
      public   double        area()         { return 3.1456 * radius * radius;  }
      // Method to compute perimeter
      public   double        perimeter()  { return 2* 3.1456 * radius;   }

      // equals method → Method Overriding
      public boolean      equals(Object   o)
      {
            Circle  c =  (Circle) o;                // First Type Cast
            return this.area() == c.area();
      } // End of method
} // End of class Circle
```

```
Circle c1 = new Circle(10.5);

Circle c2 = new Circle(6);

Circle c3 = new Circle(10.5);

System.out.println(c1.equals(c2));

System.out.println(c1.equals(c3));
```

equals() Method in this case will be invoked from Circle class

<<OUTPUT>>

```
F:\>java Test
false
true
```

Supplying equals() Method in a class: Example

```
// File Name : Test.java
class           Circle
{
      private  double        radius;       // Instance Field : radius
      // Constructor Method
      Circle(double radius)                { this.radius = radius;  }
      // Accessor Method
      public   double        getRadius()   {  return this.radius;  }
      // Method to compute area
      public   double        area()        { return 3.1456 * radius * radius;  }
      // Method to compute perimeter
      public   double        perimeter()   { return 2* 3.1456 * radius;   }

      // equals method → Method Overloading
      public boolean      equals(Circle   o)
      {
                 return this.area() == c.area();
      } // End of Method
} // End of class Circle
```

```
Circle c1 = new Circle(10.5);

Circle c2 = new Circle(6);

Circle c3 = new Circle(10.5);

System.out.println(c1.equals(c2));

System.out.println(c1.equals(c3));
```

equals() Method in this case will be invoked from Circle class

public String toString() Method

- public String toString()

- ❑ Returns string form of Object
- ❑ System.out.println() → Always displays in String form
- ❑ The default toString() method in Object class displays the output in following form

  <<class-name-of-Object> @ <<hash-code-of-object>>

- ❑ System.out.println() → calls toString() upon the parameters that belongs to class type. For Example

System.out.println(x); → System.out.println(x.toString());

public String toString() Method : Example

```
// File Name : Test.java
class           Circle
{
        private  double        radius;        // Instance Field : radius
        // Constructor Method
        Circle(double radius)                 {  this.radius = radius;   }
        // Accessor Method
        public   double        getRadius() {  return this.radius;  }
        // Method to compute area
        public   double        area()         { return 3.1456 * radius * radius;  }
        // Method to compute perimeter
        public   double        perimeter()  { return 2* 3.1456 * radius;   }

} // End of class Circle
```

- Circle class has not supplied any toString() Method
- In this Example toString() will be called from Object class

```
Circle c1 = new Circle(10.5);

Circle c2 = new Circle(6);

Circle c3 = new Circle(10.5);

System.out.println(c1);
System.out.println(c2);
System.out.println(c3);

System.out.println(c1.toString());
System.out.println(c2.toString());
System.out.println(c3.toString());
```

<<OUTPUT>>

```
Circle@139a55
Circle@1db9742
Circle@106d69c
```

Supplying a toString() Method in a class : Example 1

```
// File Name : Test.java
class           Circle
{
      private  double        radius;        // Instance Field : radius
      // Constructor Method
      Circle(double radius)                 {  this.radius = radius;   }
      // Accessor Method
      public   double        getRadius()  {  return this.radius;  }
      // Method to compute area
      public   double        area()         { return 3.1456 * radius * radius;  }
      // Method to compute perimeter
      public   double        perimeter()  { return 2* 3.1456 * radius;   }

      // Supplying toString() Method
      public String           toString()
      {
                 return “Welcome to Object World”);
      }// End of Method

} // End of class Circle
```

```
Circle c1 = new Circle(10.5);

Circle c2 = new Circle(6);

Circle c3 = new Circle(10.5);

System.out.println(c1);
System.out.println(c2);
System.out.println(c3);
```

- Circle class has supplied its own toString() Method
- In this Example toString() will be called from class

<<OUTPUT>>

```
Welcome to Object World
Welcome to Object World
Welcome to Object World
```

Supplying a toString() Method in a class : Example 2

```
// File Name : Test.java
class           Circle
{
      private  double       radius;        // Instance Field : radius
      // Constructor Method
      Circle(double radius)                {  this.radius = radius;   }
      // Accessor Method
      public   double       getRadius()  {  return this.radius;  }
      // Method to compute area
      public   double       area()         { return 3.1456 * radius * radius;  }
      // Method to compute perimeter
      public   double       perimeter()  { return 2* 3.1456 * radius;   }

      // Supplying toString() Method
      public String             toString()
      {
                return "Radius: " + this.radius + " Area=" + this.area() ;
      }// End of Method

} // End of class Circle
```

```
Circle c1 = new Circle(10.5);

Circle c2 = new Circle(6);

Circle c3 = new Circle(10.5);

System.out.println(c1);
System.out.println(c2);
System.out.println(c3);
```

- Circle class has supplied its own toString() Method
- In this Example toString() will be called from class

<<OUTPUT>>

```
Radius: 10.5 Area=346.8024
Radius: 6.0 Area=113.2416
Radius: 10.5 Area=346.8024
```

Thank You