Nesne Yönelimli Programlama

Object class

Hüseyin Ahmetoğlu

Object class

- Java.lang paketinde Object sınıfı mevcuttur.
- Java'daki her sınıf, doğrudan veya dolaylı olarak Object sınıfından türetilir.
- Bir Sınıf başka bir sınıfı genişletmiyorsa, o zaman Object'in doğrudan alt sınıfıdır ve diğer sınıfı genişletiyorsa, dolaylı olarak türetilmiş bir sınıftır.
- Object sınıfı yöntemleri tüm Java sınıfları tarafından kullanılabilir.
- Object sınıfı, herhangi bir Java Programında kalıtım hiyerarşisinin bir kökü olarak hareket eder.

toString()

- Bir nesnenin sahip olduğu özellikleri yazdırmak için kullanılır.

```
class Complex {
    private double re, im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }
}

// Driver class to test the Complex class
public class Main {
    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        System.out.println(c1);
    }
}
```

Output:

```
Complex@19821f
```

toString()

```
class Complex {
    private double re, im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }

    /* Returns the string representation of this Complex number.
       The format of string is "Re + iIm" where Re is real part
       and Im is imagenary part.*/
    @Override
    public String toString() {
        return String.format(re + " + i" + im);
    }
}

// Driver class to test the Complex class
public class Main {
    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        System.out.println(c1);
    }
}
```

Output:

```
10.0 + i15.0
```

equals(Object obj)

- Parametre olarak aldığı nesneyi kendisini çağıran nesne ile karşılaştırır.

```
class Complex {
    private double re, im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }
}

// Driver class to test the Complex class
public class Main {
    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        Complex c2 = new Complex(10, 15);
        if (c1 == c2) {
            System.out.println("Equal ");
        } else {
            System.out.println("Not Equal ");
        }
    }
}
```

Output:

```
Not Equal
```

C1 ve c2 nesneleri farklı referansları temsil ettikleri için sonuç ‘not Equal’ olacaktır.

```java
class Complex {

    private double re, im;

    public Complex(double re, double im) {
        this.re = re;
        this.im = im;
    }

    // Overriding equals() to compare two Complex objects
    @Override
    public boolean equals(Object o) {

        // If the object is compared with itself then return true
        if (o == this) {
            return true;
        }

        /* Check if o is an instance of Complex or not
          "null instanceof [type]" also returns false */
        if (!(o instanceof Complex)) {
            return false;
        }

        // typecast o to Complex so that we can compare data members
        Complex c = (Complex) o;

        // Compare the data members and return accordingly
        return Double.compare(re, c.re) == 0
                && Double.compare(im, c.im) == 0;
    }
}
```

```java
// Driver class to test the Complex class
public class Main {

    public static void main(String[] args) {
        Complex c1 = new Complex(10, 15);
        Complex c2 = new Complex(10, 15);
        if (c1.equals(c2)) {
            System.out.println("Equal ");
        } else {
            System.out.println("Not Equal ");
        }
    }
}
```

Output:

```
Equal
```

Clone()

Manuel Klonlama

- Nesne klonlama, bir nesnenin tam bir kopyasının oluşturulmasını ifade eder. Geçerli nesnenin sınıfının yeni bir örneğini oluşturur ve tüm alanlarını tam olarak bu nesnenin karşılık gelen alanlarının içeriğiyle başlatır.

```java
// Java program to demonstrate that assignment operator
// only creates a new reference to same object
import java.io.*;

// A test class whose objects are cloned
class Test {
    int x, y;
    Test()
    {
        x = 10;
        y = 20;
    }
}
```

Output

```
10 20
100 20
100 20
```

```java
// Driver Class
class Main {
    public static void main(String[] args)
    {
        Test ob1 = new Test();

        System.out.println(ob1.x + " " + ob1.y);

        // Creating a new reference variable ob2
        // pointing to same address as ob1
        Test ob2 = ob1;

        // Any change made in ob2 will
        // be reflected in ob1
        ob2.x = 100;

        System.out.println(ob1.x + " " + ob1.y);
        System.out.println(ob2.x + " " + ob2.y);
    }
}
```

Clone()

- Nesnenin kopyasını oluşturmak için sınıfın, içinde veya üst sınıflarından birinde ortak bir clone() metodu olması gerekir.
- Clone() metodunu uygulayan her sınıf, klonlanmış nesne referansını elde etmek için super.clone()'u çağırmalıdır.
- Sınıf ayrıca, nesne klonunu oluşturmak istediğimiz Java.lang.Cloneable arayüzünü de implemente etmelidir, aksi takdirde o sınıfın nesnesi ile klon metodu çağrıldığında CloneNotSupportedException'ı hatası döner.

```java
// A Java program to demonstrate
// shallow copy using clone()
import java.util.ArrayList;

// An object reference of this class is
// contained by Test2
class Test {
    int x, y;
}

// Contains a reference of Test and
// implements clone with shallow copy.
class Test2 implements Cloneable {
    int a;
    int b;
    Test c = new Test();
    public Object clone() throws CloneNotSupportedException
    {
        return super.clone();
    }
}
```

Output

```
10 20 300 40
100 20 300 40
```

```java
// Driver class
public class Main {
    public static void main(String args[])
        throws CloneNotSupportedException
    {
        Test2 t1 = new Test2();
        t1.a = 10;
        t1.b = 20;
        t1.c.x = 30;
        t1.c.y = 40;

        Test2 t2 = (Test2)t1.clone();

        // Creating a copy of object t1
        // and passing it to t2
        t2.a = 100;

        // Change in primitive type of t2 will
        // not be reflected in t1 field
        t2.c.x = 300;

        // Change in object type field will be
        // reflected in both t2 and t1(shallow copy)
        System.out.println(t1.a + " " + t1.b + " " + t1.c.x
                           + " " + t1.c.y);
        System.out.println(t2.a + " " + t2.b + " " + t2.c.x
                           + " " + t2.c.y);
    }
}
```

Derin Kopyalama ve Sığ Kopyalama

- Sığ kopyalama, sadece bir nesneyi kopyalama yöntemidir ve klonlamada varsayılan olarak gerçekleşir. Bu yöntemde, bir X nesnesinin alanları yeni nesne Y'ye kopyalanır.
- Sığ kopyalamada nesnelerin sahip oldukları ilkel alanlar nesneye özgüdür ama referans tipler bu kopyalamada her nesne için bağımsız olmayacaktır.
- X nesnesinin derin bir kopyasını oluşturmak ve onu yeni bir Y nesnesine yerleştirmek istiyorsak, başvurulan nesne alanlarının yeni bir kopyası oluşturulur ve bu başvurular Y nesnesine yerleştirilir. Bu, nesnedeki başvurulan nesne alanlarında yapılan herhangi bir değişiklik özgündür. Yani birinde yapılacak bir değişiklik diğer nesneyi etkilemeyecektir.
- Derin kopya, tüm alanları kopyalar ve alanların gösterdiği dinamik olarak ayrılmış belleğin kopyalarını oluşturur. Bir nesne, atıfta bulunduğu nesnelerle birlikte kopyalandığında derin bir kopya oluşur.

```java
// A Java program to demonstrate
// deep copy using clone()

// An object reference of this
// class is contained by Test2
class Test {
    int x, y;
}

// Contains a reference of Test and
// implements clone with deep copy.
class Test2 implements Cloneable {
    int a, b;

    Test c = new Test();

    public Object clone() throws CloneNotSupportedException
    {
        // Assign the shallow copy to
        // new reference variable t
        Test2 t = (Test2)super.clone();

        // Creating a deep copy for c
        t.c = new Test();
        t.c.x = c.x;
        t.c.y = c.y;

        // Create a new object for the field c
        // and assign it to shallow copy obtained,
        // to make it a deep copy
        return t;
    }
}
```

```java
public class Main {
    public static void main(String args[])
        throws CloneNotSupportedException
    {
        Test2 t1 = new Test2();
        t1.a = 10;
        t1.b = 20;
        t1.c.x = 30;
        t1.c.y = 40;

        Test2 t3 = (Test2)t1.clone();
        t3.a = 100;

        // Change in primitive type of t2 will
        // not be reflected in t1 field
        t3.c.x = 300;

        // Change in object type field of t2 will
        // not be reflected in t1(deep copy)
        System.out.println(t1.a + " " + t1.b + " " + t1.c.x
                           + " " + t1.c.y);
        System.out.println(t3.a + " " + t3.b + " " + t3.c.x
                           + " " + t3.c.y);
    }
}
```

Output

```
10 20 30 40
100 20 300 40
```

Clone() metodu

- Başka bir referans değişkenine bir nesne referansı atamak için atama operatörünü kullanırsak, o zaman eski nesnenin aynı adres konumuna işaret edilir ve nesnenin yeni bir kopyası oluşmayacaktır. Bu nedenle, referans değişkenindeki herhangi bir değişiklik orijinal nesneye yansıtılacaktır.
- Bir kopya yapıcı metot kullanırsak, tüm verileri açıkça kopyalamamız gerekir, yani sınıfın tüm alanlarını yapıcıda açıkça yeniden atamamız gerekir. Ancak klon yönteminde, yeni bir kopya oluşturma işi, yöntemin kendisi tarafından yapılır. Bu yüzden fazladan işlemden kaçınmak için nesne klonlama kullanılır.

hashCode()

- JVM her nesne için benzersiz bir ***hashcode*** üretir. Farklı nesneler için farklı tamsayılar döndürür.
- hashcode() yönteminin nesnenin adresini döndürdüğü gibi yanlış bir algı vardır.
- hashcode() kod nesnenin yerel adresini kullanarak her nesne için benzersiz bir tamsayı üretir.
- Bir koleksiyondaki nesneyi aramak için kullanılan hashcode arama performansı için kritiktir.
- JVM(Java Virtual Machine), nesneleri HashSet, HashMap, Hashtable vb. gibi hash veri yapılarına kaydederken hashcode yöntemini kullanır.
- Nesneleri hash koda dayalı olarak kaydetmenin ana avantajı, aramanın kolaylaşmasıdır.

```java
// Java program to demonstrate working of
// hashCode() and toString()
public class Student
{
    static int last_roll = 100;
    int roll_no;

    // Constructor
    Student()
    {
        roll_no = last_roll;
        last_roll++;
    }

    // Overriding hashCode()
    @Override
    public int hashCode()
    {
        return roll_no;
    }

    // Driver code
    public static void main(String args[])
    {
        Student s = new Student();

        // Below two statements are equivalent
        System.out.println(s);
        System.out.println(s.toString());
    }
}
```

1. Output :

```
Student@64
Student@64
```

getClass()

- Kendisini çağıran nesnenin sınıf nesnesini döndürür ve nesnenin gerçek çalışma zamanı sınıfını almak için kullanılır.
- Nesne sınıfının meta verilerini almak için de kullanılabilir.
- Override edilemez.

```
// Java program to demonstrate working of getClass()
public class Test
{
    public static void main(String[] args)
    {
        Object obj = new String("GeeksForGeeks");
        Class c = obj.getClass();
        System.out.println("Class of Object obj is : "
                           + c.getName());
    }
}
```

1. Output:

```
Class of Object obj is : java.lang.String
```

KAYNAKLAR

- Java Tutorial | Learn Java - javatpoint. (2021, March 21). Retrieved from https://www.javatpoint.com/java-tutorial
- Object class in Java - GeeksforGeeks. (2021, June 03). Retrieved from https://www.geeksforgeeks.org/object-class-in-java