

---

# Üst Düzey Programlama

REFLECTION  
(YANSIMA)

# Reflection API

## Neler yapılır ?

- \* Bir nesnenin sınıfını tespit edebiliriz
- \* Sınıfın metotları, alanları, yapıcıları ve süper sınıfları hakkında bilgi alabiliriz.
- \* Çalışma anına kadar ismi belirli olmayan bir sınıftan örnek oluşturabiliriz.
- \* Nesnenin alanlarına değer atayıp, değerleri alabiliriz.
- \* Çağrılacak olan metodun ismi çalışma anına kadar bilinmese dahi, çalıştırılabilir.
- \* Tipi ve boyutu çalışma anına kadar bilinmeyen diziler oluşturulabiliriz.

---

# Sınıfları incelemek

Java reflection API si için aşağıdaki pakette bulunan sınıflar kullanılır.

```
java.lang.reflect.*;
```

# Sınıfları incelemek

Örneğin bir sınıfın alanlarını, metotlarını, yapıcılarını listelemek, göstermek istiyorsunuz. Bu iş için **Class** nesnesine ihtiyacınız vardır.

Bu nesne reflection paketinde tanımlı bir nesnedir. Class nesnesinin metotlarını kullanarak **Constructor, Method, Field** gibi sınıfları geri alabilir ve sınıf hakkında ilgi edinebilirsiniz.

# Sınıfları incelemek

Herhangi bir nesnenin Class sınıfını alabilmek için nesne.getClass() metodu kullanılır.

```
Class klass = nesne.getClass();
```

```
Class b = java.awt.Button.class;
```

```
Class c = Class.forName("sınıfın adı");
```

# Sınıfları incelemek

```
AAA.java x
package reflectionornek;

public class AAA {

    private String a;
    private int b;
    public double c;

    public String toString(){
        return "aaa";
    }

    public void setA(String a){
        this.a=a;
    }

    public String getA(){
        return this.a;
    }
}
```

# Sınıfları incelemek

```
Main.java x
package reflectionornek;

import reflectionornek.AAA;

public class Main {

    public static void main(String[] args) {
        AAA a = new AAA();
        a.setA("AAA");
        Class c = a.getClass();

        String cName = c.getName();
        System.out.println(cName);
    }
}
```

# Sınıfları incelemek

```
SamModifier.java x
package reflectionornek;

import java.lang.reflect.Modifier;

public class SamModifier {
    public static void main(String[] args) {
        AAA a = new AAA();

        Class c = a.getClass();
        int m = c.getModifiers();

        if (Modifier.isAbstract(m)) {
            System.out.println("abstract");
        }
        if (Modifier.isPublic(m)) {
            System.out.println("public");
        }
        if (Modifier.isFinal(m)) {
            System.out.println("final");
        }
    }
}
```

# Süper Sınıf

```
SuperSinif.java x
package reflectionornek;

public class SuperSinif {

    public static void main(String[] args) {
        AAA a = new AAA();

        Class c = a.getClass();

        Class sc = c.getSuperclass();

        System.out.println(sc.getName());
    }
}
```

# Interface leri öğrenme

```
MyInterface.java x  
package reflectionornek;  
  
public interface MyInterface {  
    public void yaz();  
}
```

```
BBB.java x  
package reflectionornek;  
  
public class BBB implements MyInterface {  
    public void yaz() {  
        System.out.println("BBB");  
    }  
}
```

```
ListInterfaces.java x  
package reflectionornek;  
  
public class ListInterfaces {  
    public static void main(String[] args) {  
        BBB b = new BBB();  
  
        Class c = b.getClass();  
  
        Class[] interfaces = c.getInterfaces();  
  
        for (Class inter : interfaces) {  
            System.out.println(inter.getName());  
        }  
    }  
}
```

# Sınıfın Alanlarını Öğrenme

```
FieldsList.java x
package reflectionornek;

import java.lang.reflect.Field;

public class FieldsList {
    public static void main(String[] args) {
        AAA a = new AAA();

        Class c = a.getClass();

        Field[] alanlar = c.getFields();

        for (Field elem : alanlar) {
            String alanAdi = elem.getName();
            Class alanTipi = elem.getType();

            System.out.print(alanAdi);
            System.out.print(":");
            System.out.println(alanTipi.getName());
        }
    }
}
```

# Metotları Öğrenme

```
MethodList.java x
package reflectionornek;

import java.lang.reflect.Method;

public class MethodList {

    public static void main(String[] args) {
        Class c;
        try {
            c = Class.forName("reflectionornek.AAA");

            Method[] metotlar = c.getMethods();

            for (Method elem : metotlar) {
                System.out.println("-----");
                System.out.println("Method Adı:"+elem.getName());
                System.out.println("Return Tipi:"+elem.getReturnType().getName());

                Class[] parametreler = elem.getParameterTypes();
                System.out.println("-----parametreler-----");
                for (Class param : parametreler) {
                    System.out.println(param.getName());
                }
            }

        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
    }
}
```

# Çalışma anında yüklenen sınıftan nesne oluşturma

```
RuntimeObjectCreate.java x
package reflectionornek;

import reflectionornek.*;

public class RuntimeObjectCreate {
    public static void main(String[] args) {
        MyInterface mi = (MyInterface) createNesne("reflectionornek.BBB");

        mi.yaz();
    }

    public static Object createNesne(String sinif){
        Class c=null;
        Object o=null;
        try {
            c = Class.forName(sinif);
            o = c.newInstance();
        } catch (IllegalAccessException ex) {
            ex.printStackTrace();
        } catch (InstantiationException ex) {
            ex.printStackTrace();
        } catch (ClassNotFoundException ex) {
            ex.printStackTrace();
        }
        return o;
    }
}
```

# Sınıf Alanlarının Değerini Alma

```
GetFieldValue.java x
package reflectionornek;

import java.lang.reflect.Field;

public class GetFieldValue {

    public static void main(String[] args) {
        AAA a = new AAA();
        a.c = 10.0;

        Class c = a.getClass();
        Field cField=null;
        Double cValue=null;
        try {
            cField = c.getField("c");
            cValue = (Double)cField.get(a);
            System.out.println("degeri = " + cValue.doubleValue());
        } catch (SecurityException ex) {
            ex.printStackTrace();
        } catch (NoSuchFieldException ex) {
            ex.printStackTrace();
        } catch (IllegalAccessException ex) {
            ex.printStackTrace();
        }
    }
}
```

# Sınıf Alanlarına Değer Atama

```
SetFieldValue.java x
package reflectionornek;

import java.lang.reflect.Field;

public class SetFieldValue {

    public static void main(String[] args) {
        AAA a = new AAA();

        Class c = a.getClass();
        Field cField=null;
        Double cValue = new Double(11.22);
        try {

            cField = c.getField("c");
            cField.set(a,cValue);
        } catch (SecurityException ex) {
            ex.printStackTrace();
        } catch (NoSuchFieldException ex) {
            ex.printStackTrace();
        } catch (IllegalAccessException ex){
            ex.printStackTrace();
        }

        System.out.println("a.c = " + a.c);
    }
}
```

# Sınıf Metotlarının Çağrılması

```
CallMethod.java x
package reflectionornek;

import ...

public class CallMethod {

    public static void main(String[] args) {
        Class c=null;
        AAA a = new AAA();
        try {
            c = a.getClass();
            String aValue = "abc";
            Class[] parametreTipleri = new Class[]{String.class};
            Object[] parametreDegerleri = new Object[] {aValue};

            Method setAMetodu = c.getMethod("setA",parametreTipleri);
            setAMetodu.invoke(a,parametreDegerleri);

        } catch(NoSuchMethodException ex){
            ex.printStackTrace();
        } catch(IllegalAccessException ex){
            ex.printStackTrace();
        } catch (InvocationTargetException ex){
            ex.printStackTrace();
        }

        System.out.println("a degeri =" + a.getA());
    }
}
```

# Sınıf Metotlarının Çağrılması

```
CallMethod.java x
package reflectionornek;

import ...

public class CallMethod {

    public static void main(String[] args) {
        Class c=null;
        AAA a = new AAA();
        try {
            c = a.getClass();
            String aValue = "abc";
            Class[] parametreTipleri = new Class[]{String.class};
            Object[] parametreDegerleri = new Object[] {aValue};

            Method setAMetodu = c.getMethod("setA",parametreTipleri);
            setAMetodu.invoke(a,parametreDegerleri);

            Method getAMetodu = c.getMethod("getA",new Class[]{});
            System.out.println( getAMetodu.invoke(a,new Object[]{}));

        } catch (NoSuchMethodException ex){
            ex.printStackTrace();
        } catch (IllegalAccessException ex){
            ex.printStackTrace();
        } catch (InvocationTargetException ex){
            ex.printStackTrace();
        }

        System.out.println("a degeri =" + a.getA());
    }
}
```

---

# Üst Düzey Programlama

REFLECTION  
(YANSIMA)