Java Reflection API

Contoh-contoh

http://java.sun.com/developer/technicalArticles/ALT/Reflection/index.html#8

DumpMethods

```
public class DumpMethods {
    public static void main(String[] args) {
         try {
            Class c = Class.forName("java.util.Stack");
            Method m[] = c.getDeclaredMethods();
            for (int i = 0; i < m.length; i++)
                System.out.println(m[i].toString());
         }
         catch (Throwable e) {
            System.err.println(e);
         }
    }
}
```

Simulating the instanceof Operator

```
public class InstanceOf {
      public static void main(String args[])
      {
         try {
            Class cls = Class.forName("reflection.B");
            boolean b1
              = cls.isInstance(new Integer(37));
            System.out.println(b1);
            boolean b2 = cls.isInstance(new B());
            System.out.println(b2);
         }
         catch (Throwable e) {
            System.err.println(e);
         }
      }
   }
```

Finding Out About Methods of a Class

```
public class Method1 {
     private int f1(Object p, int x) throws NullPointerException
     {
        if (p == null) throw new NullPointerException();
        return x;
     }
     public static void main(String args[])
     {
        try {
          Class cls = Class.forName("reflection.Method1");

           Method methlist[] = cls.getDeclaredMethods();
           for (int i = 0; i < methlist.length; i++) {
              Method m = methlist[i];
              System.out.println("name = " + m.getName());
              System.out.println("decl class = " + m.getDeclaringClass());
              Class pvec[] = m.getParameterTypes();
              for (int j = 0; j < pvec.length; j++)
                 System.out.println("param #" + j + " " + pvec[j]);
              Class evec[] = m.getExceptionTypes();
              for (int j = 0; j < evec.length; j++)
                 System.out.println("exc #" + j + " " + evec[j]);
              System.out.println("return type = " + m.getReturnType());
              System.out.println("-----");
           }
        }
        catch (Throwable e) {
           System.err.println(e);
        }
     }
  }
```

Obtaining Information About Constructors

```
public class constructor1 {
     public constructor1() {}
     protected constructor1(int i, double d){}
     public static void main(String args[])
     {
        try {
          Class cls = Class.forName("reflection.constructor1");

          Constructor ctorlist[]= cls.getDeclaredConstructors();
         for (int i = 0; i < ctorlist.length; i++) {
              Constructor ct = ctorlist[i];
              System.out.println("name = " + ct.getName());
              System.out.println("decl class = " + ct.getDeclaringClass());
              Class pvec[] = ct.getParameterTypes();
              for (int j = 0; j < pvec.length; j++)
                 System.out.println("param #" + j + " " + pvec[j]);
              Class evec[] = ct.getExceptionTypes();
              for (int j = 0; j < evec.length; j++)
                 System.out.println("exc #" + j + " " + evec[j]);
              System.out.println("-----");
           }
         }
         catch (Throwable e) {
            System.err.println(e);
         }
     }
  }
```

Finding Out About Class Fields

```
public class field1 {
    private double d;
    public static final int i = 37;
    String s = "testing";

    public static void main(String args[])
    {
       try {
          Class cls = Class.forName("reflection.field1");

          Field fieldlist[]
            = cls.getDeclaredFields();
          for (int i = 0; i < fieldlist.length; i++) {
             Field fld = fieldlist[i];
             System.out.println("name= " + fld.getName());
             System.out.println("decl class = " + fld.getDeclaringClass());
             System.out.println("type = " + fld.getType());
             int mod = fld.getModifiers();
             System.out.println("modifiers = " + Modifier.toString(mod));
             System.out.println("-----");
          }
        }
        catch (Throwable e) {
           System.err.println(e);
        }
     }
 }
```

Invoking Methods by Name

```
public class method2 {
   public int add(int a, int b)
   { return a + b;
   }

   public static void main(String args[])
   {
      try {
        Class cls = Class.forName("reflection.method2");
        Class partypes[] = new Class[2];
         partypes[0] = Integer.TYPE;
         partypes[1] = Integer.TYPE;
         Method meth = cls.getMethod("add", partypes);
         method2 methobj = new method2();
         Object arglist[] = new Object[2];
         arglist[0] = new Integer(37);
         arglist[1] = new Integer(47);
         Object retobj = meth.invoke(methobj, arglist);
         Integer retval = (Integer)retobj;
         System.out.println(retval.intValue());
      }
      catch (Throwable e) {
         System.err.println(e);
      }
   }
}
```

Creating New Objects

```
public class constructor2 {
      public constructor2()
      {
      }
      public constructor2(int a, int b)
      {
         System.out.println(
           "a = " + a + " b = " + b);
      }
      public static void main(String args[])
      {
         try {
           Class cls = Class.forName("reflection.constructor2");
           Class partypes[] = new Class[2];
            partypes[0] = Integer.TYPE;
            partypes[1] = Integer.TYPE;
            Constructor ct = cls.getConstructor(partypes);
            Object arglist[] = new Object[2];
            arglist[0] = new Integer(37);
            arglist[1] = new Integer(47);
            Object retobj = ct.newInstance(arglist);
         }
         catch (Throwable e) {
            System.err.println(e);
         }
      }
   }
```

Changing Values of Fields

```
public class field2 {
      public double d;

      public static void main(String args[])
      {
         try {
            Class cls = Class.forName("reflection.field2");
            Field fld = cls.getField("d");
            field2 f2obj = new field2();
            System.out.println("d = " + f2obj.d);
            fld.setDouble(f2obj, 12.34);
            System.out.println("d = " + f2obj.d);
         }
         catch (Throwable e) {
            System.err.println(e);
         }
      }
   }
```

Using Arrays

```
public class array1 {
     public static void main(String args[])
     {
        try {
           Class cls = Class.forName("java.lang.String");
           Object arr = Array.newInstance(cls, 10);
           Array.set(arr, 5, "this is a test");
           String s = (String)Array.get(arr, 5);
           System.out.println(s);
        }
        catch (Throwable e) {
           System.err.println(e);
        }
     }
  }
```