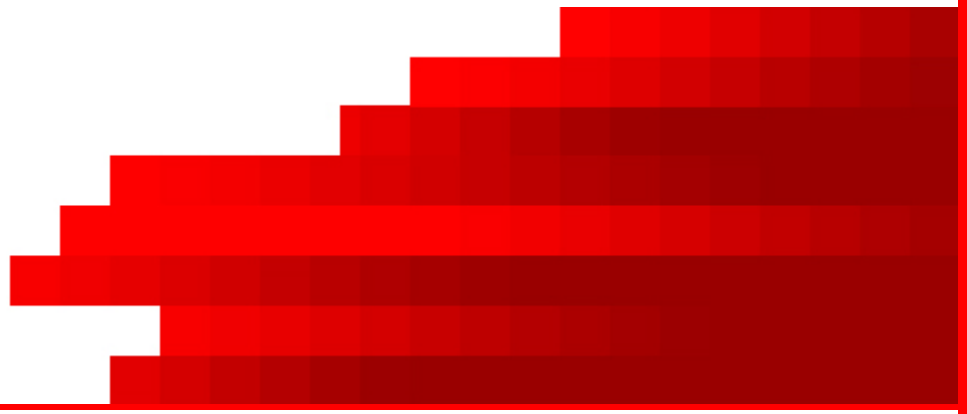
Reflection API



HSBC Technology and Services

HSBC
The world's local bank

RESTRICTED

Uses of Reflection

Reflection is commonly used by programs which require the ability to examine or modify the runtime behavior of applications running in the Java virtual machine.

This is a relatively advanced feature and should be used only by developers who have a strong grasp of the fundamentals of the language.

Drawbacks of Reflection

Reflection is powerful, but should not be used indiscriminately. If it is possible to perform an operation without using reflection, then it is preferable to avoid using it. The following concerns should be kept in mind when accessing code via reflection.

Performance Overhead

Because reflection involves types that are dynamically resolved, certain Java virtual machine optimizations can not be performed. Consequently, reflective operations have slower performance than their non-reflective counterparts, and should be avoided in sections of code which are called frequently in performance-sensitive applications.

Security Restrictions

Reflection requires a runtime permission which may not be present when running under a security manager. This is in an important consideration for code which has to run in a restricted security context, such as in an Applet.

Exposure of Internals

Since reflection allows code to perform operations that would be illegal in non-reflective code, such as accessing private fields and methods, the use of reflection can result in unexpected side-effects, which may render code dysfunctional and may destroy portability. Reflective code breaks abstractions and therefore may change behavior with upgrades of the platform.

Reflection Example—The Java Reflection API

• For every loaded class, the Java Runtime Environment (JRE) maintains an associated `Class` object

– The Class object “reflects” the class it represents

– Can use the `Class` object to discover information about a loaded class

• name

• modifiers (public, abstract, final)

• superclasses

• implemented interfaces

• fields

• methods

• constructors

– Can instantiate classes and invoke their methods via `Class` object

}

How the Java Reflection API works:

• Accessing the Class object for a loaded class:

To get the `Class` object for an object `mystery`:

```
Class c = mystery.getClass();
```

Or, using the class name:

```
Class c = Class.forName("mysteryClass");
```

Can also get the superclass of `MysteryClass`:

```
Class s = c.getSuperclass();
```

}

Java Reflection--Continued

Introspecting (examining) a class via its `Class` object:

Getting the class name:

```
Class c = mysteryObject.getClass();
String s = c.getName();
```

}

Discovering the interfaces implemented by a class:

```
Class[] interfaces = c.getInterfaces();
```

Discovering the fields of a class:

```
Field[] fields = c.getFields();
```

Discovering the methods of a class:

```
Method[] methods = c.getMethods();
```

Example Code:

```
static void showMethods(Object o) {
Class c = o.getClass();
Method[] theMethods = c.getMethods();
for (int i = 0; i < theMethods.length; i++) {
String methodString = theMethods[i].getName();
System.out.println("Name: " + methodString);
String returnString = theMethods[i].getReturnType().getName();
System.out.println(" Return Type: " + returnString);
Class[] parameterTypes = theMethods[i].getParameterTypes();
System.out.print(" Parameter Types:");
for (int k = 0; k < parameterTypes.length; k ++) {
String parameterString = parameterTypes[k].getName();
System.out.print(" " + parameterString);
}
System.out.println();
}
}
```

}

Example--Continued

Output for a call: of the form:

Polygon P = new Polygon();

showMethods(p);

Name: equals

Return Type: boolean

Parameter Types: java.lang.Object

Name: getClass

Return Type: java.lang.Class

Parameter Types:

Name: intersects

Return Type: boolean

Parameter Types: double double double double

}

Additional Features of Java Reflection

- Can obtain constructors for a class
- Can instantiate objects and invoke methods via information obtained from the reflection API.

Creating object by obtaining name:

SampleNoArg.java

HSBC

The world's local bank

About HSBC Technology and Services

HSBC Technology and Services (HTS) is a pivotal part of the Group and seamlessly integrates technology platforms and operations with an aim to re-define customer experience and drive down unit cost of production. Its solutions connect people, devices and networks across the globe and combine domain expertise, process skills and technology to deliver unparalleled business value, thereby enabling HSBC to stay ahead of competition by addressing market changes quickly and developing profitable customer relationships.

Presenter's Contact Details:
Name: Lorem ipsum
Role: Lorem ipsum
Direct: +00 00 000
Email: loremipsum@hsbc.com

Name: Lorem ipsum
Role: Lorem ipsum
Direct: +00 00 000
Email: loremipsum@hsbc.com

Restricted for company use only