Reflection

April 19, 2006

• Remember project 1?
  – Used reflection to find module class
    • http://webserver:port/HelloWorld/Foo

• We’ll cover some basics of reflection
  – Highly condensed version of
    • Reflection: Java Technology’s Secret Weapon, by Odendahl (on web page)
What is Reflection?

- Makes classes, methods, and fields objects that can be manipulated at run time
  - Can determine fields and methods of class
  - Can instantiate class given a String containing its name
  - Can invoke methods given a String with name
  - Can create classes at runtime

What Reflection Isn’t

- Doesn’t add any power to the language
  - Given access to all the source code

- Not the solution to every problem
  - Use sparingly, if at all
java.lang.Class

- Object of type Class represents a class
  - Useful for
    - Making instances of a class
    - Getting information about fields/methods
  - Most uses of reflection start with a Class
- Primitive types also have a Class
  - e.g., int.class

Getting Some Class

- Use Object method getClass()
  - Class c = “hello”.getClass();
- Use class literal
  - Class c = String.class;
- Use the class name
  - Class c = Class.forName(“java.lang.String”)
Making Objects

• Class object for no-arg constructor
  ```java
  Class c;
  Foo f = (Foo) c.newInstance();
  ```

• Constructor object otherwise
  ```java
  Class c;
  Class[] cArg = { String.class };  
  Constructor cons = c.getConstructor(cArg);
  Object[] consArg = { "hello" };
  Foo f = (Foo) cons.newInstance(consArg);
  ```

Working with Fields

• Can get Field objects from Class
  – Can also get all fields in Class
  ```java
  Class c = x.getClass(); // get class of obj x
  Field f = c.getField("foo");
  ...(Type-of-foo) f.get(x); 
  ...(Type-of-foo) f.set(x, value);
  ```
Invoking Methods

• Get from Class object
  – Invoke just like constructor

\[
\begin{align*}
\text{Class } c &= \text{x.getClass()}; \\
\text{Class[]} \ c\text{Arg} &= \{ \text{String.class } \}; \\
\text{Method } m &= c.\text{getMethod("bar", cArg)}; \\
\text{Object[]} \ m\text{Arg} &= \{ \text{"hello" } \}; \\
\text{Foo } f &= (\text{m-result-type}) \text{ m.invoke(x, mArg);}
\end{align*}
\]

Putting It All Together

• Example from JavaOne slides:

```java
public static void main(String [] args)
  throws Exception {
  Field f = System.class.getField("out");
  PrintStream out = (PrintStream) f.get(null);
  Class[] paramTypes = { String.class };
  Method m = PrintStream.class.getMethod
    ("println", paramTypes);
  String[] params = (String[]) Array.newInstance
    (String.class, 1);
  Array.set(params, 0, "Hello, world!");
  m.invoke(out, params);
}
```