Comp 401
Reflection and Action Objects

Instructor: Prasun Dewan

Prerequisites

- Inheritance

Topics

- Type Object
  - Object representing a type of objects
- Action Object
  - Object representing an action such as “Do”

printProperties

```
public static void main(String[] args) {
        BMISpreadsheet bmi = new ABMISpreadsheet(1.77, 75);
        printProperties(bmi);
        Point point = new ACartesianPoint(50, 34);
        printProperties(point);
}
```

```
public static void printProperties(Object object) {
    …
}
```

PropertiesPrinter

```
<terminated> APropertiesPrinter [Java Application] C:\Program Files\Java\jre1.6.0_03\bin\
Properties of:examples.style.ABMISpreadsheet@b1b4c3
Weight:75.0
Height:1.77
BMI:23.93948099205209
Class:class examples.style.ABMISpreadsheet

Properties of:examples.loops.ACartesianPoint@47393f
X:50
Y:34
Angle:0.5971766580926776
Radius:60.4648658313239
Class:class examples.loops.ACartesianPoint
```

Calling printProperties

```
public static void main(String[] args) {
        BMISpreadsheet bmi = new ABMISpreadsheet(1.77, 75);
        printProperties(bmi);
        Point point = new ACartesianPoint(50, 34);
        printProperties(point);
}
```

printProperties() accepts an argument of arbitrary type

Class Reflection

```
public static void printProperties(Object object) {
    System.out.println("Properties of:" + object);
    Class objectClass = object.getClass();
    Method[] methods = objectClass.getMethods();
    Object[] nullArgs = {};
    for (int index = 0; index < methods.length; index++) {
        Method method = methods[index];
        if (isGetter(method)) {
            Object retVal = methodInvoke(object, method,
nullArgs);
            System.out.println(propertyName(method) + ":" +
retVal);
        }
    }
        System.out.println();
}
```

Class is a runtime variable vs. compile time as in generics.

Class Reflection: Invoking methods on a class to create new instance or learn its properties.

Method Reflection

```
public static String GETTER_PREFIX = "get";
public static boolean isGetter (Method method) {
        return  method.getParameterTypes().length == 0 &&
                method.getReturnType() != Void.TYPE &&
                method.getName().startsWith(GETTER_PREFIX);
}
```

```
public static String propertyName(Method getter) {
  return getter.getName().substring(GETTER_PREFIX.length());
}
```

Method Reflection: Invoking methods on a method.

Invoking Target Method

```
public static Object  methodInvoke(Object object, Method
method, Object[] args ) {
        try {
                return  method.invoke(object, args);
        } catch (IllegalAccessException e) {
                e.printStackTrace();
                return null;
        } catch (InvocationTargetException e ) {
                e.printStackTrace();
                return null;
        }
}
```

IllegalAccessException: Method not visible to caller.

InvocationTargetException: Exception thrown when parameters/target object do not match method or method throws exception such as ClassCastException.

Method that takes method as a parameter is 2nd-order method.

Reflectable Type Object

Type Object = Embedded Type

- newInstance()
- getName()
- getMethods()
- getInterfaces()
- getSuperType()
- getSubTypes()

Provides reflection operations to learn properties of the action such as return type, name, and parameter types.

A supertype does not reference its subtypes. Subtypes can, can be developed all over the world, and thus can never be known.

Need not correspond to a compiled class. Can be an XML schema, for instance.

Action Object

execute (targetObject, params)

Action Object = Embedded Operation

Provides an execute operation to perform some action.

The execute operation takes as parameters the object on which the target operation is to invoked and an array of parameters of the target method.

Reflectable Action Object

Action Object = Embedded Operation

execute (targetObject, params)

getParameterTypes()

getReturnType()

getName ()

Provides an execute operation to perform some action.

Provides additional reflection operations to learn properties of the action such as return type, name, and parameter types.

The execute operation takes the object on which the target operation is to invoked and an array of parameters of the target method.

Need not correspond to a compiled method. Can be a Web Ser vice action described by XML.