Reflection

see Sun's tutorial

Reflection API

- allows fully dynamic access to internal stuff
  - actual class of an object
  - all kind of info about a class
  - as well as interfaces
  - create instances without knowing the class at compile time
  - get and set field values (without ...)
  - invoke methods (without ...)
  - create and manipulate arrays (without ...)

Retrieving Class Objects

- Class c = x.getClass();
- Class s = c.getSuperClass();
- Class t = java.awt.Button.class;
- Class u = Class.forName(“java.awt.Button”);

Inspecting Classes

```
static void printClassName(Object o) {
    Class c = o.getClass();
    String s = c.getName();
    System.out.println(s);
}
static void printModifiers(Object o) {
    int m = o.getClass().getModifiers();
    if (Modifier.isPublic(m)) ...
}
```

All superclasses

```
static List allSuperClasses(Object o) {
    List superClasses = new ArrayList();
    Class subClass = o.getClass();
    Class superClass = subClass.getSuperClass();
    while (superClass != null) {
        superClasses.add(superClass);
        subClass = superClass;
        superClass = subClass.getSuperClass();
    }
    return superClasses;
}
```

All SuperClasses (recursive)

```
static List allSuperClasses(Object o) {
    return allSuperClasses(o.getClass().getSuperClass(),
       new ArrayList());
}
static List allSuperClasses(Class c, List all) {
    if (c == null) return all;
    all.add(c);
    return allSuperClasses(c.getSuperClass(), all);
}
```

Interfaces

Interfaces are represented by class Class as well

Class[] myInterfaces = o.getClass.getInterfaces();

i.e. there is no separate class for Interfaces, but a test:

isInterface();

can use getFields() and getMethods() like with classes

Retrieving field info

```
static void printFieldNames(Object o)
    Field[] publicFields = o.getClass().getFields();
    for(int i = 0; i<publicFields.length; i++) {
        String fieldName = publicFields[i].getName();
        Class typeClass = publicFields[i].getType();
        String typeName = typeClass.getName();
        System.out.println("Name: " + fieldName + ", Type: " +
                                    typeName);
    }
}
```

Constructors

```
Constructor[] cons = c.getConstructors();
for(int i = 0; i<cons.length; i++) {
    Class[] paramTypes = cons[i].getParameterTypes();
    ...
}
```

also:

```
for(int i = 0; i<cons.length; i++) {
    Class[] exTypes = cons[i].getExceptionTypes();
    ...
}
```

Methods

```
Method[] methods = c.getMethods();
for(int i = 0; i<methods.length; i++) {
    String name = methods[i].getName();
    Class returnType = methods[i].getReturnType();
    Class[] paramTypes = methods[i].getParameterTypes();
    Class[] exTypes = methods[i].getExceptionTypes();
    int modifiers = methods[i].getModifiers();
    ...
}
```