RTTI and Introspection

- **Run-time type identification** make it possible to determine the type of an object
  - E.g. given a pointer to a base class, determine the derived class of the pointed object
  - The type (class) must be known at compile time
- **Introspection** makes general class information available at run-time
  - The type (class) does not have to be known at compile time
  - This is very useful in component architectures and visual programming
  - E.g. list the attributes of an object
RTTI and Introspection

- RTTI and introspection are powerful programming language features
  - They enable some powerful design techniques
  - We will discuss them in the context of Java

- This discussion will follow Chapter 11 in *Thinking in Java* by Bruce Eckel
  - By the way, this is an excellent book freely available online

The need for RTTI

Polymorphism Example

```java
// Shapes.java
package c11;
import java.util.*;
interface Shape {
    void draw();
}
class Circle implements Shape {
    public void draw() {
        System.out.println("Circle.draw()");
    }
}
class Square implements Shape {
    public void draw() {
        System.out.println("Square.draw()");
    }
}
class Triangle implements Shape {
    public void draw() {
        System.out.println("Triangle.draw()");
    }
}
public class Shapes {
    public static void main(String[] args) {
        Vector s = new Vector();
        s.addElement(new Circle());
        s.addElement(new Square());
        s.addElement(new Triangle());
        Enumeration e = s.elements();
        while(e.hasMoreElements()) {
            ((Shape)e.nextElement()).draw();
        }
    }
}
```

Upcasting (Type Safe in Java)

What if you want to know the exact type at run-time?
### The Class Object

- Type information is available at run-time in Java
- There is a *Class object* for each class in the program
  - It stores class information
- Class objects are loaded in memory the first time they are needed
  - A Java program is not completely loaded before it begins!
- The class `Class` provides a number of useful methods for RTTI
  - [http://java.sun.com/j2se/1.3/docs/api/java/lang/Class.html](http://java.sun.com/j2se/1.3/docs/api/java/lang/Class.html)

### Example

```java
class Candy {
    static {
        System.out.println("Loading Candy");
    }
}

class Gum {
    static {
        System.out.println("Loading Gum");
    }
}

class Cookie {
    static {
        System.out.println("Loading Cookie");
    }
}

public class SweetShop {
    public static void main(String[] args) {
        System.out.println("inside main");
        new Candy();
        System.out.println("After creating Candy");
        try {
            Class.forName("Gum");
        } catch (ClassNotFoundException e) {
            e.printStackTrace();
        }
        System.out.println("After creating Gum");
        new Cookie();
        System.out.println("After creating Cookie");
    }
}
```

Executed at Load Time

Returns a reference to class Gum
Example

• Output
  – JVM-1

inside main
Loading Candy
After creating Candy
Loading Gum
After Class.forName("Gum")
Loading Cookie
After creating Cookie

Example

• Output
  – JVM-2

Loading Candy
Loading Cookie
inside main
After creating Candy
Loading Gum
After Class.forName("Gum")
After creating Cookie
The Class Object

- Class literals also provide a reference to the Class object
  - E.g. Gum.class

- Each object of a primitive wrapper class has a
  standard field called TYPE that also provides a
  reference to the Class object

RTTI

- The type of an object can be determined using the
  \texttt{instanceof} keyword
  - See PetCount.java
  - It can be rewritten using a Class literal, see PetCount2.java
  - Notice that an object of a derived class is an instance of the
    its base classes (\textit{i.e.} any predecessor in the inheritance
    hierarchy)

- RTTI is very useful when reusing classes without
  extending them

- \texttt{Class.isInstance()} also implements the
  \texttt{instanceof} functionality
Introspection

- *Introspection* makes general class information available at run-time
  - The type (class) does not have to be known at compile time
  - E.g. list the attributes of an object
- This is very useful in
  - Rapid Application Development (RAD)
    - Visual approach to GUI development
    - Requires information about component at run-time
  - Remote Method Invocation (RMI)
    - Distributed objects

Reflection

- Java supports introspection through its reflection library
  - See classes Field (attributes), Method and Constructor
- Examples:
  - ShowMethods.java
**Python**

- The Inspect module provides introspection mechanism
  - See:
    - `getmembers(object[, predicate])`
    - `getsource(object)`
    - `getclasstree(classes[, unique])`
    - `getmro(cls)`

**Java Beans**

- Tutorial
- The JavaBeans API makes it possible to write component software in the Java programming language.
- **Components** are self-contained, reusable software units that can be visually composed into composite components, applets, applications, and servlets using visual application builder tools.
- JavaBean components are known as Beans.
Demonstration

- BeanBox application

  The BeanBox is a simple tool you can use to test your Beans, and to learn how to visually manipulate their properties and events. The BeanBox is not a builder tool. You'll use the BeanBox to learn about Beans.
Reading Assignment

- Bruce Eckel *Thinking in Java*
  - Chapter 11, RTTI

- Java Beans
  - Tutorial
  - Play with the BeanBox