CS 3723/3721 Programming Languages
Java Stack Class
Pushing Different Types on
a Single Stack of Objects

A stack of Objects in Java: Here is a simple program that uses the Java library Stack class. The stack that is provided lets one push a reference (pointer) to any Java Object, which allows one to stack anything.

The program first creates two instances of this stack: s and t. Then the program pushes four different types onto each of the two stacks: int, double, String, and Rational. The first two are primitive types, so they need to be wrapped, while the last two types are already references, and they can be directly assigned to a variable of type Object. The last type is a user-defined type.

There are two stacks now. The first stack is printed and popped, using s.pop() + " ". Except for type String, this implicitly invokes a toString method in each class, including one explicitly coded for the Rational type.

The second stack is popped and printed after first explicitly identifying the type of each stack element, using the instanceof operator.

Don’t use instanceof: This operator is messy, and its use should be avoided. The proper attitude is to realize that there should be some reason to have a stack of different types of items. For example, if one wants to print the items, each should have a toString() method, and the printing should work for any item, so that one could add new types to what can be stacked without changing the stack code, but by only supplying a toString() method for the new types. The situation is the same for other desired activities besides printing: in each case all types involved should have a method (overloaded for that type) that will let you do what you want to do. Then a new added type will still work if it has defined the overloaded method.

Use an Interface instead: So a stack of Objects should have one or more common methods that can be called for any Object on the stack. The proper way to do this in Java is then to create an Interface containing these methods. The Objects on the stack will each be declared with type this Interface. Each such object must implement the Interface, and in that case, the various methods can be called directly for any Object on the stack, without using instanceof, in the same way that the Objects above all implemented toString, and so could be printed without instanceof.
Stack of Objects in Java

// StackMain: Use Java’s library Stack class (legacy)
// push several different types
import java.util.*;
public class StackMain {

    public static void main(String[] args) {
        // create a stacks of Objects
        Stack s = new Stack();
        Stack t = new Stack();
        // push objects onto stacks. Primitives need wrappers
        Object obj = new Integer(47);
        s.push(obj); t.push(obj);
        obj = new Double(83.0);
        s.push(obj); t.push(obj);
        obj = new String("Brazil");
        s.push(obj); t.push(obj);
        obj = new Rational(355, 113);
        s.push(obj); t.push(obj);

        // pop and print the first stack, using empty()
        // method to terminate loop
        while (!s.empty())
            System.out.println(s.pop() + " ");
        System.out.println("\nEnd of the first stack\n\n");

        // pop and print the second stack, using empty(), and instanceof
        while (!t.empty()) {
            if (t.peek() instanceof Integer) {
                int i = ((Integer)t.peek()).intValue();
                System.out.println("Integer: " + i);
            }
            else if (t.peek() instanceof Double) {
                double i = ((Double)t.peek()).doubleValue();
                System.out.println("Double: " + i);
            }
            else if (t.peek() instanceof String) {
                System.out.println("String: " + t.peek());
            }
            else if (t.peek() instanceof Rational) {
                System.out.println("Fraction: " + t.peek());
            }
            t.pop();
        }
        System.out.println("\nTh-th-th-th-that’s-all-folks\n");
    }
}

Run of program

Sample run.
% java StackMain
355/113
Brazil
83.0
47
End of the first stack
Fraction: 355/113
String: Brazil
Double: 83.0
Integer: 47
Th-th-th-th-that’s-all-folks

Revision date: 2004-10-22. (Please use ISO 8601, the International Standard.)