Java has two ways to determine what class an object is: operator instanceof and function getClass().

**1. instanceof operator**

The instanceof operation has the syntax:

<object> instanceof <class-name>

Its evaluation yields true if the object has a partition named <class-name> and false otherwise. Identifier instanceof is a keyword of Java and cannot be use as a variable name.

Here are examples, using variable s and the object to which it points shown at the bottom of this page.

s instanceof Object is true s instanceof JFrame is false  
 s instanceof C is true s instanceof Time is false  
 s instanceof S is true

**Checking the specific class with getClass()**

Sometimes, you need to check whether an object was created using new C(…) —whether the name on its bottom partition is C. We explain how to do this.

Java has a class java.lang.Class each instance of which contains information about a class. You can’t create an instance of class Class; Java does that automatically. For example,

For any class C, static field C.class is an object of class Class that describes C. For example, object JFrame.class describes class JFrame.

You can use the object to find out things about the class. For example, the call JFrame.class.getFields() gives you an array that describes all the fields declared in class JFrame![[1]](#footnote-1) For our purposes here, we need only function getClass:

For an object ob, function ob.getClass() returns an object of class Class that describes the class of ob —the class whose name is on its bottom partition.

We provide some examples using variable s that appears at the bottom of this page.

s.getClass() == Object.class is false  
 s.getClass() == C.class is false  
 s.getClass() == JFrame.class is false  
 s.getClass() == S.class is true  
 s.getClass() == t.getClass() is true

s

S@6dfe

S

equals()

toString()

C(int) getB()

setF(int) S(int)

Object

C

b

1

f

2

S@6dfe

S

t

S@4

S

equals()

toString()

C(int) getB()

setF(int) S(int)

Object

C

b

1

f

2

S@4

S

1. Class Class is part of Java’s *reflection* mechanism, which allows you, within a program, to get all sorts of information about the classes in the program. A full explanation is beyond the scope of this JavaHyperText. Look at JavaHyperText entry “Reflection” for a brief intro and a program that illustrates the use of *reflection*. [↑](#footnote-ref-1)