[instanceof Vs getClass( )](http://stackoverflow.com/questions/4989818/instanceof-vs-getclass)

**instanceof**

whether the object reference on the left-hand side is an instance of the type on the right-hand side or some subtype

**instanceof** operator returns true if the object being evaluated belongs to the given type — in other words, **if our object referred to on the left side of the operator passes the IS-A test for the class or interface type on the right side**. **IS-A relationship**. *isInstance()* method, **we're checking if an object is of a particular type**, and by type, we are either talking about a class or an interface.

**inInstance()** to **verify whether an object can be cast to another class before casting it**

**isInstance()** method to **check the class of an object at runtime.**

**double** amount = 100.0; **if**(amount **instanceof** Double) { // Compilation error, no autoboxing borrower.requestLoan(amount); }

**getClass()**

The **getClass()** method simply **returns the runtime class of the object we are evaluating**, **hence, we don't consider inheritance. Same class and same runtime type**

tests whether the **types are identical**. **They should belong to the same class.**

The reason that I favor the **instanceof approach is that when you use the getClass approach, you have the restriction that objects are only equal to other objects of the same class**, **the same run time type**. If you extend a class and add a couple of innocuous methods to it, then check to see whether some object of the subclass is equal to an object of the super class, even if the objects are equal in all important aspects, you will get the surprising answer that they aren't equal. **In fact, this violates a strict interpretation of the Liskov substitution principle**.

**boolean** flag = (animal1 **instanceof** Animal); **// True**

flag = (lionAnimal **instanceof** Animal); **// True**

flag = (animal1.getClass() == animal2.getClass()); **// True**

flag = (animal1.getClass() == lionAnimal.getClass()); **// False**

flag = (tigerAnimal.getClass() == lionAnimal.getClass()); **// False**

flag = (tiger **instanceof** Animal); // True

//flag = (tiger.getClass() == lion.getClass());**// Compilation Issue**

flag = (tiger.getClass() == animal1.getClass()); **// False**

flag = (lion1.getClass() == lion2.getClass()); **// True**

flag = (animal3 **instanceof** Animal); // True

flag = (animal3.getClass() == animal4.getClass()); **// False**

flag = (lion1 **instanceof** Animal); // True

System.***out***.println("------------>: "+ flag);

Animal animal1 = **new** Animal();

Animal animal2 = **new** Animal();

Animal lionAnimal = **new** Lion();

Animal tigerAnimal = **new** Tiger();

Lion lion1 = **new** Lion();

Tiger tiger = **new** Tiger();

Lion lion2 = **new** Lion();

Animal animal3 = **new** Tiger();

Animal animal4 = **new** Lion();