

Java Instanceof Operator



Last Updated: January 30,
2022



By: Lokesh
Gupta



Java Object Oriented
Programming



Java
Operators

Java instanceof operator (also called type comparison [operator](#)) is used to test whether the object is an instance of the specified type (class or subclass or interface).

It returns –

- **true** – if variable is instance of specified class, it's parent class or implement specified interface or it's parent interface
- **false** – if variable is not instance of the class or implement the interface; or variable is null

1. Java instanceof syntax

instanceof operator tests variable to specified type. Variable is written on left hand side of operator, and type is given on right side of operator.

instanceof Syntax

```
//&lt;object-reference&gt; instanceof TypeName
```

```
boolean value = var instanceof ClassType;
```

```
//or
```

```
if(var instanceof ClassType) {  
    //perform some action  
}
```

2. Java instanceof example

Let's see an example to fully understand the usage of instanceof operator to compare types. In this example, we are using [ArrayList](#) class to test its type information.

instanceof example

```
import java.util.AbstractList;
import java.util.ArrayList;
import java.util.Collection;
import java.util.LinkedList;
import java.util.List;

public class Main
{
    public static void main(String[] args)
    {
        ArrayList<String> arrayList = new ArrayList<>();

        System.out.println(arrayList instanceof ArrayList);    //true

        System.out.println(arrayList instanceof AbstractList); //true

        System.out.println(arrayList instanceof List);         //true

        System.out.println(arrayList instanceof Collection);   //true

        System.out.println(null instanceof ArrayList);         //false

        //System.out.println(arrayList instanceof LinkedList); //Does not compile
    }
}
```

Program output.

Console

```
true
true
true
true
false
```

3. Java instanceof with arrays

In Java, arrays are also considered objects and have fields and methods associated with them. So we can use instanceof operator with [arrays](#) as well.

- **Primitive arrays** are instance of Object and self type. e.g. `int[]` is type of Object and `int[]`. Both comparison returns true.
- **Object arrays** are types of Object, Object array, classtype array, parent class type array. e.g. `Integer[]` is type of Object, `Object[]`, `Integer[]` and `Number[]` (`Integer` extends `Number`).

instanceof example with arrays

```
import java.util.AbstractList;
import java.util.ArrayList;
import java.util.Collection;
import java.util.LinkedList;
import java.util.List;

public class Main
{
    public static void main(String[] args)
    {
        int[] intArr = new int[3];
        float[] floatArr = new float[3];

        Integer[] intObjArr = new Integer[3];
        Float[] floatObjArr = new Float[3];
        String[] stringArr = new String[3];

        System.out.println(intArr instanceof Object);    //true
        System.out.println(intArr instanceof int[]);      //true

        System.out.println(floatArr instanceof Object);  //true
        System.out.println(floatArr instanceof float[]); //true

        System.out.println(intObjArr instanceof Object); //true
        System.out.println(intObjArr instanceof Object[]); //true
        System.out.println(intObjArr instanceof Integer[]); //true
        System.out.println(intObjArr instanceof Number[]); //true

        System.out.println(floatObjArr instanceof Float[]); //true
        System.out.println(stringArr instanceof String[]); //true
    }
}
```

Program output.

Console

```
true
true
true
true
true
true
true
true
true
true
```

4. Using instanceof to correctly typecast

A real life example to use instanceof operator can be typecasting a variable to another type. instanceof operator helps in avoiding **ClassCastException** in runtime.

Consider following example where we are trying to typecast a list to LinkedList class, where original variable is of type ArrayList. It will throw ClassCastException.

Incorrect casting

```
List<String> list = new ArrayList<>();

LinkedList<String> linkedList = (LinkedList<String>)
```

To correctly casting the variable, we can use instanceof operator. It will not result in ClassCastException.

Correct casting

```
List<String> list = new ArrayList<>();

if(list instanceof LinkedList)
```

```
{  
LinkedList<String> linkedList = (LinkedList<String>)  
  
//application code  
}
```

Drop me your questions related to **Java instanceof operator** used for type comparison.

Happy Learning !!

Was this post helpful?

Let us know if you liked the post. That's the only way we can improve.

Yes

No

Recommended Reading:

1. [Java diamond operator – <> operator in Java](#)
2. [Equals Operator \(== \) vs Strict Equals Operator \(=== \)](#)
3. [Java – Pattern Matching for instanceof](#)
4. [Instanceof operators don't need explicit null checks](#)
5. [Compound assignment operator \[i += j\] is not same as \[i = i + j\] in java](#)
6. [JavaScript Spread Operator](#)
7. [Encapsulation vs Abstraction in Java](#)
8. [Overloading vs Overriding in Java](#)
9. [Java Access Modifiers](#)

O. Constructors in Java

Join 7000+ Awesome Developers

Get the latest updates from industry, awesome resources, blog updates and much more.

Email Address

Subscribe

** We do not spam !!*

1 thought on “Java Instanceof Operator”

mr developer

February 3, 2020 at 6:24 pm

I need to cast an Object to ArrayList.
I did your 4th solution but im still getting a warning.
why?

[Reply](#)

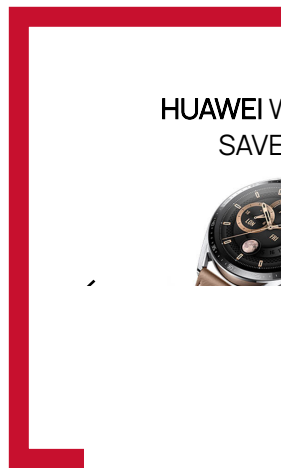
Leave a Comment

☐ Add me to your newsletter and keep me updated whenever you publish new blog posts

Post Comment







HowToDoInJava

A blog about Java and related technologies, the best practices, algorithms, and interview questions.

Meta Links

- [About Me](#)
- [Contact Us](#)
- [Privacy policy](#)
- [Advertise](#)

 Guest Posts
Blogs

REST API Tutorial



Copyright © 2022 · Hosted on [Cloudways](#) · [Sitemap](#)