Interesting facts about null in Java

Almost all the programming languages are bonded with null. There is hardly a programmer, who is not troubled by null.

In Java, null is associated java.lang.NullPointerException. As it is a class in java.lang package, it is called when we try to perform some operations with or without null and sometimes we don't even know where it has happened.

Below are some important points about null in java which every Java programmer should know:

1. null is Case sensitive: null is literal in Java and because keywords are **case-sensitive** in java, we can't write NULL or 0 as in C language.

```
public class Test
{
    public static void main (String[] args) throws java.lang.Exception
    {
        // compile-time error : can't find symbol 'NULL'
        Object obj = NULL;

        //runs successfully
        Object obj1 = null;
    }
}
```

Run on IDE

Output:

```
5: error: cannot find symbol
 can't find symbol 'NULL'
                ^
   variable NULL
 class Test
1 error
```

2. Reference Variable value: Any reference variable in Java has default value null.

```
public class Test
```

```
{
    private static Object obj;
    public static void main(String args[])
    {
        // it will print null;
        System.out.println("Value of object obj is : " + obj);
    }
}
```

Run on IDE

Output:

```
Value of object obj is : null
```

3. Type of null: Unlike common misconception, null is not Object or neither a type. It's just a special value, which can be assigned to any reference type and you can type cast null to any type
Examples:

```
    // null can be assigned to String
    String str = null;

    // you can assign null to Integer also
    Integer itr = null;

    // null can also be assigned to Double
    Double dbl = null;

    // null can be type cast to String
    String myStr = (String) null;

    // it can also be type casted to Integer
    Integer myItr = (Integer) null;

    // yes it's possible, no error
    Double myDbl = (Double) null;
```

4. Autoboxing and unboxing : During auto-boxing and unboxing operations, compiler simply throws Nullpointer exception error if a null value is assigned to primitive boxed data type.

```
public class Test
{
    public static void main (String[] args) throws java.lang.Exception
    {
            //An integer can be null, so this is fine
            Integer i = null;

            //Unboxing null to integer throws NullpointerException
            int a = i;
    }
}
```

Run on IDE

Output:

```
Exception in thread "main" java.lang.NullPointerException
   at Test.main(Test.java:6)
```

5. instanceof operator: The java instanceof operator is used to test whether the object is an instance of the specified type (class or subclass or interface). At run time, the result of the instanceof operator is true if the value of the Expression is not null.
This is an important property of instanceof operation which makes it useful for type casting checks.

```
public class Test
{
    public static void main (String[] args) throws java.lang.Exception
    {
        Integer i = null;
        Integer j = 10;

        //prints false
        System.out.println(i instanceof Integer);

        //Compiles successfully
        System.out.println(j instanceof Integer);
    }
}
```

Run on IDE

Output:

```
false
true
```

6. Static vs Non static Methods: We cannot call a non-static method on a reference variable with null value, it will throw NullPointerException, but we can call static method with reference variables with null values. Since static methods are bonded using static binding, they won't throw Null pointer Exception.

```
public class Test
{
    public static void main(String args[])
    {
        Test obj= null;
        obj.staticMethod();
        obj.nonStaticMethod();
    }

    private static void staticMethod()
    {
        //Can be called by null reference
        System.out.println("static method, can be called by null reference");

    }

    private void nonStaticMethod()
    {
        //Can not be called by null reference
        System.out.print(" Non-static method- ");
        System.out.println("cannot be called by null reference");

    }

}
```

Run on IDE

Output:

```
static method, can be called by null referenceException in thread "main"
java.lang.NullPointerException
    at Test.main(Test.java:5)
```

```
at Test.main(Test.java:5)
```

7. == and != The comparision and not equal to operators are allowed with null in Java. This can made useful in checking of null with objects in java.

```
public class Test
{
    public static void main(String args[])
    {

    //return true;
    System.out.println(null==null);

    //return false;
    System.out.println(null!=null);

    }
}
```

Run on IDE

Output:

```
true
false
```

This article is contributed by **Gaurav Miglani**. If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Java

Login to Improve this Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

Recommended Posts:

Using underscore in Numeric Literals in Java
Classes and Objects in Java
Does Java support goto?
How are Java objects stored in memory?
Establishing JDBC Connection in Java
Java Math copySign() method with Examples
Java | Converting an Image into Grayscale using cvtColor()
Java | Removing whitespaces using Regex

Java System.nanoTime() vs System.currentTimeMillis
Java toDegrees() method with Example

(Login to Rate)

2.4 Average Difficulty : **2.4/5.0**
Based on **35** vote(s)

Basic Easy Medium Hard Expert

Add to TODO List
Mark as DONE

Writing code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments Share this post!