JAVA.LANG.CHARACTER.ISMIRRORED METHOD

Description

The **java.lang.Character.isMirrored*intcodePoint*** determines whether the specified character *Unicodecodepoint* is mirrored according to the Unicode specification. Mirrored characters should have their glyphs horizontally mirrored when displayed in text that is right-to-left.

For example, '\u0028' LEFT PARENTHESIS is semantically defined to be an opening parenthesis. This will appear as a " " *intextthatisleft − to − rightbutasa* " " in text that is right-to-left.

Declaration

Following is the declaration for **java.lang.Character.isMirrored** method

```
public static boolean isMirrored(int codePoint)
```

Parameters

- **codePoint** - the character *Unicodecodepoint* to be tested

Return Value

This method returns true if the character is mirrored, false if the character is not mirrored or is not defined.

Exception

- **NA**

Example

The following example shows the usage of lang.Character.isMirrored method.

```
package com.tutorialspoint;

import java.lang.*;

public class CharacterDemo {

   public static void main(String[] args) {

      // create 2 int primitives cp1, cp2
      int cp1, cp2;

      // assign values to cp1, cp2
      cp1 = 0x0c01;
      cp2 = 0x003c; // represents <

      // create 2 boolean primitives b1, b2
      boolean b1, b2;

      /**
       *  check if cp1, cp2 represent mirrored characters
       *  and assign results to b1, b2
       */
      b1 = Character.isMirrored(cp1);
      b2 = Character.isMirrored(cp2);

      String str1 = "cp1 represents a mirrored character is " + b1;
      String str2 = "cp2 represents a mirrored character is " + b2;

      // print b1, b2 values
```

```
        System.out.println( str1 );
        System.out.println( str2 );
    }
}
```

Let us compile and run the above program, this will produce the following result:

```
cp1 represents a mirrored character is false
cp2 represents a mirrored character is true
```