JAVA.LANG.SECURITYMANAGER.CHECKWRITE METHOD

Description

The **java.lang.SecurityManager.checkWrite*FileDescriptorfd*** method throws a SecurityException if the calling thread is not allowed to write to the specified file descriptor. This method calls checkPermission with the RuntimePermission " *writeFileDescriptor* " permission.

If you override this method, then you should make a call to super.checkWrite at the point the overridden method would normally throw an exception.

Declaration

Following is the declaration for **java.lang.SecurityManager.checkWrite** method

```
public void checkWrite(FileDescriptor fd)
```

Parameters

- **fd** -- the system-dependent file descriptor.

Return Value

This method does not return a value.

Exception

- **SecurityException** -- if the calling thread does not have permission to access the specified file descriptor.
- **NullPointerException** -- if the file descriptor argument is null.

Example

Our examples require that the permissions for each command is blocked. A new policy file was set that allows only the creating and setting of our Security Manager. The file is in C:/java.policy and contains the following text:

```
grant {
  permission java.lang.RuntimePermission "setSecurityManager";
  permission java.lang.RuntimePermission "createSecurityManager";
  permission java.lang.RuntimePermission "usePolicy";
};
```

The following example shows the usage of lang.SecurityManager.checkWrite method.

```
package com.tutorialspoint;

import java.io.FileDescriptor;


public class SecurityManagerDemo extends SecurityManager {

   public static void main(String[] args) {

   // set the policy file as the system securuty policy
   System.setProperty("java.security.policy", "file:/C:/java.policy");

   // create a security manager
   SecurityManagerDemo sm = new SecurityManagerDemo();

   // set the system security manager
```

```
    System.setSecurityManager(sm);

    // perform the check
    FileDescriptor fd=new FileDescriptor();
    sm.checkWrite(fd);

    // print a message if we passed the check
    System.out.println("Allowed!");
    }
}
```

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

```
Exception in thread "main" java.security.AccessControlException: access denied
(java.lang.RuntimePermission writeFileDescriptor)
```