SECURITYMANAGER CHECKMEMBERACCESS METHOD

Description

The **java.lang.SecurityManager.checkMemberAccess***Class < ? > clazz, intwhich* method throws a SecurityException if the calling thread is not allowed to access members. The default policy is to allow access to PUBLIC members, as well as access to classes that have the same class loader as the caller. In all other cases, this method calls checkPermission with the RuntimePermission " *accessDeclaredMembers* " permission.

If this method is overridden, then a call to super.checkMemberAccess cannot be made, as the default implementation of checkMemberAccess relies on the code being checked being at a stack depth of 4.

Declaration

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

```
public void checkMemberAccess(Class<?> clazz, int which)
```

Parameters

- **clazz** -- the class that reflection is to be performed on.
- **which** -- type of access, PUBLIC or DECLARED.

Return Value

This method does not return a value.

Exception

- **SecurityException** -- if the caller does not have permission to access members.
- **NullPointerException** -- if the clazz 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.checkMemberAccess method.

```
package com.tutorialspoint;

import java.lang.reflect.Member;

public class SecurityManagerDemo extends SecurityManager {

   // checkMemberAccess needs to be overriden.
   @Override
   public void checkMemberAccess(Class<?> clazz, int which) {
   throw new SecurityException();
   }
```

```
    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
    sm.checkMemberAccess(SecurityManagerDemo.class, Member.PUBLIC);

    // 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.lang.SecurityException
```

Loading [MathJax]/jax/output/HTML-CSS/jax.js