JAVA.LANG.SECURITYMANAGER.CHECKEXIT METHOD

Description

The **java.lang.SecurityManager.checkExit*intstatus*** method throws a SecurityException if the calling thread is not allowed to cause the Java Virtual Machine to halt with the specified status code. This method is invoked for the current security manager by the exit method of class Runtime. A status of 0 indicates success; other values indicate various errors.

This method calls checkPermission with the RuntimePermission " *exitVM.* " + *status* permission. If you override this method, then you should make a call to super.checkExit at the point the overridden method would normally throw an exception.

Declaration

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

```
public void checkExit(int status)
```

Parameters

- **status** -- the exit status.

Return Value

This method does not return a value.

Exception

- **SecurityException** -- if the calling thread does not have permission to halt the Java Virtual Machine with the specified status.

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.checkExit method.

```
package com.tutorialspoint;

public class SecurityManagerDemo extends SecurityManager {

   // checkExit needs to be overriden.
   @Override
   public void checkExit(int status) {
   throw new SecurityException("Not allowed.");
   }

   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.checkExit(5);

    // 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: Not allowed.
```