JAVA.LANG.SECURITYMANAGER.CHECKACCEPT METHOD

Description

The **java.lang.SecurityManager.checkAccept***Stringhost, intport* method throws a SecurityException if the calling thread is not permitted to accept a socket connection from the specified host and port number. This method is invoked for the current security manager by the accept method of class ServerSocket. This method calls checkPermission with the SocketPermission*host* + ":" + *port*, " *accept* " permission. If you override this method, then you should make a call to super.checkAccept at the point the overridden method would normally throw an exception.

Declaration

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

```
public void checkAccept(String host, int port)
```

Parameters

- **host** -- the host name of the socket connection.
- **port** -- the port number of the socket connection.

Return Value

This method does not return a value.

Exception

- **SecurityException** -- if the calling thread does not have permission to accept the connection.
- **NullPointerException** -- if the host 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.checkAccept method.

```
package com.tutorialspoint;

public class SecurityManagerDemo {

   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
   SecurityManager sm = new SecurityManager();

   // set the system security manager
```

```
   System.setSecurityManager(sm);

   // check if accepting socket connection is enabled
   sm.checkAccept("www.tutorialspoint.com", 8080);

   // 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.net.SocketPermission www.tutorialspoint.com:8080 accept,resolve)
```