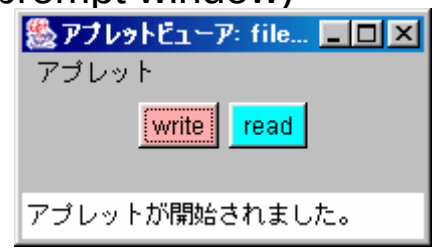
- Sample Program
  - Download the following three files from the “Mechanical Design Technology” web page: “sample2.java,” “sample2.html,” and “av.bat” (an MS-DOS batch file)
    - A filename with an extension such as ".bat" means the file may do something automatically; therefore, a file with such a name cannot be downloaded in some systems for security reasons; therefore, you must download “av.txt,” check its content, and then rename it “av.bat"
  - Compile “sample2.java"
    - Note: You may need to specify a path, such as “c:¥j2sdk1.4.2_04¥bin," to the directory where Java commands (e.g., "javac") are placed
  - Define your security policy (as described later)
  - Run Java applet by opening the file “sample2.html" using 'appletviewer' (as described later)
  - When you click the "write" button, data are printed in a text file "data.txt" and some messages are printed in the standard output stream (command prompt window); when you click the "read" button, data are read from the file "data.txt" and the data and message are printed in the standard output stream (command prompt window)
  - To end the program, select the menu [applet] -> [exit]



- Security in Java
  - Java program runs on JVM (Java Virtual Machine) -> JVM always monitors the behavior of the program -> Dangerous behavior can be checked and prohibited -> That is one of the merits of using Java from a security viewpoint
  - Java Applet and Java Application
    - Applet
      - Downloaded from other computers and executed on JVM on your computer -> Reading and writing files can be dangerous and are not permitted by JVM in the default security setting
    - Application
      - Runs on JVM on your computer -> There is less possibility of security problems -> There are fewer restrictions on reading and writing files than for Java Applet.
    - Applet and Application can easily be rewritten for each other

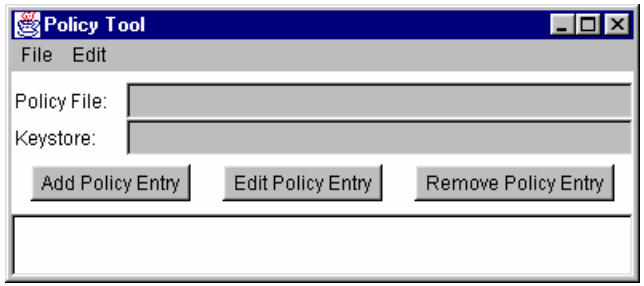
	Java Applet	Java Application
Runs on	Computers connected by the Internet	A computer locally
Default Security Setting	More restrictions	Fewer restrictions
Difference in Program	A browser provides a window	- Needs ‘main’ method - Needs to create a window ('Frame' class), as in "edit_frame.java"

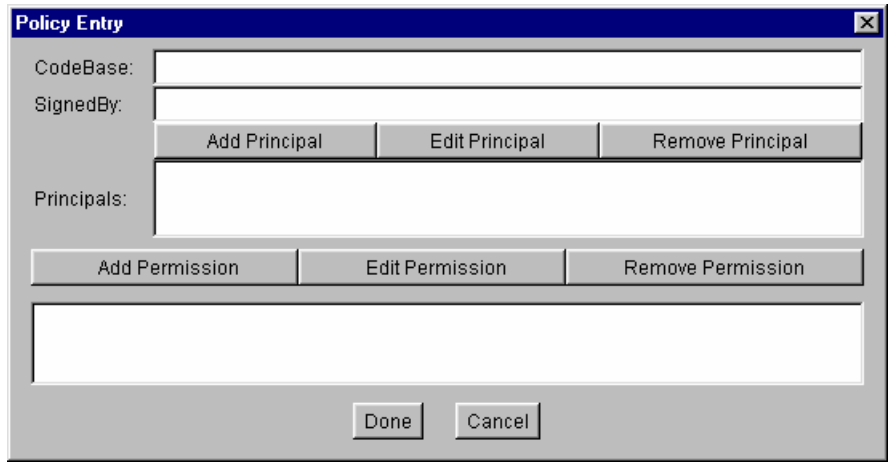
- When you open “sample2.html" using browsers such as Internet Explorer and Netscape Navigator and run a Java Applet “sample2.class," its reading/writing function does not work
- Set your security policy using Java2 SDK 'policytool' (more specifically, give your Java Applet permission to read and write specific files); Then, execute the Java Applet using Java2 SDK 'appletviewer'
  - Details on tools (e.g., 'policytool,' 'appletviewer') are found in [Tool Documentation (docs)] of J2SE 1.4.2 Documentation

Setting Java Security Policy

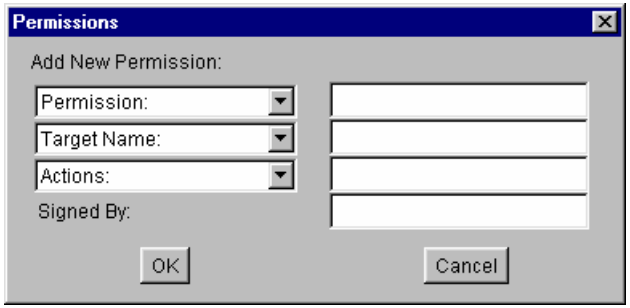
- Using the following operations, you can create and modify the policy configuration file that defines your installation's Java security policy using the Java policy tool
  - The following instruction is for the WindowsXP environment and assumes that your Java Applet file “sample2.class" and a text data file "data.txt" are located in the directory "c:¥myjava”; After the following operations, a security policy file ".java.policy" is created in "c:¥myjava“
- Open a command prompt window from the Windows [start menu] -> [programs] -> [accessories] -> [command prompt]
- Move to the target directory (e.g., by typing "cd c:¥myjava").
- Type "policytool”; The [Policy Tool] window appears
  - An error message, "A policy file is not found," may come up; Just skip it by clicking the [OK] button

- To create a new policy file, select the [Add Policy Entry] button in the [Policy Tool] window; The [Policy Entry] window appears
  - To edit an existing policy file, select the menu [file] -> [open] and open the file





- To give permission to all of your programs in the c:¥myjava directory, type the following file location information into the [CodeBase] text box
  - file:/c:/myjava/
  - Note: Giving permission to programs too widely may cause problems from the viewpoint of security; Give permission only to programs in your own directory for this course
- To give your programs new permission to access some files, select the [Add Permission] button in the [Policy Entry] window; This brings up the [Permissions] window



- Select a permission type "FilePermission" from the [Permission:] drop-down list
- To make the file "c:¥myjava¥data.txt" accessible from your programs, type the following target file name into the text box to the right of the [Target Name:] drop-down list
  - c:/myjava/data.txt
  - Note: Making files accessible too widely may cause problems from the viewpoint of security; Make only the files in your own directory for this course readable/writable
- To specify both read and write access to the target file, first select "read" (or "write," the order doesn't matter) from the [Actions:] drop-down list; The word "read" appears in the text box; Then select "write," and the word "write" will be appended, preceded by a comma and a space
- When you are done specifying the permission information, select the [OK] button; This brings you back to the [Policy Entry] window
- Since you are done adding a policy entry, select the [Done] button in the [Policy Entry] window and return to the [Policy Tool] window

- Select menu [File] -> [Save] and save the specified information in the file in your directory "c:¥myjava¥.java.policy"
- The message ".java.policy" file is saved successfully" appears; Click the [OK] button
- Select menu [File] -> [Exit] to exit 'policytool'
- After the above operations, any Java program ("*.class") in the directory "c:¥myjava" can read and write the file "c:¥myjava¥data.txt"

Java Applet Execution using 'appletviewer'

- Java2 SDK 'appletviewer' executes Java Applets marked by <OBJECT>, <EMBED>, <APPLET> tags in an HTML file; Your security policy file created above must be specified as an option; A sample file “sample2.html" can be executed by either (a) or (b)
  - (a) Type directly as follows in a command prompt window
    - "appletviewer -J-Djava.security.policy=c:¥myjava¥.java.policy sample2.html“
  - (b) Use a sample MS-DOS batch file "av.bat" containing the following text line
    - "appletviewer -J-Djava.security.policy=c:¥myjava¥.java.policy %1“
      - ‘%1’ matches the first argument that follows the batch file name; When you type “av filename,” %1 represents the filename
    - Type as follows in a command prompt window
    - "av sample2.html"

- File Input and Output by Java
  - Java Class and the Method for File Output
    - 'FileWriter': A class for writing character files
    - 'PrintWriter': A class for printing formatted representations of objects to a text-output stream
      - A class on the 'println' method of 'PrintWriter': A method for printing data and terminating the line; If you just print data and do not terminate the line, use the 'print' method instead
    - In the sample program, an output stream object with buffering and formatting is constructed using these three classes; See the sample program
      - Output from the program **->** Formatted by the 'println' method of 'PrintWriter' **->** Written to the file by 'FileWriter'

  - Java Class and the Method for File Input
    - 'FileReader': A class for reading character files
    - 'BufferedReader': A class for reading text from a character-input stream, buffering characters so as to provide for the efficient reading of characters, arrays, and lines
      - A class on the 'readLine' method of 'BufferedReader': A method for reading a line of text; A line is considered to be terminated by any one of a line feed ('¥n'), a carriage return ('¥r'), or a carriage return followed immediately by a linefeed
    - In the sample program, an input stream object with buffering is constructed using these two classes; See the sample program
      - Input to the program **<-** Buffered by 'BufferedReader' **<-** Read from the file by 'FileReader'

- Standard Input and Output Streams
  - 'System.in' ('in' field of 'System' class): A standard input stream; This stream is already open and ready to supply input data; Typically this stream corresponds to keyboard input
  - 'System.out' ('out' field of 'System' class): A standard output stream
    - This stream is already open and ready to accept output data; Typically this stream corresponds to display output such as a command prompt window or the Java console of a browser
    - A typical way to write a line of output data is to use a 'println' method, as in System.out.println(data)
  - 'System.err' ('err' field of 'System' class): A standard error output stream
    - This stream is already open and ready to accept output data; Typically this stream corresponds to display output such as a command prompt window or the Java console of a browser
    - By convention, this output stream is used to display error messages or other information that should come to the immediate attention of a user even if the principal output stream, the value of the variable output, has been redirected to a file or other destination that is typically not continuously monitored

- Details on classes (e.g., 'BufferedReader,' 'FileReader,' 'FileWriter,' 'PrintWriter,' and 'System') and their methods are found in [Java 2 platform API Specification (docs)] of J2SE 1.4.2 Documentation.