Kristianstad University
SE-291 88 Kristianstad
Sweden
+46 44 20 30 00
www.hkr.se

School of Health and Society
Nazila Hasanzade
2015-10-02

Seminar 4 Operating systems

Read the complete document before you start with the specific tasks.

Rules and Requirements

- After finishing the tasks prepare a zip-file that contains all the solved tasks.
- Upload the zip-file to it's learning before the deadline.
- Prepare before the seminar.
- Attend the seminar session that corresponds to your group and take an active role during the seminar discussions and presentations.
- This seminar is mandatory and one must be graded Pass to be able to get a grade for the full course.
- All the tasks must be solved and they should be solved individually.
- Grades on seminars are either pass or fail (G/U)
- If you fail the seminar you must wait until the end of the course to attend to re-seminar.

Introduction

The purpose of this Seminar is to study file management in Java. You will study the classes File, FileInputStream, and FileOutputStream. After this seminar you should be able to understand streams in Java and how you can use different streams in your own project. Even though we will focus on files the same methods is used also for other kind of streams such as network streams over a socket.

The seminar will also try to clarify the distinction between sequential access and direct access.

Observe that we will use the older java.io.File class instead of the newer java.nio.File class. The purpose here is that from my point of view the older class will more clearly show the properties of file management from an operating system point of view.

Preparations

Before starting with the programming, you are expected to do the following tasks. You need to understand the objective of the seminar and do the following preparations in order to complete the seminar without too much effort.

Read the following from the Java API documentation

http://docs.oracle.com/javase/7/docs/api/java/io/File.html

http://docs.oracle.com/javase/7/docs/api/java/io/FileInputStream.html

http://docs.oracle.com/javase/7/docs/api/java/io/FileOutputStream.html

Programming Tasks

Task 1

In this task we will study the fundamentals of streams in Java. In its most fundamental level a stream is a directed stream of bytes from a source to a destination. Hence there is a source and a destination. The source or destination can be a file or a lot of different things such as network connection and user input/output. Of course we can also in code write to and read from a stream.

a) Download the code for seminar4 from its learning. Run the program Seminar4a. This programs creates a fileoutputstream with the filename seminar4.a and writes the bytes 255, 254, ..,0 to it.

b) Open the file seminar4.a created by the program above in a text editor such as notepad. Observations?

c) Uncomment the call to readFile in main. This method will read the file created before and prints the byte values one by one. Each byte is written both in hexadecimal and in decimal. Note that read will return an integer even though it is reading only one byte. The reason is that sine the functions will return an integer we can use the value -1 as an integer to signal that there is no more data in the stream.

Task 2

From Task 2 you will hopefully understand how to read and write bytes sequentially to a file. Even though streams are based on bytes we will often like to write other kind of data to a stream.

a) Run the program Seminar4b. This program tries to write an integer to a file Seminar4.b. When we read the content we see that only the last byte of the file is written.

b) To be able to output numerical values to a file can use a DataOutputStream. (DataOutputStream is located in java.io) To attach a DataOutputStream to FileOutputStream we use the following code

doutStream = new DataOutputStream(reference to FileOutputStream);

Add a DataOutputStream such that we can write the integer to the file without losing

data.

c) When everything works as it should then you will read four bytes that represent the integer. Observe the ordering of the bytes. Since the most significant byte of the integer is written first we call this big endian.

d) To read an integer from a file we can use a DataInputStream. Attach a DataInputStream to the fileinputstream and read an integer and print it. Verify that it is working. Try to open the file in a text editor. Observations?

With DataInputStream and DataOutputStream you can read and write all fundamental data types in java to a file. In this exercise we have only written one integer. You can modify your program if you like to test other data types as well.

If you would like to write and read your own objects to a stream you have the classes ObjectOutputStream and ObjectInputStream. With them you can access streams with your object with readObject and writeObject. Another thing you need to do to make this work is to declare your classes to implement Serializable. I will leave this part for you to test by yourself.

Task 3

If you open the file Seminar4.b created in Task 2 in a textfile you see that the file cannot be interpreted as text. We call a file that consists of raw byte data a binary file. Binary files have many nice properties if we would like to process them by computer programs, however, for a people it is a pain in the neck to read them. Instead we would like to use text files.

a) To write data as text instead of binary data we use the class PrintWriter. Modify your code from task 2 such that instead of using a DataOutputStream we use a PrintWriter instead.

New PrintWriter(fileoutputstream)

Save the integers 1,3 and 5 to a file Seminar4.c using a PrintWriter using the method print. (Notice the similarity with Standard.out.print. This is not an accident. □)

(Observe that the old method for reading a file will generate an error. Hence comment the call such that you will not get that exception)

b) Open the file in a texteditor. What is the content?
With a PrintWriter you can print all kind of data in text. At least as long that the meaningful can be transformed into a string. (toString method)

c) To read the file we use a Scanner object. Observe that the scanner is found in java.util instead of java.io. The reason is that the scanner was added later than the traditional io methods.
In the readFile method add a scanner to the fileinputstream instead of the DataInputStream.
Read one integer from the file and print it. What is the value of this integer? Is this what you have expected given that you have saved three integers in b)?

d) From b) and c) you will hopefully have noticed that we need to add delimiters when we write individual small integers as text and would like to be able to read them back one by one as text.
Modify the code such that you add a space after each written integer. What will be the input after reading the file?

You can also test your program such that instead of read and write only integers you can try other data types and strings. Observe that reading a string is a little bit tricky since we can only read individual characters or entire lines.

Task 4

In the tasks so far we have used sequential access which is the basics for streams. A file however, can also be read by direct access or Random Access.

a) To create a random access file you do this by the following RandomAccessFile

in= new RandomAccessFile(filename,”r”) Here r is the access type for reading. For

writing we also add a w.
Then we can go directly to a specific byte in the file with

in.seek(n), where n is the position we would like to access.

To read from the current position we use in.readByte(). (We can also read other types. The problem here is to remember that in a binary file we need to remember the size of each data type. For instance an integer occupies four byte.)

Modify the program from Task 1 such that you can read individual bytes in random order. Then read and print the bytes at the following positions. 12, 5, 167, 200.

What is the result?

Task 5

So far in the seminar we have only performed reading and writing to files. But if we would like to manage files and perform more complex operations on a file or a directory we can use the class File. Even though the name suggest that it only cover files it will also be used to represent a directory. Here we will use a file object to print some simple file and directory information. Note that this part of the seminar can be combined with the code from seminar 1 and then you have created your own command line interface. Maybe something you can do if you are not satisfied with the regular command interface and have a lot of spare time. □

To create a file object we call

File file= new File(filename)

To open the current directory use File(“.”).

When we have opened a file there are several operations that can be performed on a file. For instance we can write the complete filename including path with

file.getCanonicalPath();

Check the Java documentation for more details.

a) Run the program **Seminar4e** given in the code for seminar 4. As you will see this program only waits for user input and then exits.

b) Add the following modifications. (They are also given as comments in the source file. In principle you do not need to modify main, only the methods called by main.)
i. Before a user selects a filename print content of current directory
ii. After a user entered a filename check that it is a valid filename. (File exist)
iiia. If it is a file print some information as specified in the comments.
iiib) If it is a directory list the files in the directory.

Final Observations

Before you present your seminar, make sure that you have saved all your code and have it ready for the presentation. It is important that you can explain your solution and provided answer to questions regarding the implementation.