

Data Structure and algorithms using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 46
Random Access File

So, we are discussing about Java IO and files and one important facility is that Java language supports it is basically random access file. In the last video lectures, we have discussed about sequential access of files.

(Refer Slide Time: 00:47)

The slide is titled "CONCEPTS COVERED" and features a list of topics on the left and two diagrams on the right. The list includes:

- Random Access Files in Java
- Reading a Random Access File
- Writing a Random Access File
- Simultaneous Read Write Operation
- Appending

The "Sequential access" diagram shows a horizontal bar divided into eight segments labeled 1 through 8. Arched arrows above the bar indicate a linear, left-to-right progression from segment 1 to segment 8.

The "Random access" diagram shows a horizontal bar divided into eight segments labeled 1 through 8. Curved arrows above the bar indicate non-linear jumps between segments, such as from 1 to 3, 3 to 7, 7 to 2, 2 to 8, 8 to 6, 6 to 4, and 4 to 5.

So, random access file is another unique addition in this Java java dot io package. So, in this lecture will cover the basic concept of random access file, and how some data can be read from a random access file and how some data also can be stored into a random access file. Then simultaneous read and write operation that is why the random access file is meant for actually, we can do it and then how the data can be appended. And finally, we will discuss about how an object can be stored into a random access file and then the same object can be read from a random access file. So, this is basically the topic that we want to discuss.

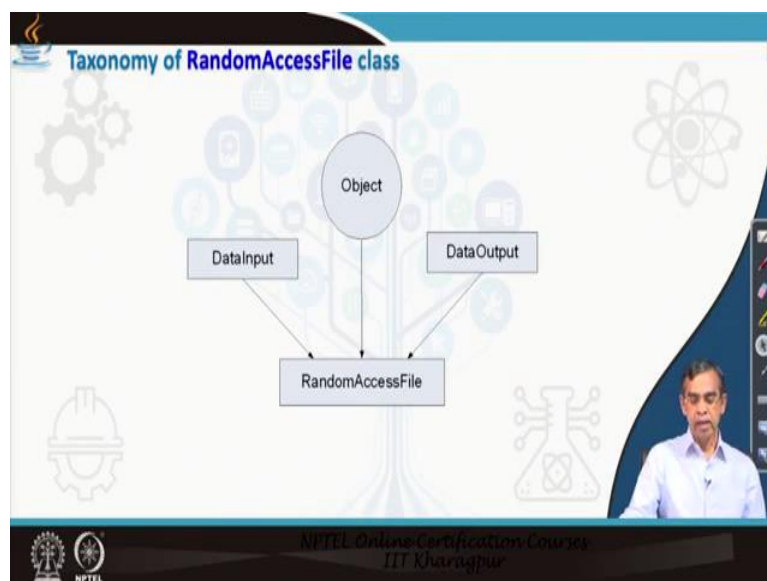
(Refer Slide Time: 01:38)



So, what exactly a random access file is? So, as the name implies, random access file allows you to write anywhere in the file and read from anywhere in the file. So, this is the concept it is there, if this is in contrast to the sequential access file, where it allow you to read sequentially one after another. So, at the moment where you finish your writing next time if you want to write then it is it will be at the next point actually. So, it is sequentially the file goes.

So, reading also starting from the 0 position, it will read one by one either as a byte or character (())(2:20). So, is a sequential access.

(Refer Slide Time: 02:26)



And random access as you said that it can be access either reading or writing in a random access file. Now, in java dot io package, there are two interfaces; data input and data output. Random access file basically implements these two interfaces. So, whatever the methods are there, it is basically available into the random access file there.

(Refer Slide Time: 02:53)

RandomAccessFile class

This class is used for reading and writing to random access file.

- A random access file behaves like a large array of bytes.
- There is a cursor implied to the array called file pointer, by moving the cursor we do the read write operations.
- If end-of-file is reached before the desired number of byte has been read than EOFException is thrown. It is a type of IOException.

NPTEL Online Certification Courses
IIT Kharagpur

So, this basically interface implements are there. Now, so like other files, random access file also different in the concept that other files that we have studied can be opened either in input mode or output mode. So, either it is as a it will work as an input stream or it will work as the output stream which we have already discussed and studied and learned also. So, one file if you open as an input stream you will not be able to write into the same file, then you have to close the input stream then open it again as an output stream and then write it there, so this is the concept.

This is in contrast in case of random access file you will be able to open a file both in read and write mode actually. So, both read and write operation can be carried out simultaneously, and even if the end of file occurs then in case of random access file it will just throw an IO exception that exception is there. It is basically now another concept that I want to convey here is that random access file it is basically the byte stream, it is array of bytes. So, whole the file will be considered as an array of bytes actually.

So, a random access file can be treated in that sense is an array of bytes as an object or is a collection, is the array of bytes like. So, whole the contents can be stored in array byte that is

why it allow you to pinpoint in a particular position. So, byte by byte accessing is possible there.

(Refer Slide Time: 04:36)

The slide displays a table with two columns: 'Constructor' and 'Description'. The first row shows the constructor `RandomAccessFile(File file, String mode)` with a description: 'Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument.' The second row shows the constructor `RandomAccessFile(String name, String mode)` with a description: 'Creates a random access file stream to read from, and optionally to write to, a file with the specified name.' A red circle highlights the parameters of the first constructor.

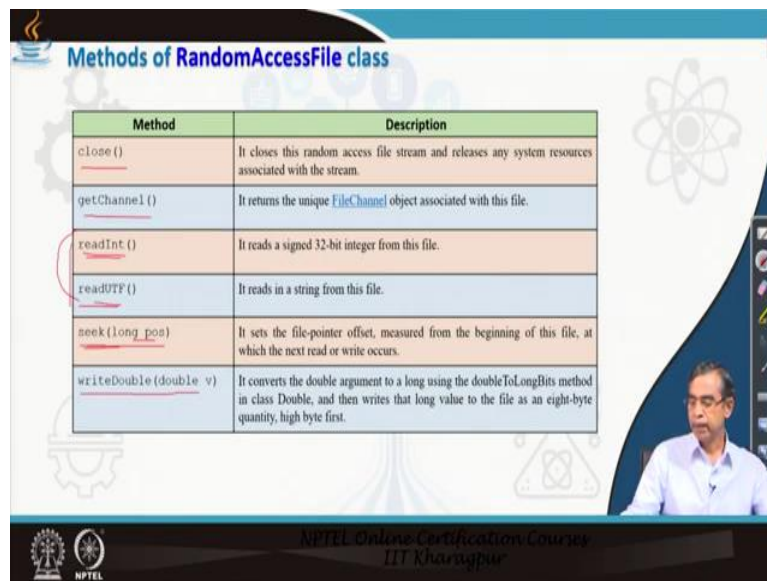
Constructor	Description
<code>RandomAccessFile(File file, String mode)</code>	Creates a random access file stream to read from, and optionally to write to, the file specified by the File argument.
<code>RandomAccessFile(String name, String mode)</code>	Creates a random access file stream to read from, and optionally to write to, a file with the specified name.

Now, so in order to create the object of this kind of class, that is the random access file in java dot io random access file class is different, this class has to construct to create the object of this type. The two constructors are listed here in this table. The first constructor is called the name of the file to be passed because it is always a file object and then second is the mode, the mode can be read or write both so that is why the mode you can specify.

So, this is a mode, mode can be specified as a string and then second constructor basically give the name of the file as a write, here we are giving the file as an object, but here you give the name of the file, automatically file object will be created, if the file does not exist it will create newly and then mode can be specified. So, these are two constructor; one you can pass file as an object as a first argument, second argument is mode always in the two constructor.

So, a file random access file can be created if originally a file is available, initially a file is available then this mode and if is not available or even if it is available, this constructor can be used. So, there are two constructor you can use to create the file object of type random access file. And this is important to note actually, because we will follow these two constructor while we create a random access file.

(Refer Slide Time: 06:16)



Method	Description
<code>close ()</code>	It closes this random access file stream and releases any system resources associated with the stream.
<code>getChannel ()</code>	It returns the unique <code>FileChannel</code> object associated with this file.
<code>readInt ()</code>	It reads a signed 32-bit integer from this file.
<code>readUTF ()</code>	It reads in a string from this file.
<code>seek (long pos)</code>	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
<code>writeDouble (double v)</code>	It converts the double argument to a long using the <code>doubleToLongBits</code> method in class <code>Double</code> , and then writes that long value to the file as an eight-byte quantity, high byte first.

Now, so there are different methods which you can consider to operate with the random access file. The close is a trivial method is always there so, is getChannel so there are different channel that can be used to open a random access file. So, in order to know that which channel it is there, now these are to basically read you can read as an integer, read as an UTF, but it is ultimately written as a byte actually, but you can use this as a readint, as a primitive form, this is basically integer form and this is the text.

You can write in the form of double also, seek long position if you want to skip certain or if you want to position your reading or writing pointer into a particular position in the file, then the seek method can be called giving the position, the position can be 0 to any length of the file that is the position actually. And if you specify the seek beyond this end of file or something, then it will return an IO exception. So, these are the different methods are there, few more methods also in this file access is there which you have listed here.

(Refer Slide Time: 07:29)

Method	Description
<code>writeFloat(float v)</code>	It converts the float argument to an int using the <code>floatToIntBits</code> method in class <code>Float</code> , and then writes that int value to the file as a four-byte quantity, high byte first.
<code>write(int b)</code>	It writes the specified byte to this file.
<code>read()</code>	It reads a byte of data from this file.
<code>length()</code>	It returns the length of this file.
<code>seek(long pos)</code>	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.

And here the write float write double like, it is write also you can write integer as a byte, read this is basically reading as a byte into the file, length is a method by which you can calculate the length of the random access file and this also we have discussed about it. So, these are the different methods available or defined in the class, this class is random access file, which you can consider for your file management actually.

(Refer Slide Time: 08:02)

-
- As the name implies the class `RandomAccessFile` allows us to handle a file randomly in contrast to sequentially in `InputStream` or `OutputStream` classes.
 - It allows to move file pointer randomly.
 - Moreover, it allows read or write or read-write simultaneously.

Now, what is the utility of this random access file? I have mentioned that it is basically useful for both reading and writing just like input stream and output stream classes that we have discussed about, but unlike these input stream and output stream classes, it basically allow you to read or write in a random fashion, and the file pointer can move actually as a random.

File pointer means from which pointer you want to either read or to write. And it basically can be open both read and write mode, we have already mentioned it. Now, let us have certain utility of this program.

(Refer Slide Time: 08:46)



Using the reading from a random access file first we will discuss then we will discuss about our writing prompt random access file and then both reading and writing from random access file. So, this will make a sense about using the random access file then. So first let us consider how we can read some data which is already stored in a random access file.

(Refer Slide Time: 09:14)

A presentation slide titled "Methods used to read RAF" in blue text. It contains a table with two columns: "Method" and "Description". The table lists seven methods: read(), length(), seek(long pos), close(), getChannel(), readInt(), and readUTF(). In the bottom right corner, a small video inset shows the same man from the previous slide. The NPTEL logo and "NPTEL Online Certification Courses IIT Kharagpur" are visible at the bottom.

Method	Description
read ()	It reads a byte of data from this file.
length ()	It returns the length of this file.
seek (long pos)	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
close ()	It closes this random access file stream and releases any system resources associated with the stream.
getChannel ()	It returns the unique FileChannel object associated with this file.
readInt ()	It reads a signed 32-bit integer from this file.
readUTF ()	It reads in a string from this file.

So regarding read, these are the methods that you can consider.

(Refer Slide Time: 09:19)

```
import java.io.IOException;
import java.io.RandomAccessFile;

public class RandomAccessFileExample {
    static final String FILEPATH = "NPTEL.txt";
    public static void main(String[] args) {
        try {
            System.out.println(new String(readFromFile(FILEPATH, 0, 18)));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static byte[] readFromFile(String filePath, int position, int size)
        throws IOException {
        RandomAccessFile file = new RandomAccessFile(filePath, "r");
        file.seek(position);
        byte[] bytes = new byte[size];
        file.read(bytes);
        file.close();
        return bytes;
    }
}
```

Now here is a program, this program let us look at this program little bit carefully what we are doing. Now we have to import this class otherwise all these methods that we are referring can lead to an error. So this import is important, import as the java dot io random access file. This is a demo of random access read file demo. So, let this name of the file, I give the name of the file is like this the NPTEL dot txt. This is the name of the file that we want to read from it.

Then what we are doing is that this is our main method, in this main method, we call this method, which is the user defined method or I'll just define this method. So, read from file the name of the file and it basically read from 0 to 18, this is basically position and basically the size that we want to read it actually. So, this is our main method actually there, now let us have the discussion about defining this read from file whose name is there and read from starting point is 0.

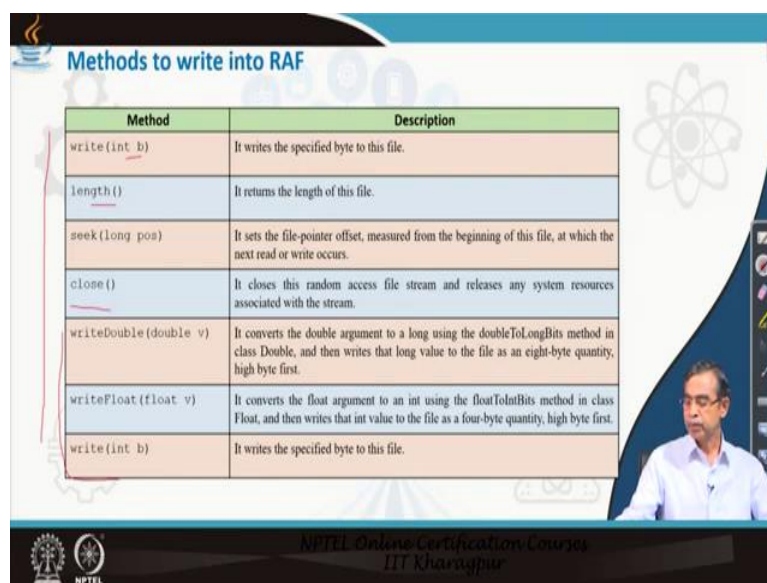
So, now so this method is defined here, this method written as a byte as we have mentioned here, so read from file defining, string is the file path, this is the name of the path, name of the file rather, integer position and this is the size. Now let us see how we define this program here. So, we first create the object of type random access file that is a constructor we used, this is basically the name of the file object and then we create we open it in a read mode only, so this is the mode. So, this is a file object and this is the mode that we pass it so we create an object random access file.

Then file seek position that means we start from 0 position whatever we pass it here 0. So, now 0 position so start from 0, then here we can see we define a temporary bytes, and the size of the byte is 18 because we want to read 18 bytes. So, let us declare an array of bytes here. Now read and these bytes so whole content of the file random access file will read for this file object, here this is the file object we will read and it will store into the bytes. So this is the one read method, which is defined for this random access file, who is basically read whole bytes starting from 0 to 18 like.

And so this basically completes the reading of the file, then this file again you can open, later on we will see how we can open this file and then content can be displayed. So, this basically explain you how a file can be read from a particular position to some position like. Here we read 0 to 18, instead of 0 to 18 we can read from any position to any position whatever it is there, but this limit you have to check it carefully so that it should within the range and should not go the last size and beyond the end of file. If it is go to the end of file, then it is an exception.

So this program, you can check later on, you can run it, you can practice and then you can see how it can read a random access file in a reading mode. Now, let us continue another program who is basically open the file that we have just created and then we can open it and you can write the data into that. So, this is about reading the file and let us see how the writing can be done.

(Refer Slide Time: 13:31)



The slide displays a table with two columns: 'Method' and 'Description'. The methods listed are write(int b), length(), seek(long pos), close(), writeDouble(double v), writeFloat(float v), and write(int b). The descriptions explain the functionality of each method, such as writing bytes, getting file length, seeking to a specific position, closing the stream, and writing double and float values.

Method	Description
<code>write(int b)</code>	It writes the specified byte to this file.
<code>length()</code>	It returns the length of this file.
<code>seek(long pos)</code>	It sets the file-pointer offset, measured from the beginning of this file, at which the next read or write occurs.
<code>close()</code>	It closes this random access file stream and releases any system resources associated with the stream.
<code>writeDouble(double v)</code>	It converts the double argument to a long using the <code>doubleToLongBits</code> method in class <code>Double</code> , and then writes that long value to the file as an eight-byte quantity, high byte first.
<code>writeFloat(float v)</code>	It converts the float argument to an int using the <code>floatToIntBits</code> method in class <code>Float</code> , and then writes that int value to the file as a four-byte quantity, high byte first.
<code>write(int b)</code>	It writes the specified byte to this file.

NPTEL Online Certification Courses
IIT Kharagpur

So, this is another example that we are going to discuss about writing into a random access file. It is basically just opposite to the previous one, earlier it was reading and writing, first you could write then you can read anyway, whatever it is here. So, these are the different methods we have listed, we have already mentioned this method once. So write as a byte, the length and these are the different method, write double, write float, write in all these methods you can use to write the data into the random access file. Now, let us come to the discussion about how we can do ultimately in a program.

(Refer Slide Time: 14:06)

```
import java.io.IOException;
import java.io.RandomAccessFile;

public class RandomAccessFileExample {
    static final String FILEPATH = "myfile.txt";
    public static void main(String[] args) {
        try {
            writeToFile(FILEPATH, "Data Structures and Algorithms Using Java", 41);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    private static void writeToFile(String filePath, String data, int position)
        throws IOException {
        RandomAccessFile file = new RandomAccessFile(filePath, "rw");
        file.seek(position);
        file.write(data.getBytes());
        file.close();
    }
}
```

And this is a simple program that we can consider for you. So this is okay, because we are using the random access file, these files should be imported into here. So here random access file write demo you can say, and this is the file that we want to store some data into it, let the name of the file is myfile dot txt. So, this is the name of the file where you want to write data and the data that we are, so this the main method, in this main method, we just declare one method called write to file.

Now it require the name of the file, and the message which we want to store and in which position we want to store, because it is a random access file so we have our absolute liberty to write in any position here. So, this basically allow you to write into some positions that is there. Now 41 is the position, if suppose initially file is empty, it will automatically create the bytes for that and then it will create and so that we can write this in a 41 position.

Now before 41, 0 to 40 there may not be any data into there, it can be garbage data or whatever it is there otherwise, you can write some data there also assuming that it is initially

some data there up to 0 to 40 and we want to write something after 41 for an example, here also we can write 0 also so that starting from the beginning we can write if the data contents there in overwrite also. So, this basically is the main method, now we have to define this write to file method we are defining write to file method here.

So, this is the name of the file, this is the message that we want to write and this is basically where we want to write in the random access file. So, now file object file is created giving the file path and we want to create its read write mode. So, it is basically this file can be used both for reading and writing purpose as we need otherwise you could write it only write mode also in this case.

Now, here you can see we position the file pointer into the position so here 41 like, and then we write using this method. Now, here the data that is there so data is basically passed as an argument, this is basically input data that is there, we need to convert into the byte so `getBytes` method for the string that can allow you to convert this into byte. So, this basically converting an array of bytes, which basically write method will be called for and then it will write that bytes into the file and then finally file will be closed.

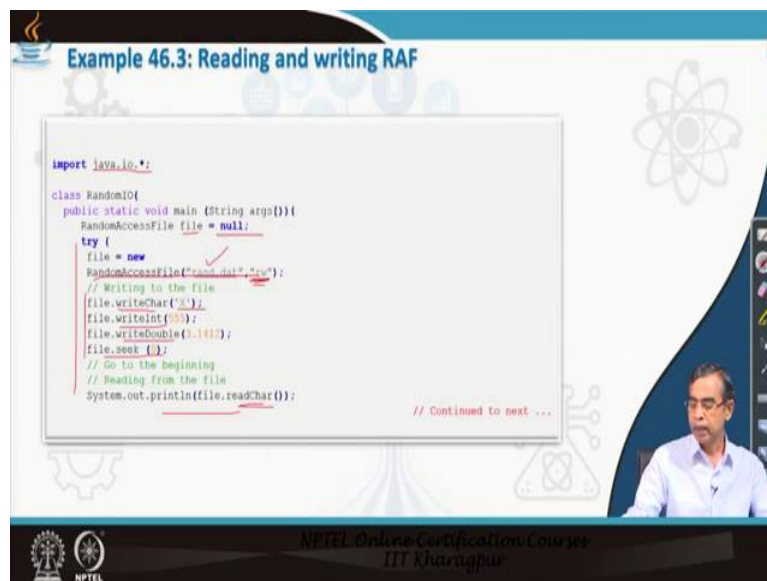
So, the file is ready, you can just simply open the file using any text editor and then see the content of the file will be stored there automatically. Or you can open using read file concept that we have discussed earlier, read character by character and print it on the console so you can do that thing also. So, this is left as an exercise for your own practice. So, now, we have understood about how a random access file can be open and data can be stored in it, and then how a random access file can be open to read data from it. So, these two programs are useful program, which basically gives you an idea about how this can be done.

(Refer Slide Time: 17:30)



Now let us discuss about how read-write operations because random access file is one way advantageous in that sense because it allows you to open both in reading and writing together.

(Refer Slide Time: 17:43)



So, we will consider another example, I am giving few examples so that you can understand these things better. So, this example is basically we will read some data into a random access file, and subsequently the same file will we will write some data into random access file first and then we read the same file to retrieve that data actually, this is the program that we are going to write. Now here, this is the import java dot io dot star includes their random access file is imported here and we create the file object of type random access file, initially it is

null, then here within try block we are just okay file these are random access file constructor is created, this is the name of the file and this is a mode that we want to create.

So, the name of the file is rand dot assuming that rand dot is available in the current working directory and opening the file in read-write mode. Then here few statement you can follow; write char is basically write a character into the file name of the file Rand dot, writeInt and then writeDouble, after writeInt it just position the file pointer to 0, then system dot out dot println find read, now you see go to the 0, here till time it is basically writing mode then here coming here we are reading mode.

So, the same file can be read-write in the same way, so file readChar so it basically read character it will print on the screen, then if you continue the program it will basically continue and you will see the program here.

(Refer Slide Time: 19:24)



The slide displays a code editor window with the following Java code:

```
// Continued on...  
  
System.out.println(file.readLine());  
System.out.println(file.readDouble());  
file.seek(1); // Go to the second item  
System.out.println(file.readInt());  
// Go to the end-end append=false to the file  
file.seek(file.length());  
file.writeBoolean(false);  
file.seek(1);  
System.out.println(file.readBoolean());  
file.close();  
  
catch(IOException e)  
{  
    System.out.println(e);  
}
```

The slide also features a video feed of a presenter in the bottom right corner and the NPTEL logo in the bottom left corner.

Then file readInt, so it basically read the integer then we readDouble from the file so basically there. And then file seek to, we position the file into the second position there, in the second bytes there. So, 0 is in the readChar, 1 is a read and then the second and this one. Then okay file readInt we can again readInt then because in the second position the second byte is basically integer is stored there, we can read again integer, then file seek file length, it basically position the file pointer at the end of the file.

Then writeBoolean is basically we are writing some Boolean value there at the end of the file so, we can put it in and then seek for you can know which is the fourth byte stored there as you mean 2 bytes, 4 bytes for integer, 8 bytes for double and everything, you can seek 4, it

basically go to the byte position and then read Boolean there so, it basically read the Boolean from the position there and finally you can close the file.

Now here, if you little bit position your file position by means of seek method in a controlled manner. So, you can position your file pointer anywhere and from that position you can either read or write whatever you can do it is there. So, this is a good way that the random access file is useful that you can randomly write and read data from there, but you have to be a little bit careful, if you do the position, then overwrite may takes place because, if you want to write something where some data is already there, you can lost that data because it will overwrite whatever it is there.

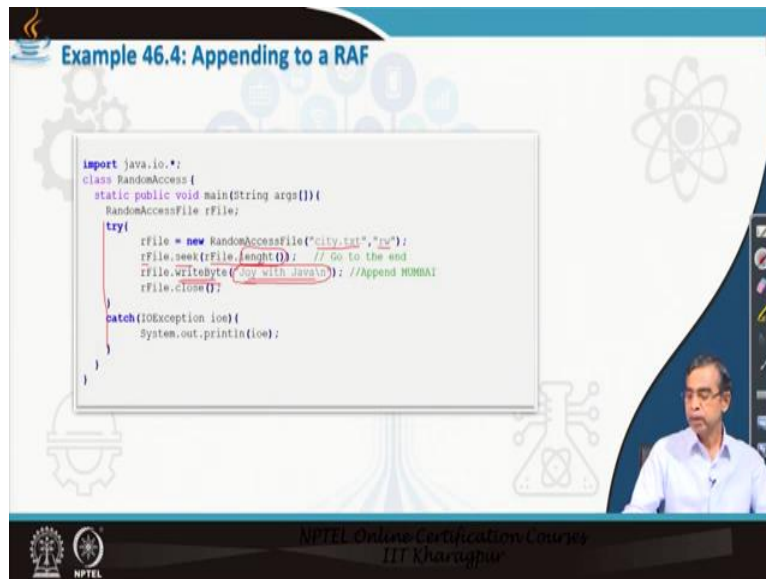
So, that considerations should be taken into account so that you can avoid the data overwriting while you are working with the random access file.

(Refer Slide Time: 21:19)



Now, let us consider another example, which basically gives you how we can append data into random access file. Now, appending data is very simple, because you can open the file both read and write whatever it is there even the write also, you can position your file pointer at the end of the file and then you can start writing. So, appending in this way it is possible. So, this is why it can be open as a read mode, write mode, read and write mode, but there is no any mode like append mode, which allow you to open a random access file.

(Refer Slide Time: 21:54)



The slide displays a code editor window with the following Java code:

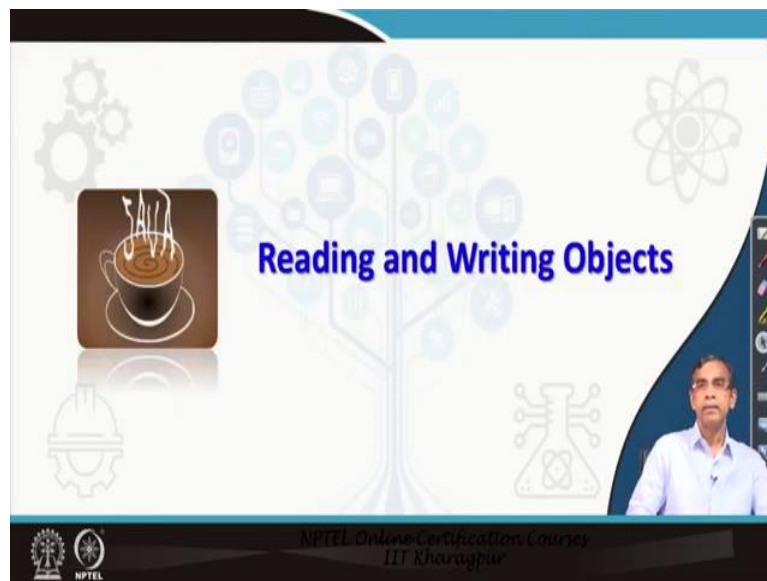
```
import java.io.*;
class RandomAccess {
    static public void main(String args[]) {
        RandomAccessFile rFile;
        try {
            rFile = new RandomAccessFile("city.txt", "rw");
            rFile.seek(rFile.length()); // Go to the end
            rFile.writeBytes("Joy with Java\n"); //Append MUMBAI
            rFile.close();
        }
        catch (IOException ioe) {
            System.out.println(ioe);
        }
    }
}
```

The slide also features a small video inset of a man in a light blue shirt in the bottom right corner. The background has a blue and white theme with various icons like gears and a molecular structure. The footer contains the NPTEL logo and the text 'NPTEL Online Certification Courses IIT Kharagpur'.

Now, this is one example that I can give you for an illustration, how wave file can be opened and then some data can be append. Write the name of the file which you want to process, it is as city dot txt, assume that this file exists and we want to open it in a read-write mode. Now, here, this file you seek the position file pointer position at the length, length is basically what is the amount of byte that is available in the file. So, if we position the file at the length with this that means, it basically position the file pointer at the end of the file. And at the end of the file, we can call the write byte method this passing as a string.

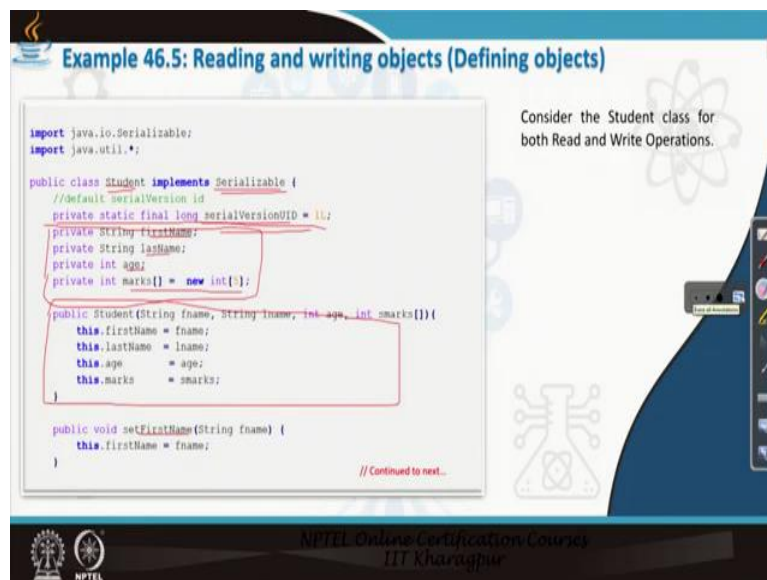
So this thing if you pass then it will automatically convert into the byte and then store this byte into the file so this basically write byte this one. And then this basically says that how the file can be appended, you can later on open it and you can see the readInt content and everything. So, this example, say how the file can be open using random, random access file can be used as an appending purpose.

(Refer Slide Time: 23:14)



Now, this is another very important topics that you can think about, how we can store object user defined object. So, writing objects into a file and later we can read the same object from the same file. So, this is an example that we can cover it, it can give us an idea about how the objects can be stored and how the object can be made persistent rather using random access file mechanism.

(Refer Slide Time: 23:48)



So, let us consider a program, first of all we should consider one user defined class for which we want to store the object, let the user defined class B student, here we just define that class student and here you see implements serializable that is one concept, serializable is a

interface basically is required in order to read and write to be synchronized, what is the meaning of that? Suppose, you store one file with some class definition using one program.

Now you change the class definition say suppose student is a one class for which you have written the data into a file, you want to read the data from the same file, but for other object assuming that this is a person object, which is different from the student, then it basically clash and then your reading writing program will not be synchronized in that sense and therefore, this will leads to an error.

So Java, whenever it is declared a class as serializable, it is checked that whether you can have the serializability access that means the two files are same or not, if it is not there two classes are same or not, if it is not then it will not allow you to write into the same object like this so that file serializable is there. Anyway, so regarding the serializable there is a huge story huge discussion is required actually. Anyways, the serializable to make that your passion the class that you are using, they are matching actually, it is not that older version you are using.

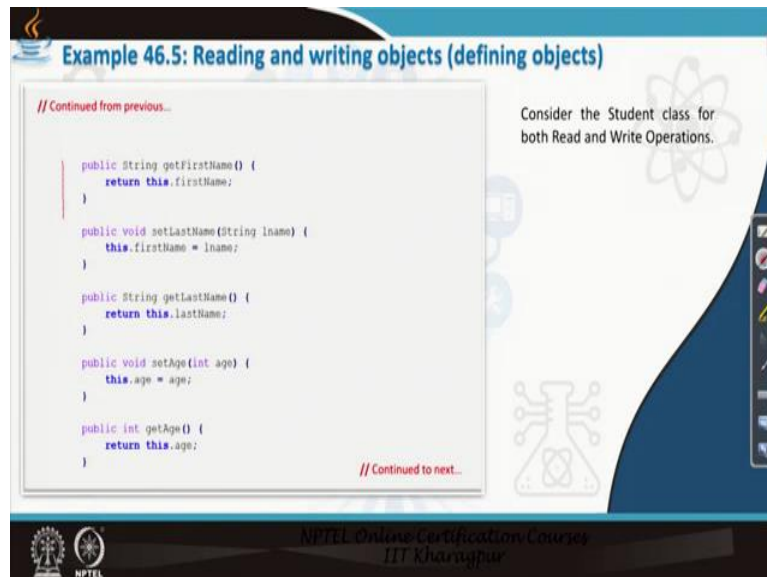
Older version means older definition, older definition means that in earlier version you have certain fields, newer version you add more fields or you have deleted some field that is why the two different definitions actually that is not allowed. So, it should be consistent that means same definition should be used so that object can be of same type that is what important concept it is there. Now, here it is basically, let us define the class first, class student, class student is defined here. Now, here again for this serialization this is the one field that needs to be defined, it is called the serial version UID.

Actually, whenever you run this program and you store this, automatically the Java Virtual Machine create an ID for that. So, you can initialize this ID with any data, but it will be automatically initialized by the Java Virtual Machine for understanding and whenever you read or write it basically check that whether to version same or not that is why for this purpose this field is there, but it is not the field for your actual student object in that sense.

Now, student class has the following fields, the first name, last Name as a string, integer as age and then is a integer as a marks. So, these are the marks obtained by types, obtained by the student in 5 different subject will be stored. So, this is basically the record of a student data that we want to store and we want to read it. And this is a constructor that basically to initialize the data, initialize the student object and few setting methods, setFirstName that

means initialization, the set method get method you can ignore also, these are setter and getter method, I have just given for an illustration, you can ignore it also. So, this is basically the first few points about the class students, there are two more points, few setter methods are there again.

(Refer Slide Time: 27:37)



Example 46.5: Reading and writing objects (defining objects)

// Continued from previous...

```
public String getFirstName() {
    return this.firstName;
}

public void setLastName(String lname) {
    this.firstName = lname;
}

public String getLastName() {
    return this.lastName;
}

public void setAge(int age) {
    this.age = age;
}

public int getAge() {
    return this.age;
}

// Continued to next...
```

Consider the Student class for both Read and Write Operations.

NPTEL Online Certification Courses
IIT Kharagpur

There are a few more setter methods as I mentioned here, these are the setter and getter methods actually, so different fields can be updated and everything. You can ignore if you want actually, this is not necessary, this is not mandatory, for the sake of illustration I have given here, they are not so much important in the context of current discussion. Anyway, so this is basically the definition of class.

Now, what we want to do is, we want to create some objects of this class and store into a file that means writing objects and then later on we want to read the objects from the file. You can create number of objects to be stored there, but in this example, I will consider only one object, later on you can write extend this program to use the multiple objects how it can be stored.

(Refer Slide Time: 28:27)

Example 46.5: Reading and writing objects (defining objects)

```
// Continued from previous...
public void setMarks(int smarks[]) {
    this.marks = smarks;
}

public int[] getMarks() {
    return this.marks;
}

@Override
public String toString() {
    return new StringBuffer(" \nfirst Name : ").append(this.first_name)
        .append(" \nlast Name : ").append(this.last_name)
        .append(" \nAge : ").append(this.age)
        .append(" \nmarks : ")
        .append(Arrays.toString(this.marks)).toString();
}
// Continued to next...
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, here is the write program, that is, this is again continuation of the class declaration. So, this basically here you can see this is another setter method getter method and this basically to convert the string so that we can print all the object into this what is called the format actually. So, this method is basically converting the different field values into the string, so that we can nicely display the student object. Anyway so this completes the declaration of the class students which we can consider for storing and retrieving the data related to this object.

(Refer Slide Time: 29:08)

Example 46.5: Reading and writing objects (create objects and writing)

```
import java.io.FileOutputStream;
import java.io.ObjectOutputStream;

public class WriteObjectDemo {
    private static final String filepath = "D:\\Java\\obj1";
    public static void main(String args[]) {
        WriteObject objectIO = new ObjectIO();
        int n[] = {23, 45, 33, 11};
        Student student = new Student("John", "Frost", 23, n);
        objectIO.writeObjectToFile(student);
    }
}

+

public void writeObjectToFile(Object serObj) {
    try {
        FileOutputStream fileOut = new FileOutputStream(filepath);
        ObjectOutputStream objectOut = new ObjectOutputStream(fileOut);
        objectOut.writeObject(serObj);
        objectOut.close();
        System.out.println("The Object is successfully written to a file");
    } catch (Exception ex) {
        ex.printStackTrace();
    }
}
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, here is a program we are considering about write object demo, first we will write an object into a file and then we will write another method or another program, which basically read object demo. So, let us pass write that class write an object, this is a program that we are

considering. Now let us look at the program first, so this is the program, the name of the program is write object demonstration. Here we use consider name of the file where the data will be stored.

So, D colon Java object is the name of the file where you want to store the data. You can give the name of your own understanding, maybe simple name of the file so that it can be in the current directory, so this is the name of the file. And we start the main method, then write object we are defining another what is called the write object, we are defining one class actually, so write object, object IO, this is the name of the object is the object IO is the class. So, we will discuss the write object to file method for this purpose here.

Now, so write object is basically the object that we want to write into the file actually, write object method it is already declared in the serializable interface that we can use it for there. Now, Int m is a marks obtained by the student and here the student object, one student object is created passing this is the first name, last name, integer age and the marks obtained in 5 different subjects. Now, for this objects IO we call these write object to file, the method of which we have to just define it later on and passing the argument as a student.

So, this is basically it will complete the object IO that we have created writing into that file into there. So, this is basically the object. Now, let us come to the write object to file, this is the object that we want to store, so it is a serializable object, it is basically the student is basically the type of object. You have the object argument is given that means, it can take any object actually as an argument. Anyway, so first we have to create the file as an output stream, a file out, here is a file path.

Now, object out is basically the object that we want to create into it. So, object output stream now, here object output stream is a new class which is basically we have already mentioned this class they are in input byte stream, output byte stream. So, object output stream means, if we want to write some object into the output stream that can be considered. Then object out this is object out, then we write object we can just call the write object method which is defined in object output stream class passing the student object that means, it basically write the object into the file.

Then finally you close the file and this basically completes how we can store an object into a file. So, this is the program that you can think for writing the file. Now, we can consider next

part of the program where we can read the object which is already stored in the file like this one.

(Refer Slide Time: 32:47)

```
import java.io.FileInputStream;
import java.io.ObjectInputStream;

public class ReadObjectDemo {
    private static final String filepath="D:\\Java\\obj1";
    public static void main(String args[]) {
        ReadObject objectIO = new ReadObject();
        objectIO.readObjectFromFile();
    }
    public void readObjectFromFile() {
        try {
            FileInputStream fileIn = new FileInputStream(filepath);
            ObjectInputStream objectIn = new ObjectInputStream(fileIn);
            Student student = (Student) objectIn.readObject();
            System.out.println("student.toString()");
            objectIn.close();
            System.out.println("The Object was successfully read from the file");
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}
```

So, this is a next part of the program which we can consider reading the object from the file. So, it is basically the program read object demo and here is basically read object, we just create object Infosys, the new read object, so this is the read object one that object is created, it is from the file, this is the name of the file format, we want to read the object here and then lead object from file this method we want to define. So, we are defining this method here, these are main method that we write and this is the method which we have to discuss here.

Now, we create the file input string because we want to create the object from this file which is stored here and then we want to create as an object input stream. So, this is the input byte stream class can be considered, this is the name of the object input stream, basically it is a wrapper class which basically is the interface from your program to the file and this interface is basically reading objects or writing objects in the last version.

Now here, a student object is declared here and then read object method, here the read object method is called for this object in so that it reads the object from the input file and this is casted as a student type so the student object can be stored there so this is basically read the object which is basically of type student and then object reading is there then that reading object you can print so, that you can see that system dot out dot print ln student if you write it.

For example here, System dot out dot print ln that object it basically print the object that you could retrieve from the program. Anyway, so, this program explained you how the reading and writing can be done from a file. So, this is basically reading object and writing object and here we have considered only simple file sequential access file. You can repeat the same program, but for the writing the object into the random access file also. So, this is left as an exercise for you so that you can study it later.

(Refer Slide Time: 35:14)



So, this is regarding the file handling, we have learned many topics in this module regarding input-output stream and then character stream, finally the file as an IO and then random access file. So, if you want to learn to more then definitely Oracle tutorial documentation is good for you, this complete reference covers many more other things also that you can and all the program discussion that we have made here and many more things also related to the random access file, you can give the first link that I have given.

With these things, I would like to stop it here today. Thank you very much. We will study next module regarding the next topics in the next video. Thank you very much.