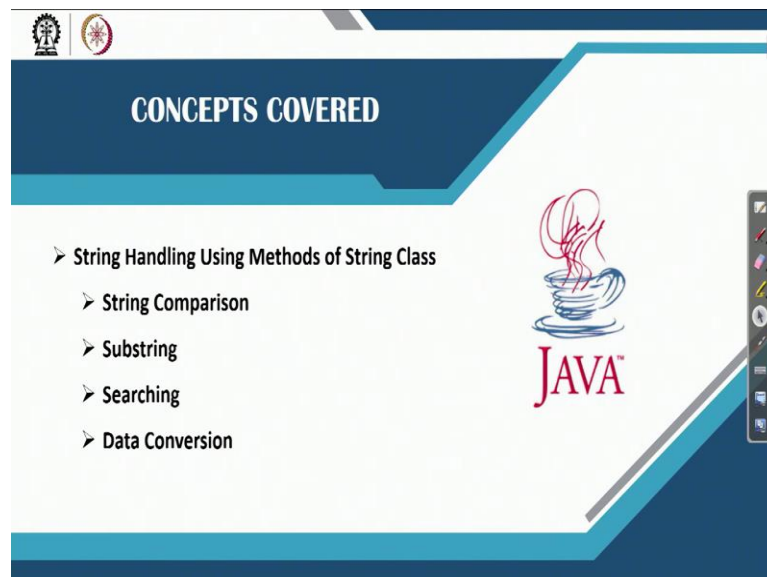


Data Structures and Algorithms Using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Lecture 57
Application of String

So, we are discussing String and in the last video lecture we have learned few aspects about string and the different objects of it and the how to instantiate string objects and then different methods simple methods, which basically essential to manipulates strings. So, in today's lecture, we will cover many more interesting aspects of string, which basically makes your understanding complete.

(Refer Slide Time: 00:47)



So, today we will discuss about some other additional methods, those are there to handle strings, they are an namely string comparison, how we can compare two strings or how to create substring, how to search some part of the string in a string and finally the conversion, this is very important because a data of any type can be converted to string, also a string can be converted to data of any primitive type. So, these are data conversion and what are the methods are there, which are defined in class string will be discussed in this video lecture.

(Refer Slide Time: 01:25)

The slide features a central title "String Comparison" in blue text. To the left is an icon of a coffee cup with steam. The background is decorated with various icons like gears, a tree, and a hard hat. A small video feed of a presenter is visible in the bottom right corner. The footer includes the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

The slide is titled "Methods of comparing strings" and contains a table with two columns: "Method" and "Description".

| Method | Description |
|-----------------------------------------------------|-----------------------------------------------------------------------------------------|
| <code>equals(Object anObject)</code> | Compares this string to the specified object. |
| <code>equalsIgnoreCase(String anotherString)</code> | Compares this String to another String, ignoring case considerations. |
| <code>contains(CharSequence s)</code> | Returns true if and only if this string contains the specified sequence of char values. |
| <code>contentEquals(CharSequence cs)</code> | Compares this string to the specified CharSequence. |
| <code>contentEquals(StringBuffer sb)</code> | Compares this string to the specified StringBuffer. |
| <code>compareTo(String anotherString)</code> | Compares two strings lexicographically. |
| <code>compareToIgnoreCase(String str)</code> | Compares two strings lexicographically, ignoring case differences. |
| <code>matches(String regex)</code> | Tells whether or not this string matches the given <u>regular expression</u> . |
| <code>endsWith(String suffix)</code> | Tests if this string ends with the specified suffix. |

The footer includes the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

So, let us first see string comparison, it is very important. This is because we have to check whether two strings are exactly same or not. Now, this kind of comparison is essential in many application development. So, that is why many methods are defined there. So, for the two, in order to compare two strings, the methods are basically here I have listed few important methods, which basically you can consider to compare the string.

You can compare two strings, just giving one string as argument and calling this method for the string for which you want to compare, you can compare without, I mean, considering the cases with a lowercase uppercase ignoring or you can consider all changing into uppercase and lowercase. So, there are many ways the string can be compared. So, these are the few methods that we have listed here.

(Refer Slide Time: 02:19)

Example 57.1 : equals()

Boolean equals (Object o) Compares this string to the specified object.

```
public class StringEqualsDemo {
    public static void main (String args[]) {
        String text1 = "DATA STRUCTURE WITH JAVA";
        String text2 = "DATA STRUCTURE WITH C++";
        String text3 = "DATA STRUCTURE WITH JAVA";
        boolean output1 = text1.equals(text2);
        boolean output2 = text1.equals(text3);
        System.out.println(output1);
        System.out.println(output2);
    }
}
```

Output

```
false
true
```

NPTEL Online Certification Courses
IIT Kharagpur

And now, we will discuss about how the different methods that it can work for any programmer in their programs. So, here is an example we have to test using the equals method to compare two string. So, here text 1, text 2 and text 3 are the three string objects that is created like this and as you can see, text 1 and text 2 are the same actually.

Now, here we can just check it and see here equals text 2, text 1. Now, so this equals method is basically call for the text 1, that means this one and it basically compare this text one with text 2, the equal method returns true if they are equals otherwise they return false, now in this case text 1 and text 2 as we see they are not same, so it basically return false.

So, as you see this return false on the other hand, if we can out, you could compare again text 1 and text 3, text 1, this one string and text 3 this thing, because that two strings are same. So, it basically return true. Now, you can call this method alternatively text 3 then dot equals text 1 or text 2 dot equals text 3 the different way you can write and also you can compare text 1 equals give some string here within double quote that also will work whatever the way you can check it. Now, so this is an example about how using equals method you can compare two strings and then you can get the results.

(Refer Slide Time: 04:09)

Example 57.2 : equalsIgnoreCase()

Boolean equalsIgnoreCase (String s): Compares string to another string, ignoring case considerations.

```
Code
public class StringequalsIgnoreCaseDemo {
    public static void main(String args[]) {
        String text1 = "DATA STRUCTURE WITH JAVA";
        String text2 = "DATA STRUCTURE WITH C++";
        String text3 = "DATA STRUCTURE WITH JAVA";
        String text4 = "data structure with c++";
        String text5 = "data structure with java";
        boolean output1 = text1.equalsIgnoreCase(text2);
        boolean output2 = text1.equalsIgnoreCase(text3);
        boolean output3 = text1.equalsIgnoreCase(text4);
        boolean output4 = text1.equalsIgnoreCase(text5);
        // Continued to next...
    }
}
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, here is another programs this basically we can compare, but we will compare while we will compare will ignore cases. So, equals method with additional features of ignore cases. Now, here we create I mean few objects for our testing purpose and then we call the different way that equal ignore cases methods and if we run this program.

(Refer Slide Time: 04:39)

Example 57.2 : equalsIgnoreCase()

```
Code
// Continued on...
System.out.println(output1);
System.out.println(output2);
System.out.println(output3);
System.out.println(output4);
}
}
```

Output

```
false
true
false
true
```

NPTEL Online Certification Courses
IIT Kharagpur

So these are continuation of this program if you run this program and print it and you can see these result you will get it. So, you can run this program of your own and you can see the output and you can convince yourself how you are getting output and there are many more comparison methods are there you can also apply them and you can check how they work in general, so this is a comparison.

(Refer Slide Time: 05:03)

Example 57.3 : compareTo()

int compareTo(String s) Compares two string lexicographically.

```
public class StringCompareDemo {
    public static void main(String args[]){
        String text1 = "DATA STRUCTURE WITH JAVA";
        String text2 = "DATA STRUCTURE WITH C++";
        String text3 = "DATA STRUCTURE WITH JAVA";
        String text4 = "data structure with c++";
        String text5 = "data structure with java";
        int output1 = text1.compareTo(text1);
        int output2 = text1.compareTo(text3);
        int output3 = text1.compareTo(text4);
        int output4 = text1.compareTo(text5);

        System.out.println(output1);
        System.out.println(output2);
        System.out.println(output3);
        System.out.println(output4);
    }
}
```

Output

```
7
0
-32
-32
```

Handwritten annotations: 'ABD' and 'ABC' are written in red. An arrow points from '0' to 'ABC', and another arrow points from '-32' to 'ABD'.

NPTEL Online Certification Courses
IIT Kharagpur

Now, another aspects of this comparison method is very important. So, is a compareto method is also there, here again few strings and we call the compareto methods. Now, here this is, this method is different, so that it basically compared to with respect to Unicode character format. So, that means, actually every character in Java you know, so here every character in Java basically stored in the form of a 16 bits Unicode like.

So, they basically compared with respect to Unicode. For example, here text on and takes to this string these two strings as it is here, when compared it basically returned the results the seven, these basically results according to the Unicode formalism. Now, if the two strings are equal for example, text 1 and text 3 are in this case, so text 1 and text 3 these thing and this strings are same. So, it returns 0. So, this output to a 0 here.

Now, if two strings are equal, then they are in a Unicode formalism and if it is compare it will give 0. On the other hand, here if you see text 4 and text 1 is companion text 4, it gives minus 32. Now, actually the concept is that if they are basically considered about lexicographic ordering.

Lexicographic ordering means, for example A, B and D, so this is a one string and a b c, now in lexicographic order ABC comes before this one, but it follows there, if it is there, now if you compare this string with this string 1, it will give negative and if so they are, so positive it gives, if they are in lexicographic order. But for example here ABC when it is compared with ABC, this will return a positive, but ABC when ABD return compared with ABC, it will give negative and ABCD when compared with ABT it will give 0.

So, this basically according to our lexicographic order formalism it will give you either a positive value negative value or 0 in that way you can check that whether a string comes before one string or comes after string. So, if it is a before string positive and after string is negative that is a concept that you can think for. So, this is the idea about that string comparison how it works.

(Refer Slide Time: 07:44)

Example 57.4 : compareToIgnoreCase()

int compareToIgnoreCase (String s) Compares two string lexicographically without case matching.

```
public class StringCompareCaseDemo {
    public static void main (String arg []) {
        String text1 = "DATA STRUCTURE WITH JAVA";
        String text2 = "DATA STRUCTURE WITH C++";
        String text3 = "DATA STRUCTURE WITH JAVA";
        String text4 = "data structure with c++";
        String text5 = "data structure with java";
        int output1 = text1.compareToIgnoreCase (text2);
        int output2 = text1.compareToIgnoreCase (text3);
        int output3 = text1.compareToIgnoreCase (text4);
        int output4 = text1.compareToIgnoreCase (text5);
        System.out.println (output1);
        System.out.println (output2);
        System.out.println (output3);
        System.out.println (output4);
    }
}
```

Output

```
7
0
7
0
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us consider another program to convince what I told earlier. So, compared to here actually compared to can be again with additional feature called ignore case and then you can take that if we ignore the cases it will give the, sub for this output and for this is a comparison that we have made here.

So, this way, so staying competitive is useful in many applications, particularly when you have to write your sorting, algorithms where the string needs to be considered then you can think about these are the method that you can.

(Refer Slide Time: 08:16)



Now let us consider about some other manipulation regarding the substring competition. Now, given a string, we can check whether some substring is there or we can extract some substring starting from a given index or in between some index and etc.

(Refer Slide Time: 08:33)

The slide is titled "Methods for substring computation" and contains a table with two columns: "Method" and "Description". The table lists seven methods: `replace(char oldChar, char newChar)`, `replace(CharSequence target, CharSequence replacement)`, `replaceAll(String regex, String replacement)`, `replaceFirst(String regex, String replacement)`, `split(String regex)`, `split(String regex, int limit)`, and `startsWith(String prefix)`. A video feed of a presenter is in the bottom right corner. The footer contains the NPTEL logo and the text "NPTEL Online Certification Courses IIT Kharagpur".

| Method | Description |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------|
| <code>replace(char oldChar, char newChar)</code> | Returns a new string resulting from replacing all occurrences of oldChar in this string with newChar. |
| <code>replace(CharSequence target, CharSequence replacement)</code> | Replaces each substring of this string that matches the literal target sequence with the specified literal replacement sequence. |
| <code>replaceAll(String regex, String replacement)</code> | Replaces each substring of this string that matches the given regular expression with the given replacement. |
| <code>replaceFirst(String regex, String replacement)</code> | Replaces the first substring of this string that matches the given regular expression with the given replacement. |
| <code>split(String regex)</code> | Splits this string around matches of the given regular expression . |
| <code>split(String regex, int limit)</code> | Splits this string around matches of the given regular expression . |
| <code>startsWith(String prefix)</code> | Tests if this string starts with the specified prefix. |

So, there are many methods regarding the substring computations and then changing a part of the string and giving another string all those concepts it is there I have listed few methods here like replace split and then starts with, there are many ways the substring can be computed.

(Refer Slide Time: 8:53)

Methods for substring computation

| Method | Description |
|--------------------------------------------------------|----------------------------------------------------------------------------------------------------------|
| <code>startsWith(String prefix, int toffset)</code> | Tests if the substring of this string beginning at the specified index starts with the specified prefix. |
| <code>subSequence(int beginIndex, int endIndex)</code> | Returns a new character sequence that is a subsequence of this sequence. |
| <code>substring(int beginIndex)</code> | Returns a new string that is a substring of this string. |
| <code>substring(int beginIndex, int endIndex)</code> | Returns a new string that is a substring of this string. |

NPTEL Online Certification Courses
IIT Kharagpur

And here is one method you can take substring integer beginindex and endIndex, so you can understand that, so it basically for a given string and it will retrieve you a part of the string of that string. So, this a substring can be obtained also, there are many ways the different methods which you can apply to solve any problem, which is relevant in order for the string manipulation.

(Refer Slide Time: 09:19)

Example 57.5 : substring()

String.substring(int i) Returns the sub string starting from the character with i-th index.

```
Code
public class SubstringDemo {
    public static void main(String args[]) {
        String text = "A STRUCTURE WITH JAVA";
        String data = text.substring(3);
        System.out.print(data);
    }
}
```

Output
A STRUCTURE WITH JAVA

NPTEL Online Certification Courses
IIT Kharagpur

Now, here is an example. So, this is a substring demo, this is a string 1 input string and substring 3 that means we want to create a substring starting from the third index. So starting from here, so as you can see, so starting means 0 1 2 3. So starting from here, actually right. So, starting from here and then it basically gives all the remaining part of the string as an

output. So, as you see this is output for this method and you can see we just substring return the substring which is stored in this new string object. So, these are the one example.

(Refer Slide Time: 10:03)

Example 57.6 : substring()

String substring(int i, int j) Returns the substring from character with i to j-1 indices.

```
Code
public class SubstringmyDemo{
    public static void main(String args[]){
        String text = "DATA STRUCTURE WITH JAVA";
        String data = text.substring(5, 14);
        System.out.print(data);
    }
}
```

Output
STRUCTURE

NPTEL Online Certification Courses
IIT Kharagpur

There is another example, it basically do the same thing and it okay we can say the length, it should be substring actually takes dot substring 5 and 14. So, it basically this is not correct, we should write a substring method here. So, 5 and 14. So, it will return the substring between the 5 and 14 in the input string actually, so this, the output you will get it.

(Refer Slide Time: 10:35)

Methods of searching

| Method | Description |
|-----------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------|
| <code>indexOf(int ch)</code> | Returns the index within this string of the first occurrence of the specified character. |
| <code>indexOf(int ch, int fromIndex)</code> | Returns the index within this string of the first occurrence of the specified character, starting the search at the specified index. |
| <code>indexOf(String str)</code> | Returns the index within this string of the first occurrence of the specified substring. |
| <code>indexOf(String str, int fromIndex)</code> | Returns the index within this string of the first occurrence of the specified substring, starting at the specified index. |
| <code>lastIndexOf(int ch)</code> | Returns the index within this string of the last occurrence of the specified character. |
| <code>lastIndexOf(int ch, int fromIndex)</code> | Returns the index within this string of the last occurrence of the specified character, searching backward starting at the specified index. |
| <code>lastIndexOf(String str)</code> | Returns the index within this string of the last occurrence of the specified substring. |
| <code>lastIndexOf(String str, int fromIndex)</code> | Returns the index within this string of the last occurrence of the specified substring, searching backward starting at the specified index. |

NPTEL Online Certification Courses
IIT Kharagpur

Now, searching of a part of a substring, also may be interesting one, operation that needs to be performed on many string objects, there are many methods in this for this purpose that is

already developed in string class, I have listed here all these methods are self-explanatory, I do not want to discuss explicitly all these methods here and there are many methods.

(Refer Slide Time: 11:01)

The slide is titled "Example 57.7 : indexOf() method". It features a description of the method: "int indexOf (String s) Returns the index within the string of the first occurrence of the specified string." Below this, there is a code editor window labeled "Code" containing the following Java code:

```
public class StringSearchDemo{
    public static void main(String args[]){
        String text = "DATA STRUCTURE WITH JAVA";
        int output = text.indexOf("ITE");
        System.out.print(output);
    }
}
```

Below the code editor is an "Output" window showing the result "15". The slide also includes a small video inset of a man in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

So, now, let us see one example applying the substring concept that are rather indexing ((11:07)). So, there is a method called indexof for which an argument is an input string and this method can be called for a given string, it basically return the index within the string of the first occurrence of the specified string.

So, what is my idea about, so suppose this is the string and this is the argument. So, we have to find the index from higher this string occurs. So, these basically 0 to this one, as you see this is a 15. So that mean this thing occurs at 15th index position of this thing. So, this is the output. So, like this that many methods are there and all the methods are easy to understand, only you have to apply and then practice then you can understand the strength of these methods there.

(Refer Slide Time: 11:55)

The slide is titled "Example 57.8 : indexOf() method". It contains the following text and code:

int indexOf (String s, int i) Returns the index within the string of the first occurrence of the specified string, starting at the specified index.

```
public class Example{
    public static void main(String args[]){
        String text = "DATA STRUCTURE WITH JAVA";
        int output = text.indexOf('T', 4);
        System.out.print(output);
    }
}
```

Output: 6

The slide also features a small video inset of a man in the bottom right corner and logos for NPTEL and IIT Kharagpur at the bottom.

Now, this is another, this is another example this is polymorphic method of the indexOf method. So, here basically there are there are many occurrence of substring that is there where we have considered substring of only one character T. So, 4 onwards it will basically try to find if there are substring of this kind is there or not.

So, this basically this is within three, so this is the 4. So, these basically print that at this t after index 4 which occurs there it is 6, there are many R, but it will only print the first occurrence. So, if I say 10 then it will print this in index of this T and so on. So, that is that method is basically to tell you the occurrence of a particular part of the string, here again you can write with after 4, so it will be give. There may be many with as a substring it will give you the first occurrence of the string.

(Refer Slide Time: 12:53)

Example 57.9 : lastIndexOf()

int lastIndexOf (String s) Returns the index within the string of the last occurrence of the specified string.

```
public class Example{
    public static void main(String args[]){
        String text = "DATA STRUCTURE WITH JAVA";
        int output = text.lastIndexOf("A");
        System.out.print(output);
    }
}
```

Output: 17

NPTEL Online Certification Courses
IIT Kharagpur

And here again like indexOf there is a lastIndexOf that means it is just opposite one. So, T is starting from here it basically what is a index of the last T from the beginning. So, it is basically 17 that it can print as you see it here.

(Refer Slide Time: 13:17)

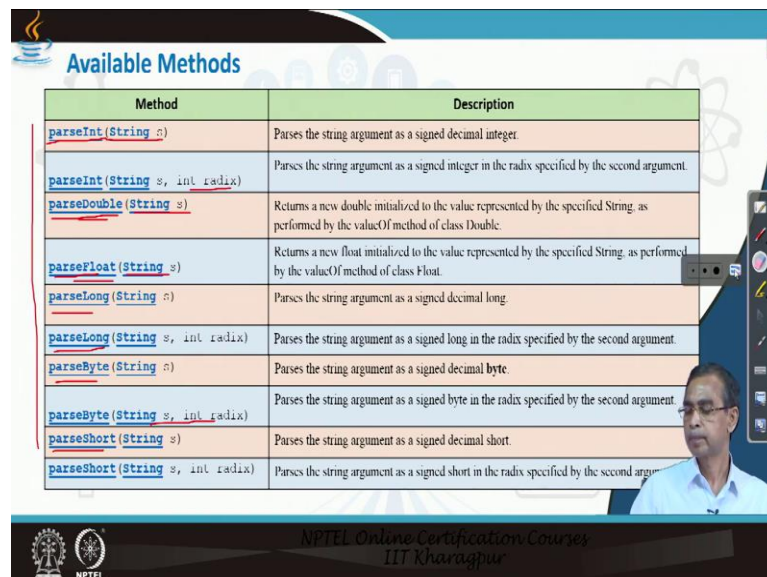
Data Conversion from String

NPTEL Online Certification Courses
IIT Kharagpur

Like there are many more methods out there. So, regarding the different ways the particular portion of a string can be accessed or it can be retrieved whatever the idea it can be there. Now, this is a one important topics that is required to discuss in detail, it is called the data conversion from string. I told you that string is one versatile, for which it basically embed any type, whether any primitive data type or any other type also can be there. So, the conversion

form a string to a desirable data type and vice versa is an important, what is a concept for either Java developer, gave many methods for the purpose.

(Refer Slide Time: 13:59)



| Method | Description |
|----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>parseInt(String s)</code> | Parses the string argument as a signed decimal integer. |
| <code>parseInt(String s, int radix)</code> | Parses the string argument as a signed integer in the radix specified by the second argument. |
| <code>parseDouble(String s)</code> | Returns a new double initialized to the value represented by the specified String, as performed by the <code>valueOf</code> method of class <code>Double</code> . |
| <code>parseFloat(String s)</code> | Returns a new float initialized to the value represented by the specified String, as performed by the <code>valueOf</code> method of class <code>Float</code> . |
| <code>parseLong(String s)</code> | Parses the string argument as a signed decimal long. |
| <code>parseLong(String s, int radix)</code> | Parses the string argument as a signed long in the radix specified by the second argument. |
| <code>parseByte(String s)</code> | Parses the string argument as a signed decimal byte. |
| <code>parseByte(String s, int radix)</code> | Parses the string argument as a signed byte in the radix specified by the second argument. |
| <code>parseShort(String s)</code> | Parses the string argument as a signed decimal short. |
| <code>parseShort(String s, int radix)</code> | Parses the string argument as a signed short in the radix specified by the second argument. |

So, here are a few examples that you can consider and possibly you are already familiar to this method is `parseInt`. That means, given a string we want to parse or you want to convert the string value of this one. So, here `parseInt` with radix also radix means whether it is a decimal or octadecimal or hexadecimal. So, that also you can parse that means, from the given string we will be able to get the numbers in a particular radix form and the `parseDouble` it is just a opposite integer that means, if you want to parse the double value from a given string and here is a `parseFloat` from the given string, we can pass the value of float, `parseLong`.

So, for all different methods `parseByte`, `parseByte` with radix and then `parse sort`. So, for all the different type of primitive data, there is a parsing method. This means that from a given steam will be able to parse up any numeric data type.

(Refer Slide Time: 15:13)

The slide displays a Java code snippet and its execution output. The code is as follows:

```
public class StringToIntegerDemo {
    public static void main(String args[]) {
        String number = "95";
        int num = Integer.parseInt(number);
        int output = num + 5;
        System.out.println(output);
    }
}
```

The output of the program is 100. The slide also features the NPTEL logo and the text 'NPTEL Online Certification Courses IIT Kharagpur' at the bottom.

Now, here is an example that we can think about. So, here how we can convert string to integer demo. So, here we just store this as string as you can see this number is basically here in this statement 95 within double quote, it is a string, but if you write string number equals 95, it is an error, because a numeric value cannot be used as an initialization because the constructor cannot be create for an object of 95 line.

Now, here you see we call the paring, this string number and this parsing method actually is defined in the class called, what is called integer wrapper object, wrapper class, wrapper class you know there are many numeric wrapper class integer float, where the past character is basically in capital letter, this basically wrapped an integer value into an object. So, integer float wrap a value into a floating point value like this one.

Now, here actually num it basically this method parsing method parse a string and return the integer value because we call parsing for an integer value, so it is like and then we can okay use the obvious arithmetic calculation here num plus 5. So, 95 actually will be returned and if the output will be 100 as you see the result will be 100. So, this basically shows how for me string and integer value can be converted, likewise there are many examples that we can also follow.

(Refer Slide Time: 16:58)

The slide displays a Java code snippet and its output. The code is as follows:

```
public class StringToFloatDemo{
    public static void main(String args[]){
        String number = "99.99";
        float num = Float.parseFloat(number);
        float output = num + (float)0.01;
        System.out.println(output);
    }
}
```

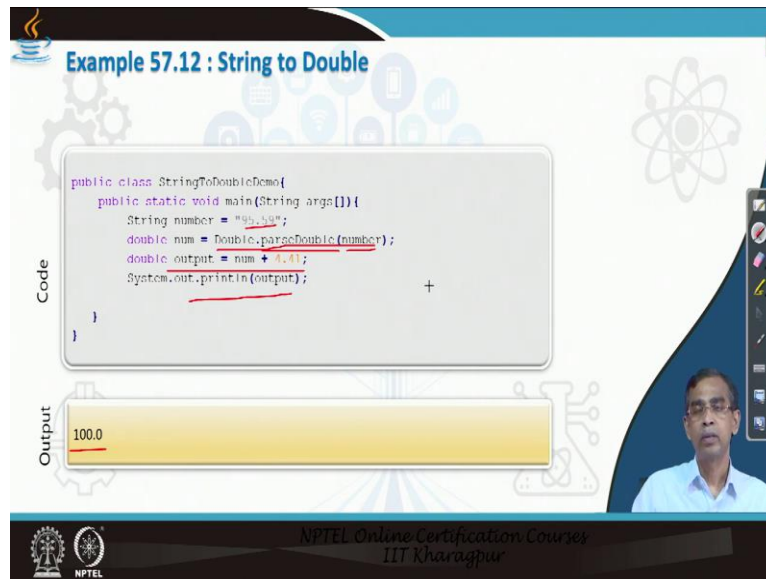
The output of the program is 100.0. The slide also features a small video inset of a man in the bottom right corner and the NPTEL logo in the bottom left corner.

Here, so parseFloat and here we passed the string this thing is here, this thing is basically a floating point value and these parse float we call for the float wrapper object because we want to convert the, these basically here actually, you know, so this is a stream, but is wrapped into the form of a float object.

So, that is why we have to call the parseFloat method for the float object for each number will be the argument as a string. Now, these basically return a floating point number, now so this num plus again if we want to add this number or num within another floating point value, this is because it is just like usually floating point numbers and you can do it and then output you will get it like this is output.

So, you have learned about how the parsing is possible from the string to get the required input required values in a particular type. So, this is a different method, which is there, there are many methods I have already listed. So those can be considered and you can test with your own program.

(Refer Slide Time: 18:16)



The slide displays a Java code snippet for converting a string to a double. The code is as follows:

```
public class StringToDoubleDemo{
    public static void main(String args[]){
        String number = "99.99";
        double num = Double.parseDouble(number);
        double output = num + 0.01;
        System.out.println(output);
    }
}
```

The output of the program is shown as 100.0.

NPTEL Online Certification Courses
IIT Kharagpur

And these are same example here we can see parse double these are number is string, this is a double value, this is a double and float basically same only the 8 byte or that is a 16 byte that matters, because you can store as a double much more bigger number here and this is output that you can get and it is output that you can check it. So, these are data conversion from string to the number, numeric values the opposite way conversion is also possible.

(Refer Slide Time: 18:50)



The slide features a coffee cup icon with steam rising from it, symbolizing data conversion to a string. The title of the slide is "Data Conversion to String".

NPTEL Online Certification Courses
IIT Kharagpur

Methods of data conversion

| Method | Description |
|-------------------------------------------------|------------------------------------------------------------------------------------------------------|
| <code>Integer.toString()</code> | Returns a String object representing this Integer's value. |
| <code>Integer.toString(int i)</code> | Returns a String object representing the specified integer. |
| <code>Integer.toString(int i, int radix)</code> | Returns a string representation of the first argument in the radix specified by the second argument. |
| <code>Float.toString()</code> | Returns a string representation of this Float object. |
| <code>Float.toString(float f)</code> | Returns a string representation of the float argument. |
| <code>toString()</code> | Returns a string representation of this Double object. |
| <code>toString(double d)</code> | Returns a string representation of the double argument. |

NPTEL Online Certification Courses
IIT Kharagpur

The opposite way conversion is possible. So, it is basically data to string where a data can be converted to a string type. Now, let us see the different methods in this direction. These are the different methods which you can call and the method is basically toString. toString is a very important one method which is defined in the string class to convert everything which is there in the form of string, if it is already string there, it is always there. So, but if you pass any value as an argument integer here the integer will be converted to a string and here float can be converted to a string.

So, any type of values can be converted to a string, all these conversion value is, I mean this method is basically called the toString method is basically the method for the class, but it can be called for any type of object, because it is the super methods actually, it is basically defined in object class, class object and it basically implemented in all methods, all classes like string and other so this way it is coming. Now let us have some example, so that you can understand about these concepts better. How this can be used.

(Refer Slide Time: 20:09)

| Method | Description |
|----------------------------------------------------------|--------------------------------------------------------------------------------------|
| <code>valueOf(boolean b)</code> | Returns the string representation of the boolean argument. |
| <code>valueOf(char c)</code> | Returns the string representation of the char argument. |
| <code>valueOf(char[] data)</code> | Returns the string representation of the char array argument. |
| <code>valueOf(char[] data, int offset, int count)</code> | Returns the string representation of a specific subarray of the char array argument. |
| <code>valueOf(double d)</code> | Returns the string representation of the double argument. |
| <code>valueOf(float f)</code> | Returns the string representation of the float argument. |
| <code>valueOf(int i)</code> | Returns the string representation of the int argument. |
| <code>valueOf(long l)</code> | Returns the string representation of the long argument. |
| <code>valueOf(Object obj)</code> | Returns the string representation of the Object argument. |

Handwritten notes on the slide include: "class Student" with an arrow pointing to a diagram of a class structure, and "println()" with an arrow pointing to a code snippet. The diagram shows a class structure with a circle containing "Student" and a list of members.

And there are a few more method also value of. So, is basically if we call some for a given string the value of and it is a Boolean it basically convert to that string. So, if they return the string representation of this Boolean. So, from the Boolean value to string it will convert, so here different, here character to be string, here character data is a part of the portion can be represented as the string, a double can be converted to a string float value can be converted in the stringInt long and so on.

Here object also can be converted in the string this means that any type of value, if you passed it will be it will convert into the string and this is an interesting properties that I will explain. So, idea is that possibly if you declare any class, so this you can declare any class let it be say class student.

So, suppose this is the student class you have declared, there are many members some is integers, some is long, some is character arrays and many more things, some string maybe. Now, you can write a println method of system dot out possibly you have already checked we have done it while we are discussing many collection the entire collection also.

Now, so it is basically a let S is a (())(21:38) of type student, now if we print S for this, then this basically print the string representation of all the members which is there. So, all integers all the floating point values or other data types whatever it is there, they basically converted to the string and finally, print l will basically turn into to string to string representation and ultimately it will display on the console. So, this concept is basically is there actually. Now, let us see some illustration, so that we can understand this concept better.

(Refer Slide Time: 22:16)

The slide displays a Java code snippet and its execution output. The code defines a class `IntegerToStringDemo` with a `main` method. It declares an integer `i` with the value 10, converts it to a string `s` using `String.valueOf(i)`, and prints `s`. It then performs an arithmetic operation `i + 10` and prints the result. The output shows the string `10` and the integer `20`.

```
public class IntegerToStringDemo{
    public static void main(String args[]){
        int i=10;
        String s=String.valueOf(i);
        System.out.println(i+10); // + is binary plus operator
        System.out.println(s+20); // + is string concatenation operator
    }
}
```

Output

```
10
20
```

NPTEL Online Certification Courses
IIT Kharagpur

And this is one example here string value of `i` if `i` is an integer 20. So, it basically `s` represents 20 in the form of a string and `i` plus 10, now here you see `i` is an integer 10 is an integer value because `i` is 10 here and this plus is basically is a arithmetic operation. So, in this case it basically 30, so the result is 30 and `println` will print this 30 in the form of string `println` always convert into string, because our argument that is required for the `println` is always a string.

So, this thing is printed here as a, this data output is printed as a string because it is converted. Now, here `s` is basically string which is basically 20 and then concatenation 20. So, 20 and 20 will be concatenated So, new string will be produced. So, here basically 20 20. So, the result will be 20 20. So, these are important property that you can think about how the string conversion takes place for any type of data.

(Refer Slide Time: 23:19)

The slide displays a Java program for converting a float to a string. The code is as follows:

```
public class FloatToStringDemo {
    public static void main(String args[]) {
        float f = 15.5f; // F is the suffix for float
        String s = String.valueOf(f);
        System.out.println(s);
    }
}
```

The output of the program is 15.5.

NPTEL Online Certification Courses
IIT Kharagpur

Now, here is another example that you can see is basically floating point value. So, value of, we convert the string and represent return a string. So, we represent so, this basically 15.5 in this case, this is basically a string representation of this value.

(Refer Slide Time: 23:44)

The slide displays a Java program for converting an integer to binary, octal, and hexadecimal strings. The code is as follows:

```
// The following program demonstrates binary, hexadecimal, and octal conversion:
class StringConversions {
    public static void main(String args[]) {
        int num = 19648;
        System.out.println(num + " in binary: " +
            Integer.toBinaryString(num));
        System.out.println(num + " in octal: " +
            Integer.toOctalString(num));
        System.out.println(num + " in hexadecimal: " +
            Integer.toHexString(num));
    }
}
```

The output of the program is:

```
19648 in binary: 100110011000000
19648 in octal: 46300
19648 in hexadecimal: 4cc0
```

NPTEL Online Certification Courses
IIT Kharagpur

Now, here is a few more examples that is interesting. So, this is an integer number and here is a number we are printing in a binary, but here you see integer to binary string, this basically convert into the binary form and it will basically represent as a string. So, here this is the input, the output will be this one.

This is a binary representation of the one, but two octal string that also you can do, so this means it will convert in the octal form and this is octal representation of this number and here

also hexadecimal, this is a hex, so the value of the string can be converted into hexadecimal form and this is the hex form of the string. So, from one type of form to another type, we will be able to convert using this method.

(Refer Slide Time: 24:34)

Example 57.16: Numbers as command line arguments

```
class CommandLineDemo {
    public static void main(String args[]) throws IOException {
        System.out.println("Number of arguments: " + args.length);
        int sum = 0;
        if(args.length == 0)
            return;
        for(int i=0; i<args.length; i++)
            sum + Integer.parseInt(args[i]);
        System.out.println("Sum = " + sum);
    }
}
```

Hint: Run the program with following:

```
java CommandLineDemo
java CommandLineDemo 1 2 3 4
java CommandLineDemo 1.0 2.0 3.0 4.0
java CommandLineDemo 123.45 Java 2020
```

The slide includes a video inset of a man speaking and a footer for NPTEL Online Certification Courses at IIT Kharagpur.

Example 57.16: Numbers as command line arguments

```
class CommandLineDemo {
    public static void main(String args[]) throws IOException {
        System.out.println("Number of arguments: " + args.length);
        int sum = 0;
        if(args.length == 0)
            return;
        for(int i=0; i<args.length; i++)
            sum + Integer.parseInt(args[i]);
        System.out.println("Sum = " + sum);
    }
}
```

Hint: Run the program with following:

```
java CommandLineDemo
java CommandLineDemo 1 2 3 4 X
java CommandLineDemo 1.0 2.0 3.0 4.0
java CommandLineDemo 123.45 Java 2020 X X
```

The slide includes a video inset of a man speaking and a footer for NPTEL Online Certification Courses at IIT Kharagpur.

Now, here is another example. This example shows the common line argument again and as we see the number of arguments it is there and here, this is the important thing that you can check it about. So, here the, whatever the input that you pass to this program through these command line arguments as an input, it basically parses it.

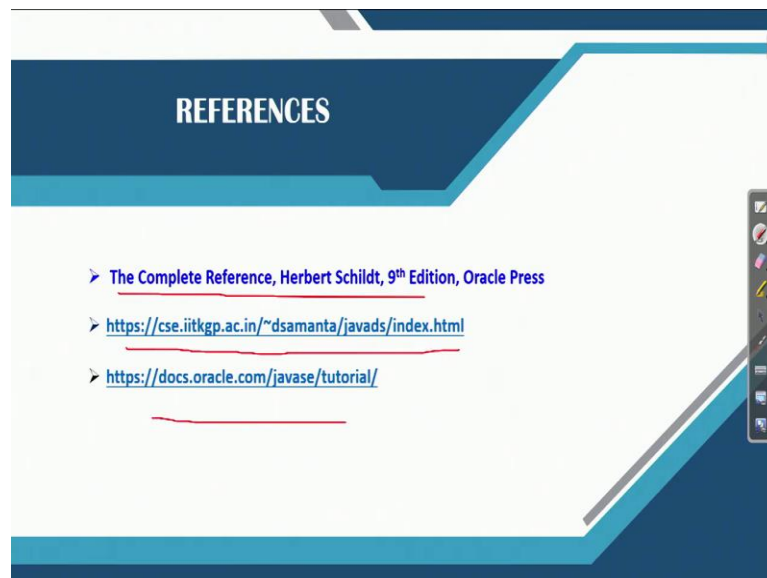
Now in this case as we are parsing integer, this means each always expect that the arguments should be integers. Now, for this, this program will run successfully, it will basically print the sum up 1 plus 2 plus 3 plus 4. So, if you run this program with this input, it will run correctly,

but if you run the same program with this input, it is error because here integer does parsing throw an exception as this is not an integer.

Similarly, this is also throws an exception because this is not an integer. Now, if you run this program with 123, it will pass but it will stuck at here because it is not a string. But here if we give the 245 and 2 to 0, then it will run successfully. But if anyone argument is not compatible with the parsing type, they need basically through an accept exception.

Now again this program you can modify by writing this program by float dot parse float and this is argument that you can parse here and then you can run that this will not pass but this program will pass this will not run because it passed although this one, but this will not pass. So, there is an error so that you can take and then you can run the program. So you can test it, the common line arguments should be compatible with the parsing of this type that you are calling for otherwise, there is a type a type mismatch and type mismatch means it is a exception error, IO exception.

(Refer Slide Time: 26:35)



Now there are many more discussion that you can find in this book and this is a document where the supplementary material you can get it and this is the Oracle official website for further studies. Thank you very much. Thank you for your attention.