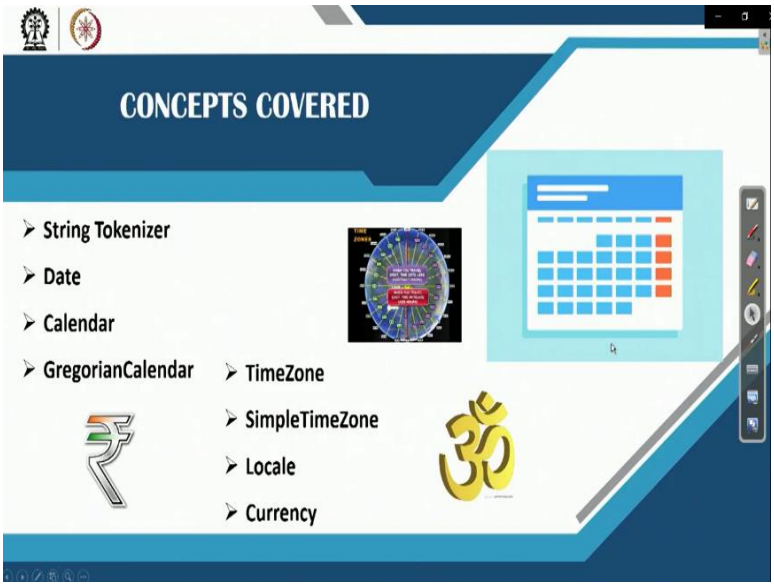
**Data Structures and Algorithms Using Java**
**Professor Debasis Samanta**
**Department of Computer Science and Engineering**
**Indian Institute of Technology, Kharagpur**
**Lecture - 59**
**Miscellaneous Utilities**

(Refer Slide Time: 00:45)


So in data structure, data is very important and as we have witnessed, the data can be of different form. Now, so we have already checked about data in view of collection, where the data can be stored, data can be retrieved in a fastest way.

But there are certain unique data type that is required, which is basically from the iteration point of view. Many utilities program which required to be discussed. Now, in this lecture, we will study about the miscellaneous utilities and handling data in different form.

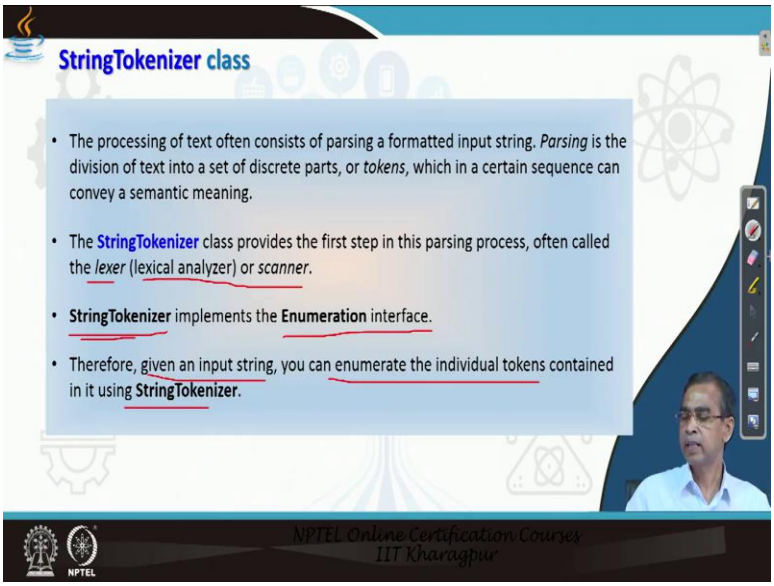
(Refer Slide Time: 01:15)




Now, the different form of data, which are very important, those are basically called date data, calendar data. And calendar can be of Gregorian calendar or any other calendar, then calculation of time zone, and then locale and currency and so many things are there.

And before going to discussion about this miscellaneous utility, one important another class is there; it is called the tokenizer. This tokenizer is very important when you process language or in the natural language processing activities. So I will start with first-string tokenizer, then I will discuss about most advanced data from the utilization point of view.

(Refer Slide Time: 02:00)

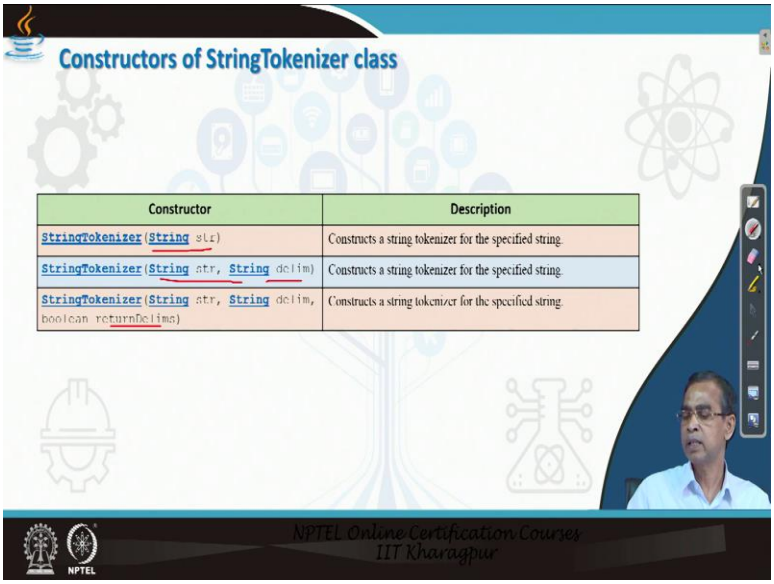

Now, let us first discuss about string tokenizer, and it is basically one utility. As you know, string tokenizer basically the idea is that given a string, it basically return the token. What is that token? A token is basically is a subpart of the string or other we can say substring of a string. But all these tokens are separated from another token with certain, what is called the tokenizer concept.

So here, actually, this string tokenizer can provide parsing of a large string. And these are basically called, are useful for lexical analysis where the system programs needs to be developed. Or parser, lexical analysis, also called parser or scanner or sometimes it is also called lexer.

Now, there is a class. The name of the class is called StingTokenizer, which is define in java dot util, which basically this class is implements the enumeration interface. Enumeration is an interface, basically it store the enumerated type data.

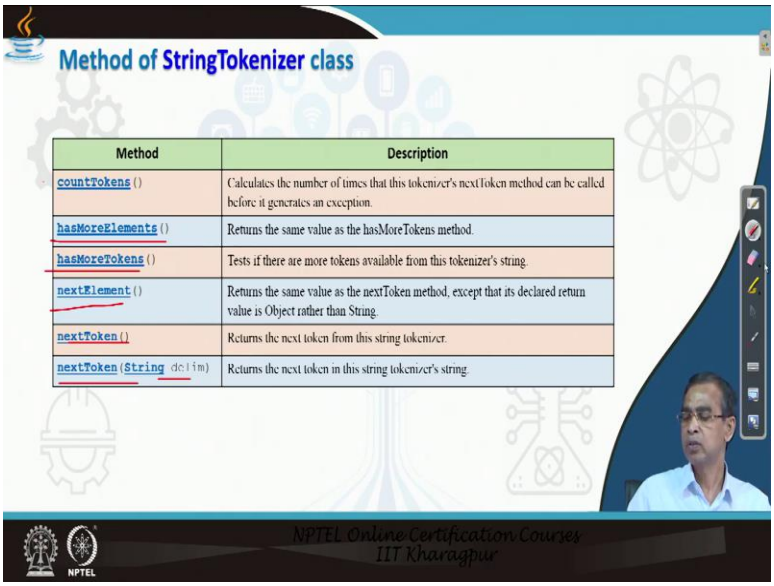
So here, given an input string, you can enumerate the individual tokens, which content in an input string using this string tokenizer. Now, let us consider the illustration of this with an example so that you can understand better.

(Refer Slide Time: 03:25)



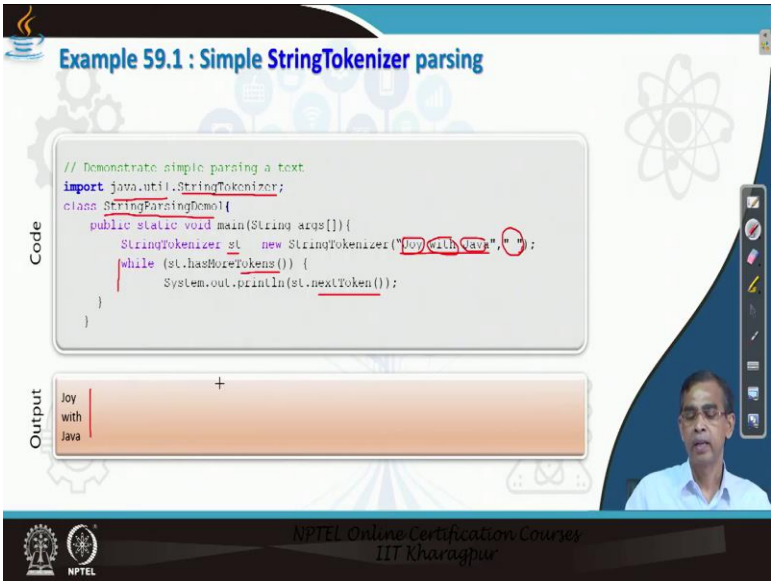

The constructor are there in order to create the object of type string tokenizer. I have given here, if you pass an input string, then string tokenizer object will be created. Here the string with string delimiter that mean up to which string that you want to create. And here, is basically same thing as the second string. But here, return delimiter; whether delimiter will be included or not. So there the perspective, three different perspective the string object will be created.

(Refer Slide Time: 03:55)




Now, let us have these are different methods which are defined in string tokenizer class, namely count tokens, whether it has more elements or not, it return Boolean; has more tokens or not, next element, next token, and next token with delimiter. So these are the different methods those are, defining the class StringTokenizer, which can be used to manipulate this method.

(Refer Slide Time: 04:18)

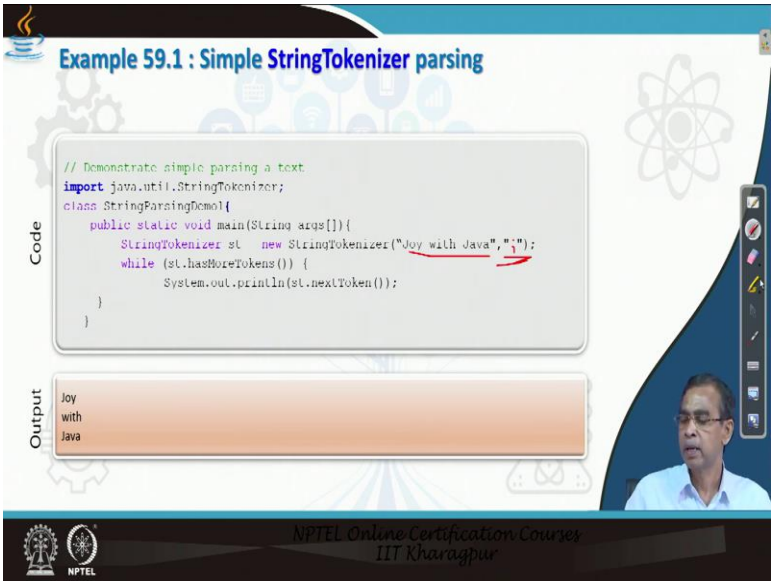


Now, let us have an example so that you can find its utility. And this is one program which we want to give a demo. It is called the string parsing demo. And as this string tokenizer is defined in util, so we have to import this method there, means, class there.

Now, here we create a string tokenizer object for this string as what is called the input. And here, is a delimiter that what is a delimiter means that, which basically will distinguish from one string to another string.

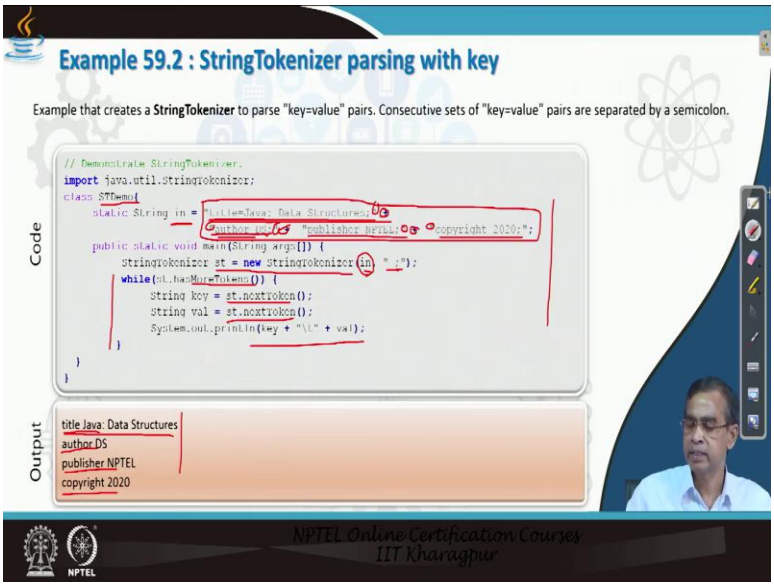
That means in this string as the delimiter this one. So this is the one token, this is another token. And this is another token. Now, here you just took a parsing the string returning the token and then printing the token, and this will give this one.

(Refer Slide Time: 05:15)




Now, this program, if you modify and with say new delimiter defining say, here. And then it will print the whole thing because it does not have these delimiter. So it will print this one. So delimiter indicates that okay, how you can parse string and which is basically identification or marking that this basically starts a new token and from the previous token. So this is a concept. Now, let us first discuss on more examples so that we can understand this concept better.

(Refer Slide Time: 05:46)




And here is the next illustration that you can think about. Here we are giving a new another delimiter. These are second demo we can say, so demo two. And here, we create a new object and this is basically a string we can. This is the input string we can think about it.

And in this input string, as we can see, the delimiter is particularly defined. Here, this and this together form the delimiter. How we can do it? We can mention this one. And this is a input string. And this is basically the string tokenizer that we have created. This is the input string and this is a delimiter. That means here, the token in between the two, what is called the elements, I will be considered the token.

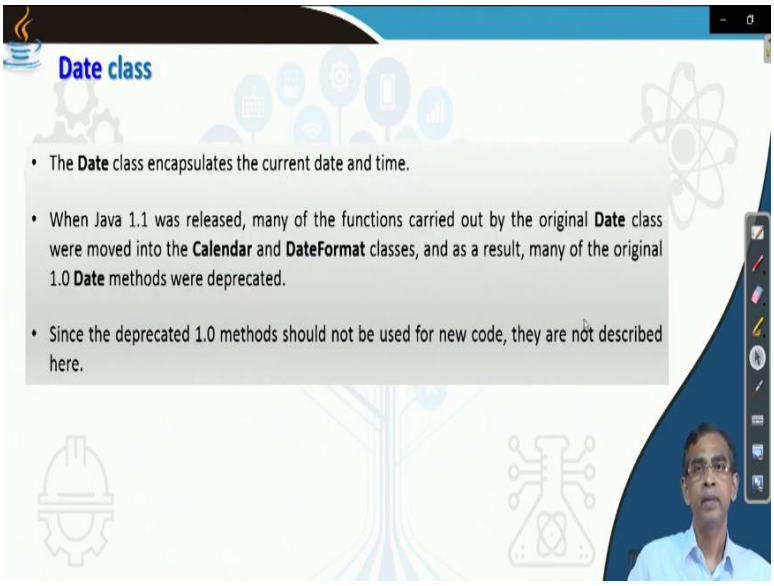
So, as you see, title here, for example, if we parse this string as a writing this, this is a basically parser here. We see the check that weather token exist or not. And then it basically next token, and then return the next token, and then print the token value. So here, this basically, this is the first print that it basically gives you.

Here this is basically as we see title. And for this string, this is basically the token that it return and for this string, this is basically the another token as we can see because this is the token within this delimiter at it is possible. And this is another token that we can see. And this is a another token, as you can get it. So here, basically, all these are the string and we apply the tokenizer this one.

Now, if you remove all these things and then create another completely one sting like, so this if you and if you remove it, you can check the program, and then you can write it and then again that way. And then if you do it, it will basically give you the same program like but here the token is you this one only. But for this spot only, it is showing the result. But if you combine both the things you can check.

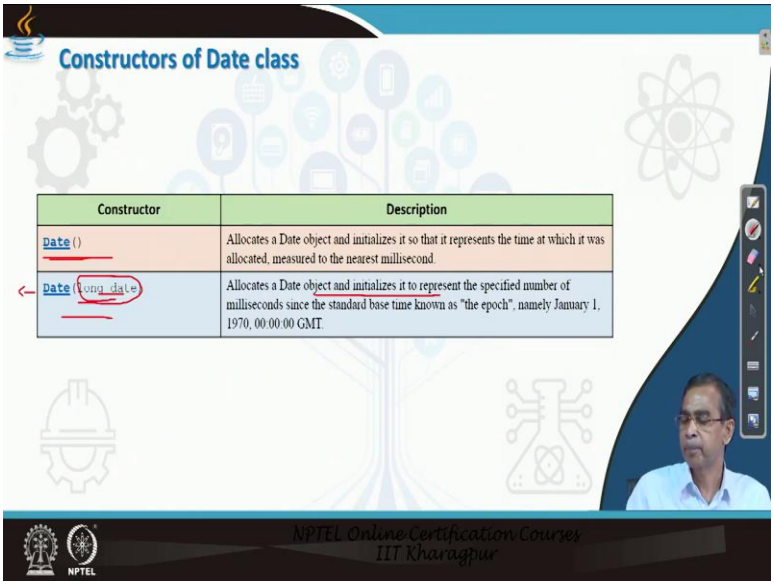
So you can see, removing only or making only one string, but giving the delimiter and you run the program, how it runs and how it produces output that you can check it. So this you can try and then you can see the difference, how the two things are working better.

(Refer Slide Time: 08:29)




Now, let us consider another utility and it is called the date. Date is very important. Now, given a date, you have to find another date; given a data, you have to find the month; given a date, you have to find a day. So many utilities are required. And in many application development, these things are obvious things and you have to apply it. So java provide, java developer that provided one utility in the form of a date class. So it has a versatile application.

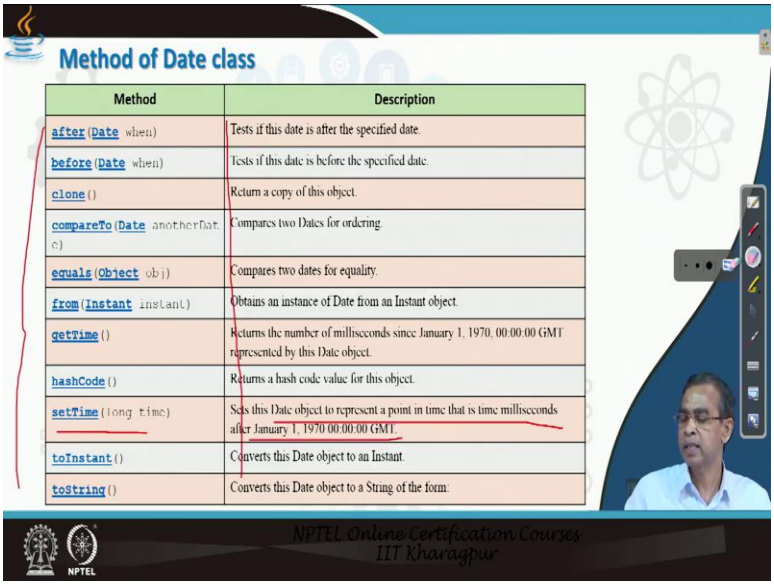
(Refer Slide Time: 09:02)



There are many methods, many operations that you can do it. And whatever the operations are there for which the method is already defined, you have to call this. I will try to give some illustration of some call of some methods for your purpose.

Anyway, date object can be created, these basically return the current date. And the date long date, it basically gives you the date object initialized to represent. So if you initialize the date object with the current date, then you can pass the current date here. And then this date is basically the current date will be created, otherwise, it will create the default date that is there, the system date, actually.

(Refer Slide Time: 09:38)

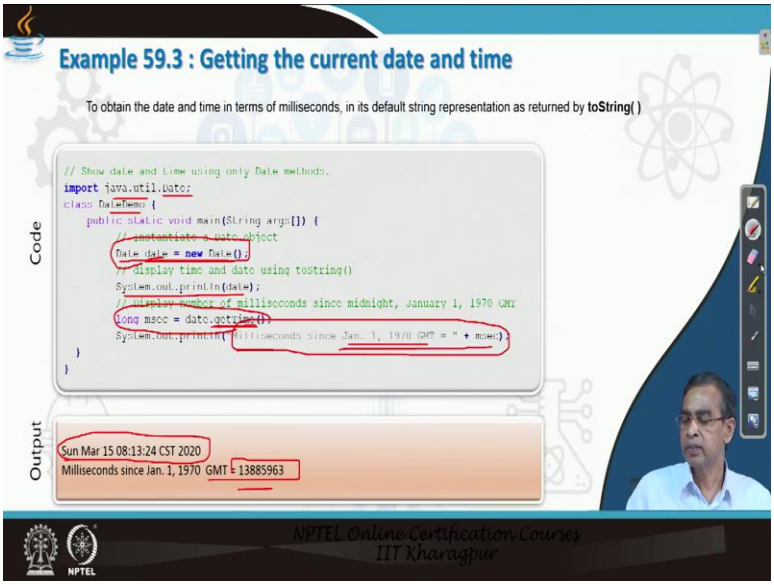



Now, these are the date and there are many methods which are required. Whether two dates are same, whether difference between the two dates, whether the month, year and date of a particular date. And then hash code of a date, and then setting a time for a date. I mean, changing the time for a date.

So many utilities that you can apply and all these are the self-explanatory, you can just follow this and better idea will be that you can write your program, and then call if method in your program and check the output how it is giving. That is a very simple way that you can learn better.

So the description of each method is given here for your understanding. And here set time, this basically to represent a point time, that in the millisecond after January 1. I will give an explanation so that you can understand this concept better.

(Refer Slide Time: 10:25)




Here is an example that you can follow. I will not be able to give much more illustration because the time is not so permissible to do that. Anyway, I will give just an idea so that you can understand about it.

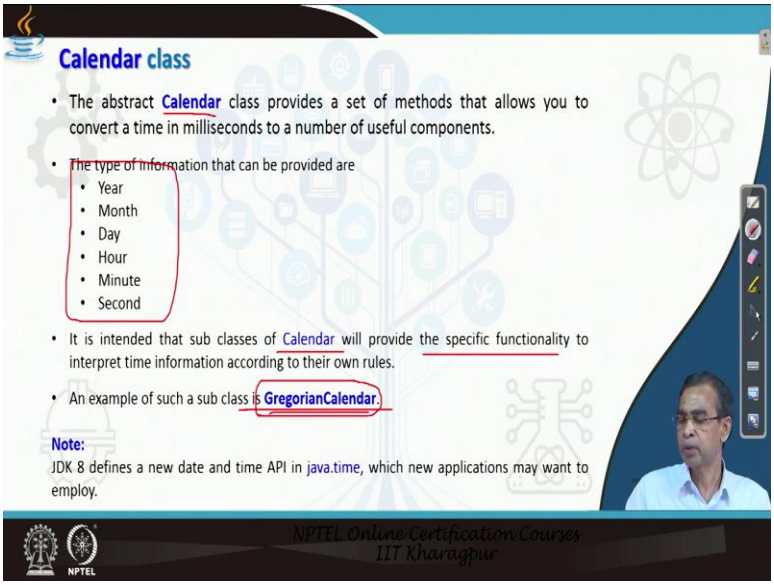
Now, let us consider this program. And we are using date class, so we have to import it and these are date demo. So we create a date, new date. So this is a default date that you create is basically the system date that will basically create it.

And then if you print it, and you can print that this is the date it is created. So this basically, sorry, this is basically give you the current date, current date means system date, yes. So what are the system date? It basically store it this one.

Now, get time; now, this basically, it tell that what is the time that we have elapsed. Now, this get time method if we call this, then it will basically calculate the current time. That is basically the default time that is stored there in a java field. So this is basically the time. It will basically calculate the time elapsed from this date to till time.

So this basically it says that this is the time that elapsed from that date to this one. So this is the idea that you can call, and then similarly you can extend this program, calling the different method and practice it, and you will be able to understand how the different method work.

(Refer Slide Time: 11:52)

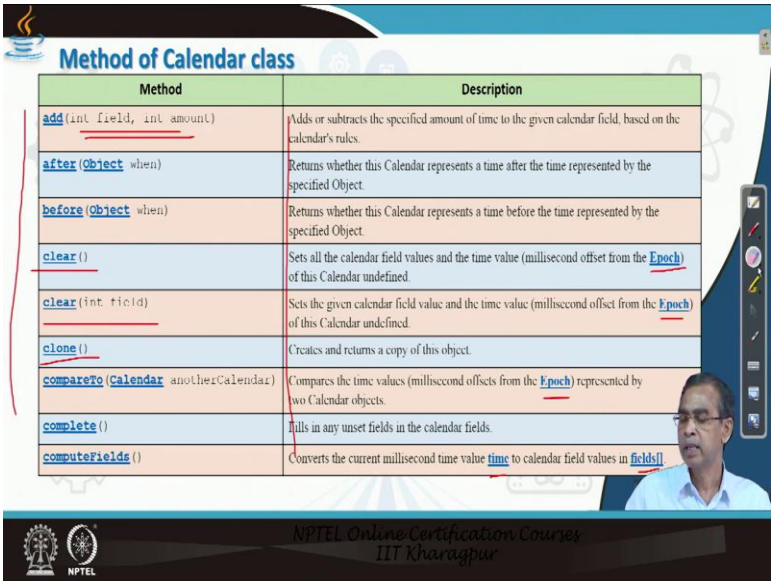




Now, let us come to the calendar. Calendar is another important one utility, which you can consider to develop many application. So you can see the calendar is also another important data. Now, the calendar can provide many information which we have mentioned, including time-related year, month, days; everything is there. And so the calendar class has been planned so that

you can maintain one calendar in your program and you can have certain support related to the calendar activities.

Now, so there is basically the popular this is a normal calendar we usually follow, it is called the Gregorian calendar. There are many other calendar also can the java developer can support. Even a programmer can define their own calendar also depending on, for example, you can maintain your own calendar according to your own rituals or own locales that also you can do it. So these are the things it is possible. And now, let us have some examples so that how the calendar object can be created.

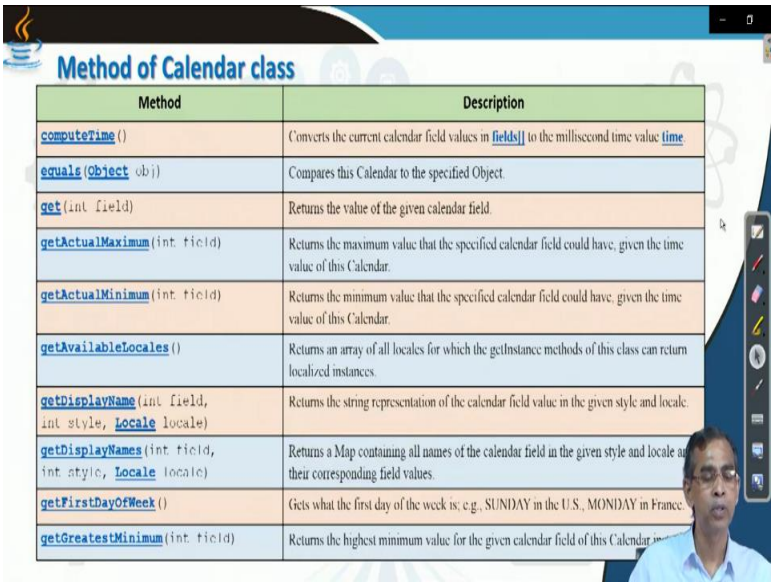
(Refer Slide Time: 12:56)



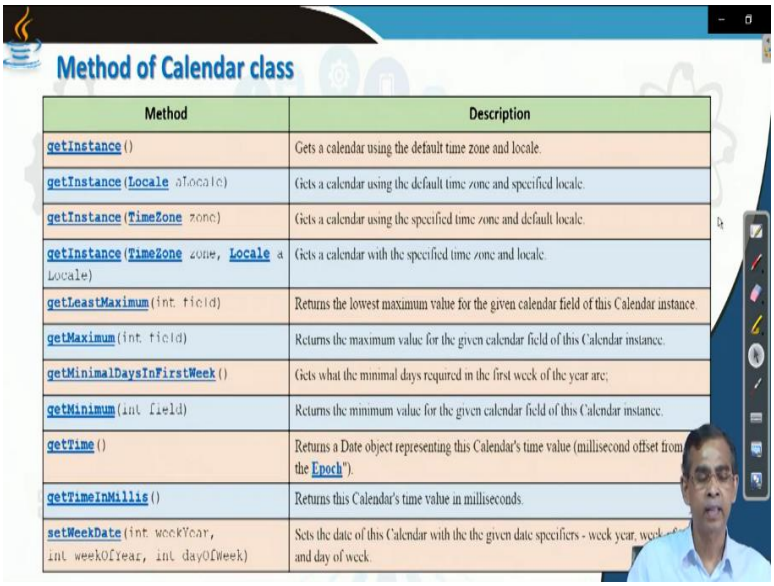

Now, there are many methods are declared in the calendar class, which I have declared here. These basically add, we can add some time or date or month into a particular calendar or a calendar can be reinitialize also. And also, you can clear, remove a certain values in the calendar. So it basically delete a particular calendar. A copy, calendar can be copied to another calendar also.

So there are many methods which you can read all the description and you can understand. And then there are few terminology that you can consider. Epoch will be discussed when we will consider time zone then you will be able to understand about it.

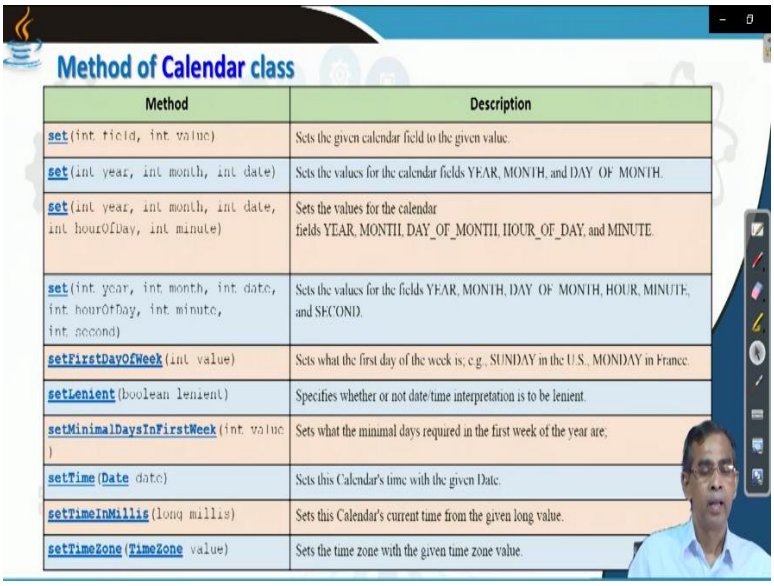
(Refer Slide Time: 13:39)




| Method | Description |
|---|---|
| computeTime() | Converts the current calendar field values in fields[] to the millisecond time value time. |
| equals(Object obj) | Compares this Calendar to the specified Object. |
| get(int field) | Returns the value of the given calendar field. |
| getActualMaximum(int field) | Returns the maximum value that the specified calendar field could have, given the time value of this Calendar. |
| getActualMinimum(int field) | Returns the minimum value that the specified calendar field could have, given the time value of this Calendar. |
| getAvailableLocales() | Returns an array of all locales for which the getInstance methods of this class can return localized instances. |
| getDisplayName(int field, int style, Locale locale) | Returns the string representation of the calendar field value in the given style and locale. |
| getDisplayNames(int field, int style, Locale locale) | Returns a Map containing all names of the calendar field in the given style and locale a their corresponding field values. |
| getFirstDayOfWeek() | Gets what the first day of the week is; e.g., SUNDAY in the U.S., MONDAY in France. |
| getGreatestMinimum(int field) | Returns the highest minimum value for the given calendar field of this Calendar in |

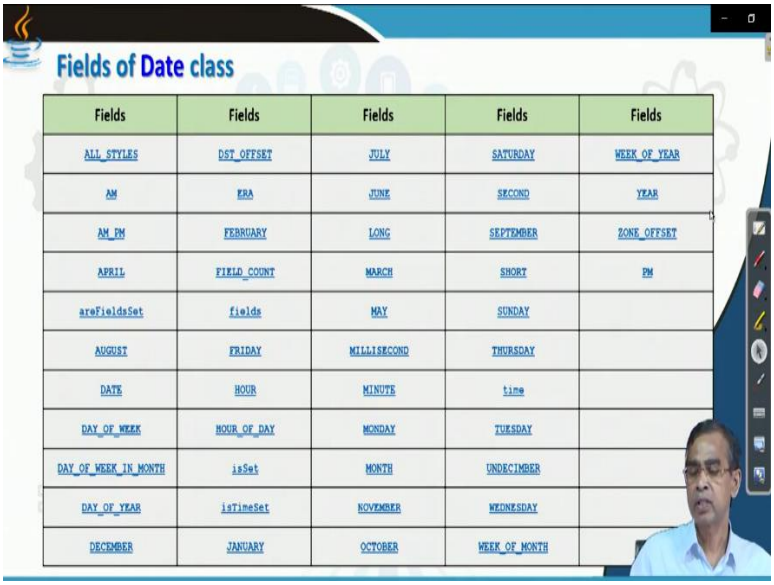



| Method | Description |
|---|---|
| getInstance() | Gets a calendar using the default time zone and locale. |
| getInstance(Locale aLocale) | Gets a calendar using the default time zone and specified locale. |
| getInstance(TimeZone zone) | Gets a calendar using the specified time zone and default locale. |
| getInstance(TimeZone zone, Locale a Locale) | Gets a calendar with the specified time zone and locale. |
| getLeastMaximum(int field) | Returns the lowest maximum value for the given calendar field of this Calendar instance. |
| getMaximum(int field) | Returns the maximum value for the given calendar field of this Calendar instance. |
| getMinimalDaysInFirstWeek() | Gets what the minimal days required in the first week of the year are; |
| getMinimum(int field) | Returns the minimum value for the given calendar field of this Calendar instance. |
| getTime() | Returns a Date object representing this Calendar's time value (millisecond offset from the Epoch"). |
| getTimeInMillis() | Returns this Calendar's time value in milliseconds. |
| setWeekDate(int weekYear, int weekOfYear, int dayOfWeek) | Sets the date of this Calendar with the the given date specifiers - week year, week and day of week. |



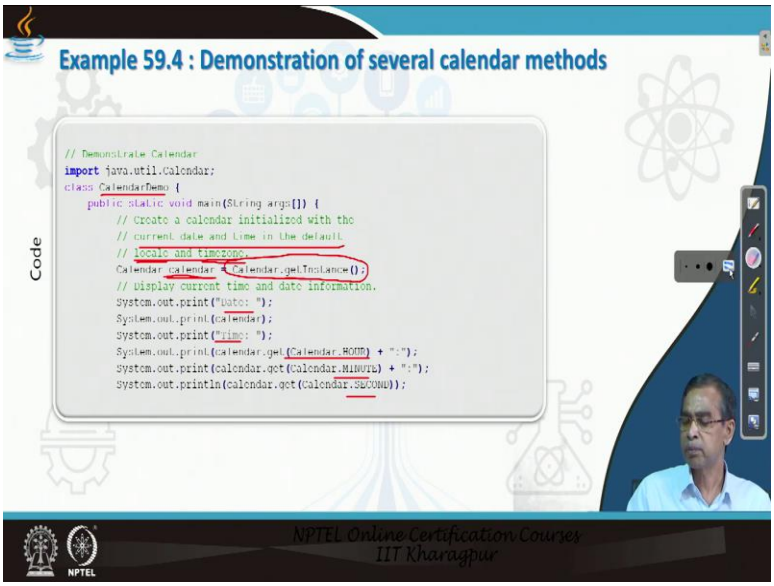

Now, let us see some, these are few more methods. There are a huge number of methods to deal with calendars so that you can process any sort of calendars that have it. As we have listed many calendars methods related to it.

(Refer Slide Time: 13:54)




These are the different fields also defined in the calendar. So it is a very vast concept actually. It takes some time to understand everything in details.

(Refer Slide Time: 14:02)

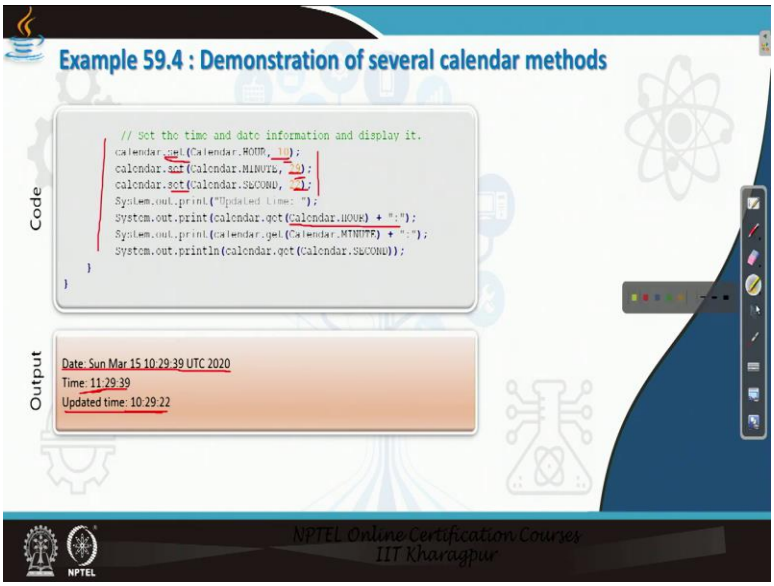



Now, here is the different, this is an example that you can see. It is very simple example so that you can understand how the demo of a calendar can be. And this is the calendar demo. Now, first, we will create a calendar and this calendar will be initialized with the current date and time in the default locale and time zone.

Now here, default locale means, it is a standard US locale and the time zone is basically standard time that is your system is running according to your current settings. Now, this basically you see how we can create a calendar object. So calendar and you can note that calendar does not have its own any constructor. So we have to use the factory method those are define those can be used there.

So as a get instance for a calendar object is created, and once this calendar is created then we will be able to print the date, time, here is a month, hour, and seconds. So this is okay if we continue this program, there are few more statements that you can think about.

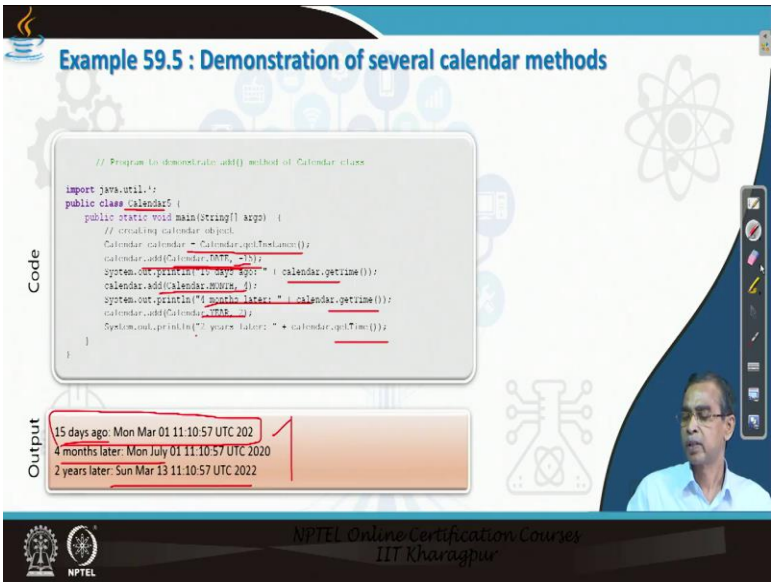
(Refer Slide Time: 15:11)




So this is in continuation of previous one. The setting also can be done; set. That means we can change the hour by 10, 29, 22. Now, the output that you can get, so this is the earlier calendar that we have created. This basically is the calendar date that can be created; time also, it is basically the previous one. And updated time, after setting time, as you see, this is 11, and then 10, 29 and 22, we have set it. So we can change the setting of the calendar.

So this way you can set the calendar both day, time, month, everything that is related to this calendar you can do. That method is basically get, set, all these methods by which you can have the new calendar.

(Refer Slide Time: 16:01)

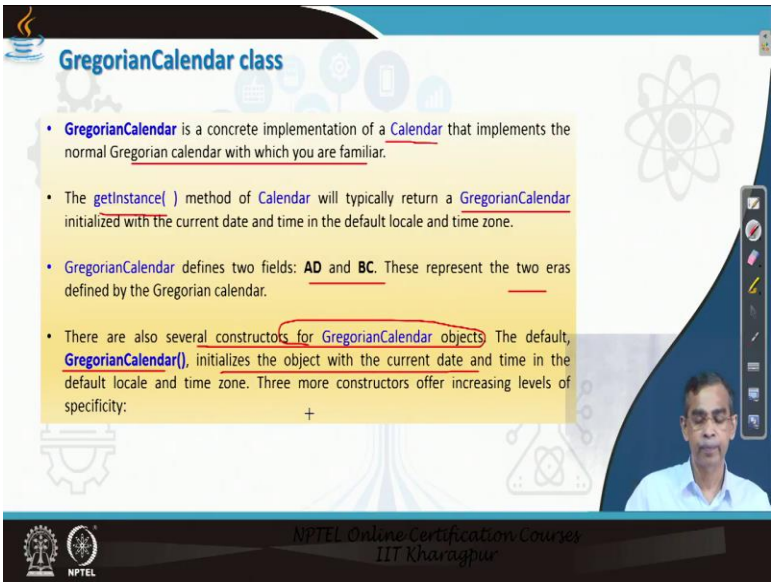



So this is a simple example of calendar. There is another example here. We can see how we can create your own calendar also, that is the one idea you can do, or you can set a calendar also. So this is the one class, demo class we can say. We create a calendar object get instance and then we change the setting of the calendar. So it will basically add. We are adding and get time, get time, get time.

So this actually, this is basically the time. This basically, if you set it minus 15 means, is basically say that 15 days ago and it basically give the value it is there. And here again, 4 months later. So it basically set by 4. So it is a 4 months later. If you year to set, then 2 years later.

So this way you can reset the calendar from the current calendar or any instance of calendar to any other calendar. So in many business applications, it is required to reset the calendar and accordingly you use it accordingly to do certain calculation that can be useful for that.

(Refer Slide Time: 17:12)

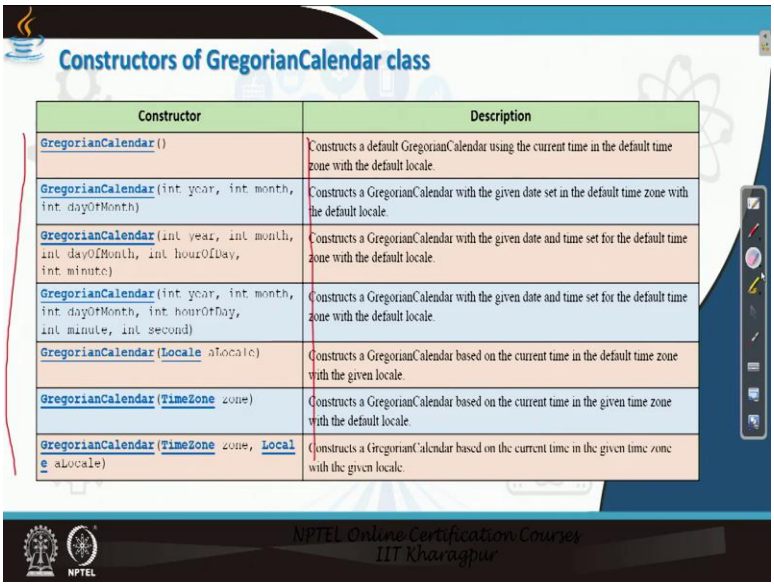

Now, let us see some Gregorian calendar utility. It has the most versatile calendar and usually the more common calendars. So it is basically, it implements the calendar. It is a one special type of calendar object we can say. And it is more familiar as we use it, we are accustomed to this calendar.

Here the get instance method will typically return the Gregorian calendar that we have already studied about it. And it also define two fields AD and BC to represent two different time versions actually. And there are many constructors defined in Gregorian calendar objects, which we can

create to create your own object. And there is a default constructor also, which basically initialize object with the current date and time.

Now, let us see some examples so that we can understand this concept better.

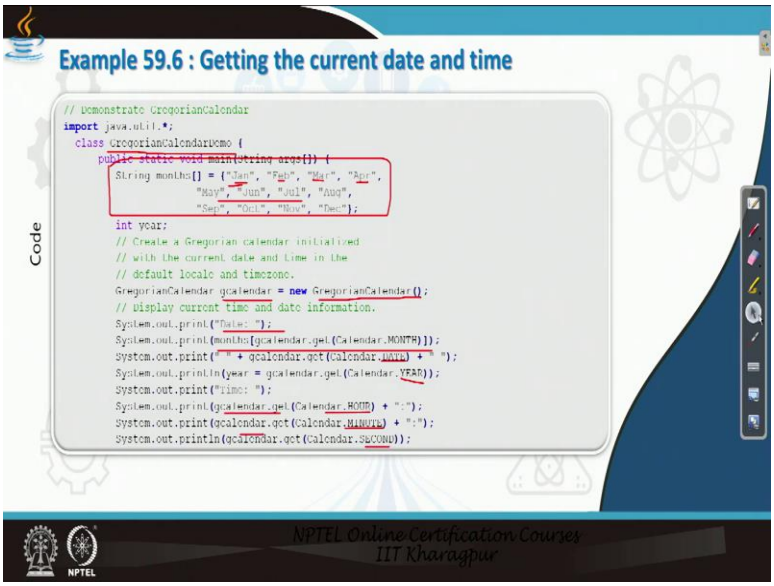
(Refer Slide Time: 18:06)




| Constructor | Description |
|---|---|
| GregorianCalendar() | Constructs a default GregorianCalendar using the current time in the default time zone with the default locale. |
| GregorianCalendar(int year, int month, int dayOfMonth) | Constructs a GregorianCalendar with the given date set in the default time zone with the default locale. |
| GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute) | Constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale. |
| GregorianCalendar(int year, int month, int dayOfMonth, int hourOfDay, int minute, int second) | Constructs a GregorianCalendar with the given date and time set for the default time zone with the default locale. |
| GregorianCalendar(Locale aLocale) | Constructs a GregorianCalendar based on the current time in the default time zone with the given locale. |
| GregorianCalendar(TimeZone zone) | Constructs a GregorianCalendar based on the current time in the given time zone with the default locale. |
| GregorianCalendar(TimeZone zone, Locale aLocale) | Constructs a GregorianCalendar based on the current time in the given time zone with the given locale. |

And here is a demo. These are the different methods which are defined for the Gregorian calendar. These are constructors, the different constructor, how we can create the different objects of this type is mentioned here?

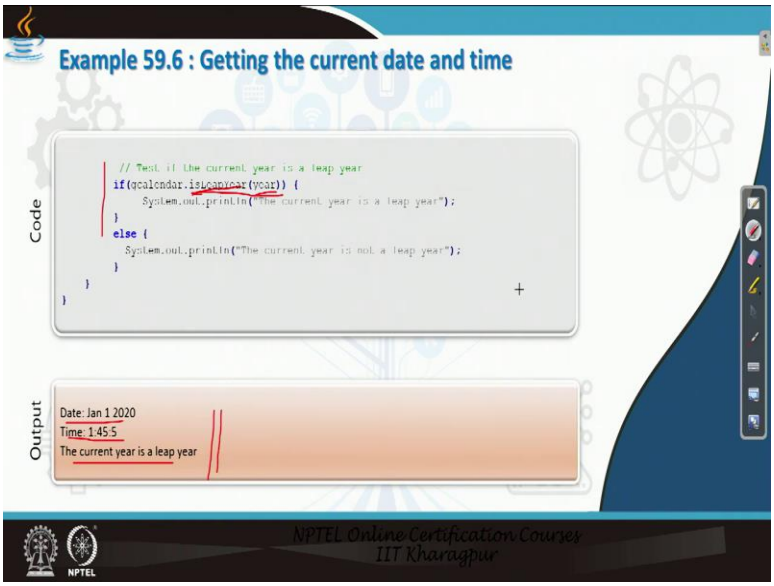
(Refer Slide Time: 18:29)




And this is an example that you can think about; the demo of Gregorian calendar. And here, we just initialize the month of a calendar by this string, we can say. Otherwise, different other symbols also we can use to represent the different month also you can possible.

And here you see, we create a Gregorian calendar objects by calling this is the default constructors. Now, the date of the Gregorian calendar can be printed using this format. You can see the month, date, and year; so this way, the date can be created, printed. Then time also can be printed hour, minutes, and seconds. These are the form of a string we can print. Now, here is the output that you can get it for this program.

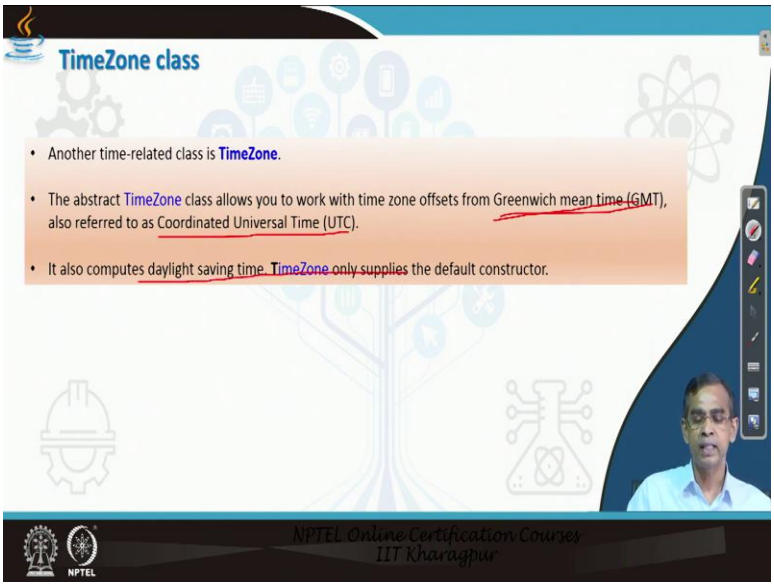
(Refer Slide Time: 19:21)




Writing some print statement here, the output is basically as you see this is the date, this is the time, the current year is leap year or not. This is the one method that is declared in Gregorian calendar class. And then the method is leap year to decide whether particular year is leap year or not, it automatically check it.

So these are the different utility, there are many utilities are defined there and which you can practice and exercise and then you can check how they works for you. So that is basically I am giving highlights about the different class. But for the details of this class declaration, you can have the details practice then you can find it.

(Refer Slide Time: 20:00)

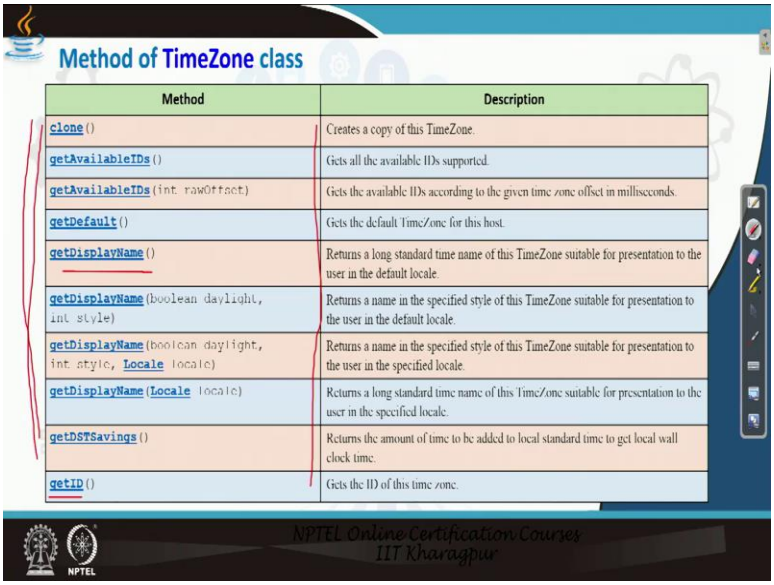

And now, time zone is one important aspect. So for the utilities concerned, as you know, our globe is divided into 24 equal parts. And this is basically one longitudinal starting from the Greenwich for which the zero-hours started. And every 124th of the part is basically one hour difference. And this way the time zone can be calculated.

We are in India having some time zone. There are different time zone have their unique name, and everything is defined in the class time zone so that you can understand and accordingly, you can set time, change the time, depending on this thing. Probably, in your mobile phone, you can

set time zone or reset some time zone; or you can, you are in a particular time zone, if you want to know the time of other time zone that all those utilities is developed. So those things, you can do it.

And time zone is basically follows the concept, it is called Greenwich Mean Time, that is the zero meantime, you can say for which there. And then, all other time which with respect to this time actually will be created. And then they are basically expressed UTC. UTC is called the coordinated universal time. Now, so these are the idea that you can use it. Now, let us see the different constructor and methods which are defined there.

(Refer Slide Time: 21:26)




| Method | Description |
|---|---|
| clone() | Creates a copy of this TimeZone. |
| getAvailableIDs() | Gets all the available IDs supported. |
| getAvailableIDs(int rawOffset) | Gets the available IDs according to the given time zone offset in milliseconds. |
| getDefault() | Gets the default TimeZone for this host. |
| getDisplayName() | Returns a long standard time name of this TimeZone suitable for presentation to the user in the default locale. |
| getDisplayName(boolean daylight, int style) | Returns a name in the specified style of this TimeZone suitable for presentation to the user in the default locale. |
| getDisplayName(boolean daylight, int style, Locale locale) | Returns a name in the specified style of this TimeZone suitable for presentation to the user in the specified locale. |
| getDisplayName(Locale locale) | Returns a long standard time name of this TimeZone suitable for presentation to the user in the specified locale. |
| getDSTSavings() | Returns the amount of time to be added to local standard time to get local wall clock time. |
| getID() | Gets the ID of this time zone. |

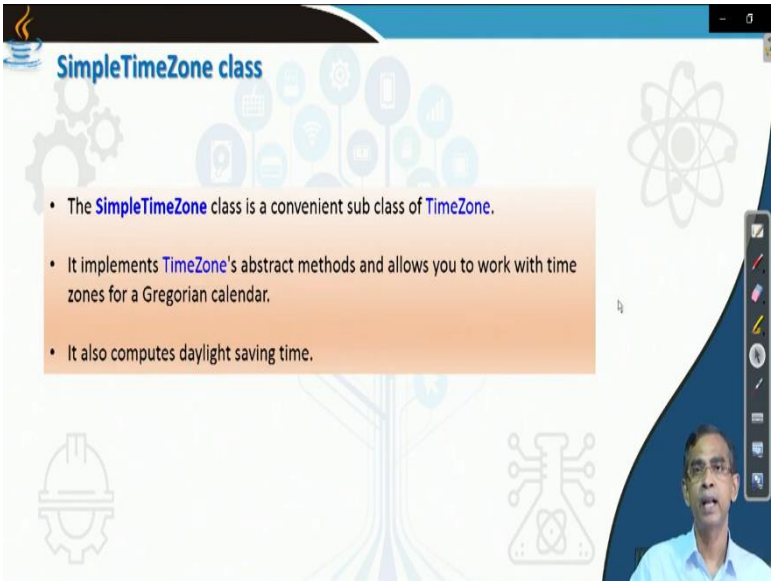
So in the time zone class, several methods are defined, which I have listed here. And the discussion of the different methods is given there. And it basically tell that how the time zone object can be manipulated by the different method.

For example, say get ID method, it basically gives the ID of a time zone. Every time zone has its own ID that is defined in the time zone class. It can return it and this can be utilized to have certain inference in your program actually. So the different get display name is basically a name of a time zone, every time zone has its own name and everything.

So you can set your time zone also. If you are from a current time zone, if you want to move into other time zone that, also you can. So this is related to time zone you can do many other

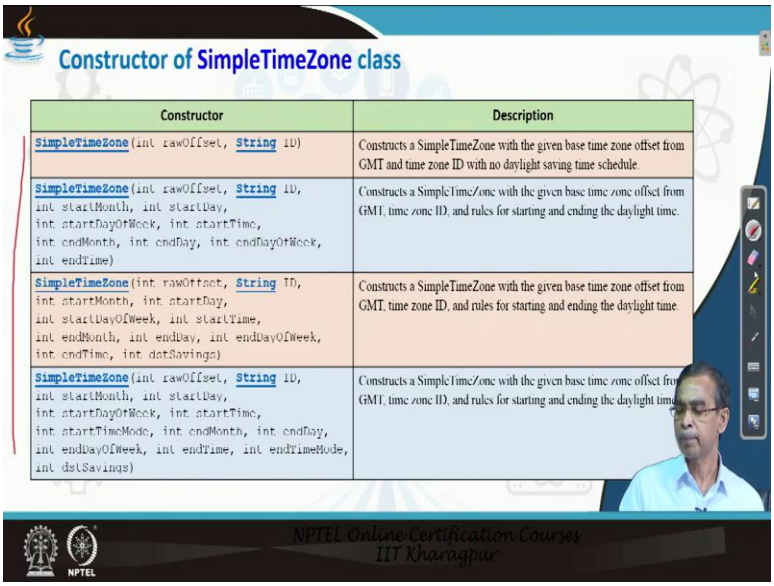
manipulations that is mentioned by that is that can be done by different methods, which is listed here.

(Refer Slide Time: 22:22)


Now, I will just discuss about one simple time zone utility that is interesting to learn.

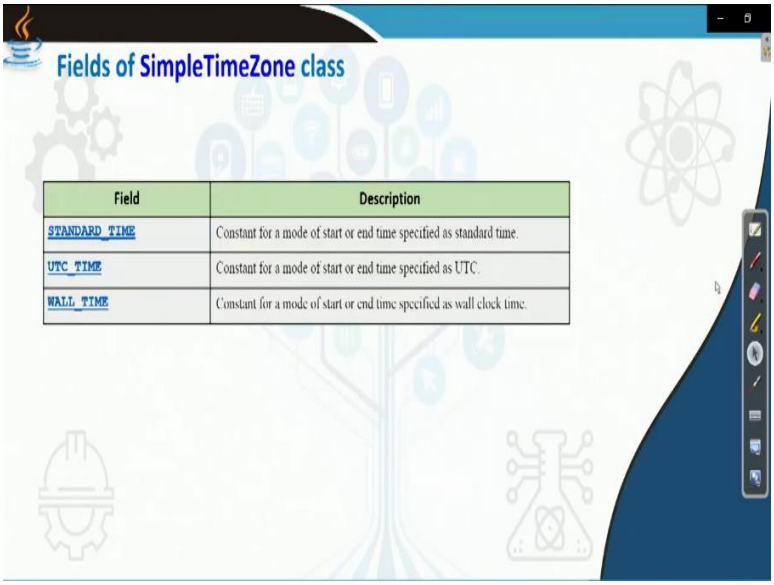
(Refer Slide Time: 22:27)




So there are many methods also defined there. These are the constructor, which is there. With this constructor, you will be able to create the time zone. They are called simple time zone. The time zone is basically more utilised, this one and it has many other ideas about.

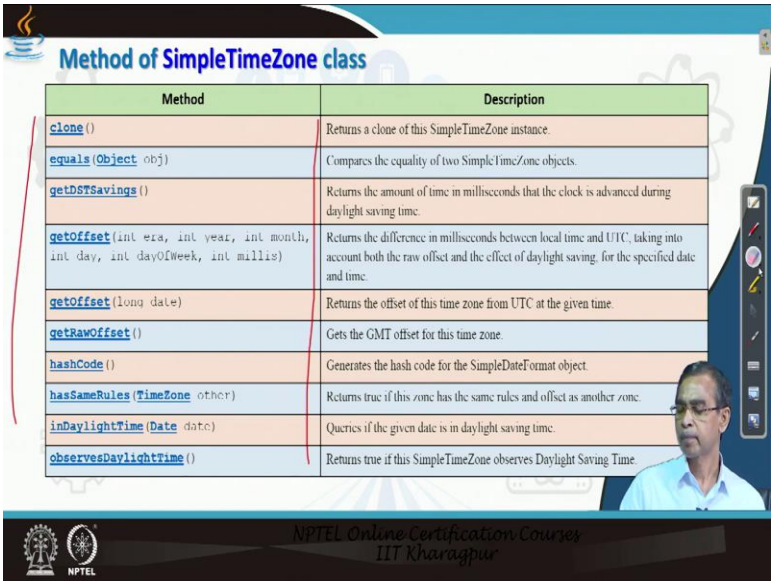
So that with respect to a current time zone, if you want to add it or you can, to move into the sometime zone, you can differ it all those things, you will be able to do that. Now, there are many constructor, these are constructors.

(Refer Slide Time: 23:00)

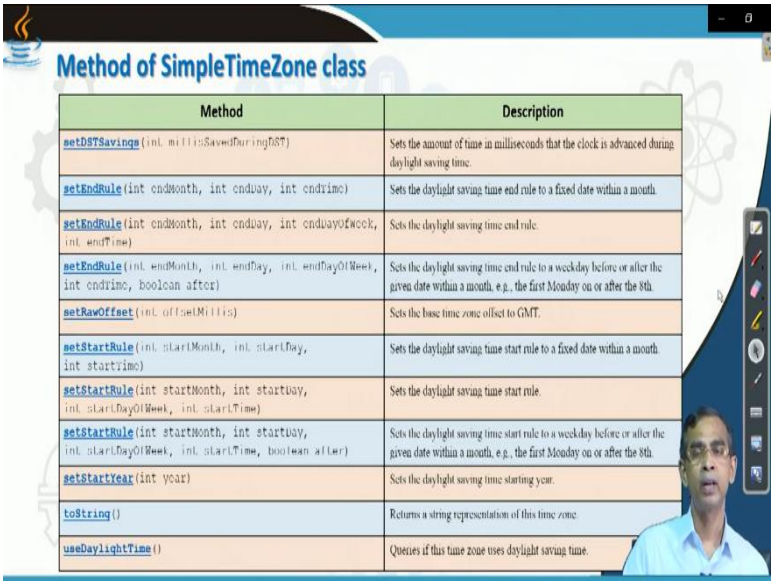


So there are many methods also defined. These are the few fields also defined there.

(Refer Slide Time: 23:04)



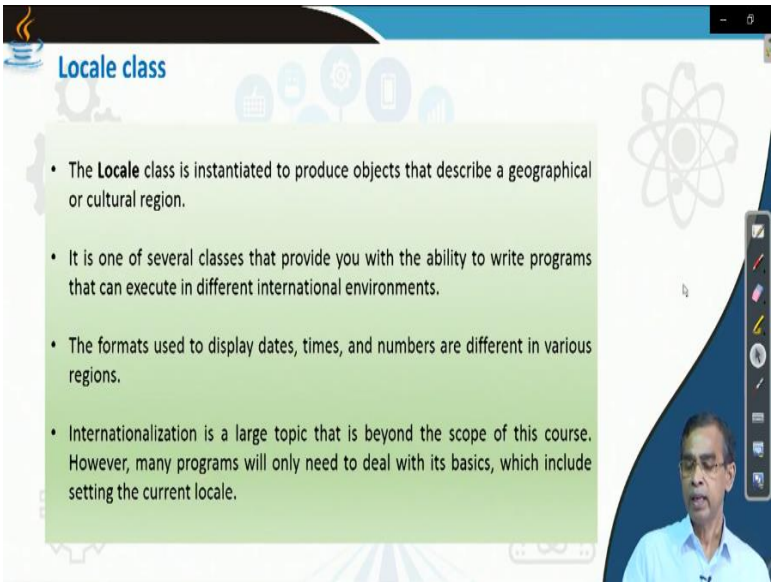

| Method | Description |
|---|---|
| clone() | Returns a clone of this SimpleTimeZone instance. |
| equals(Object obj) | Compares the equality of two SimpleTimeZone objects. |
| getDSTSavings() | Returns the amount of time in milliseconds that the clock is advanced during daylight saving time. |
| getOffset(int era, int year, int month, int day, int dayOfWeek, int millis) | Returns the difference in milliseconds between local time and UTC, taking into account both the raw offset and the effect of daylight saving, for the specified date and time. |
| getOffset(long date) | Returns the offset of this time zone from UTC at the given time. |
| getRawOffset() | Gets the GMT offset for this time zone. |
| hashCode() | Generates the hash code for the SimpleDateFormat object. |
| hasSameRules(TimeZone other) | Returns true if this zone has the same rules and offset as another zone. |
| inDaylightTime(Date date) | Queries if the given date is in daylight saving time. |
| observesDaylightTime() | Returns true if this SimpleTimeZone observes Daylight Saving Time. |




| Method | Description |
|---|---|
| setDSTSavings(int millisSavedDuringDST) | Sets the amount of time in milliseconds that the clock is advanced during daylight saving time. |
| setEndRule(int endMonth, int endDay, int endTime) | Sets the daylight saving time end rule to a fixed date within a month. |
| setEndRule(int endMonth, int endDay, int endDayOfWeek, int endTime) | Sets the daylight saving time end rule. |
| setEndRule(int endMonth, int endDay, int endDayOfWeek, int endTime, boolean after) | Sets the daylight saving time end rule to a weekday before or after the given date within a month, e.g., the first Monday on or after the 8th. |
| setRawOffset(int offsetMillis) | Sets the base time zone offset to GMT. |
| setStartRule(int startMonth, int startDay, int startTime) | Sets the daylight saving time start rule to a fixed date within a month. |
| setStartRule(int startMonth, int startDay, int startDayOfWeek, int startTime) | Sets the daylight saving time start rule. |
| setStartRule(int startMonth, int startDay, int startDayOfWeek, int startTime, boolean after) | Sets the daylight saving time start rule to a weekday before or after the given date within a month, e.g., the first Monday on or after the 8th. |
| setStartYear(int year) | Sets the daylight saving time starting year. |
| toString() | Returns a string representation of this time zone. |
| useDaylightTime() | Queries if this time zone uses daylight saving time. |

And here are many methods which you can use for the simple time zone to manipulate it. The discussion it is given there. You can check all these statement, these are the self-explanatory, easy to understand by which you can just manipulate the time zone of whatever the way that you want to do it. There are many more methods that is there.

(Refer Slide Time: 23:29)

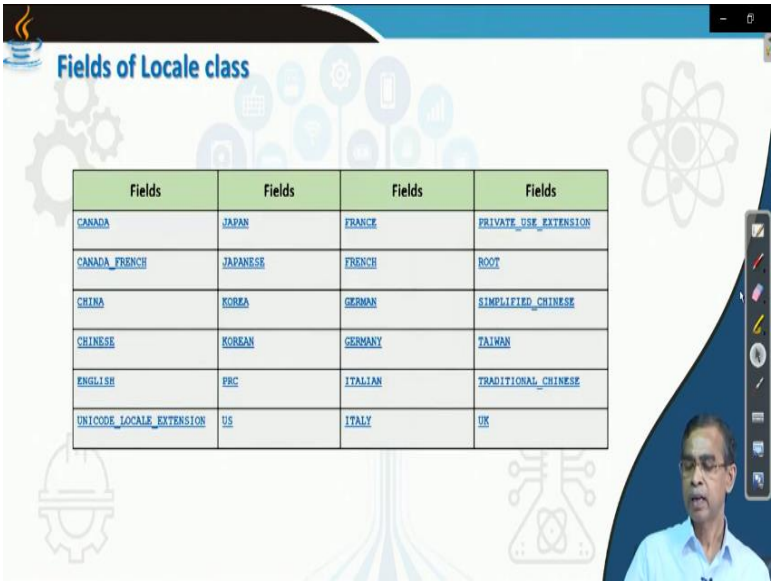




Now, let us see the locale. Locale is basically, as I told you, the java provides Unicode format, and it is one important utilization is that you can express your, I mean text according to the character set of certain locales. The usual locales that we usually follow is the US locale. But there are many other locales that is there, for example, expressing some text in Chinese, or France, or Romanian language or Hindi, everything, you can do it.

So these kind of things is possible by means of this utility class called the locale. So it basically, the idea of this locale class is to produce object that describes geographical or cultural region and

then locale also can be used to represent many other I mean, user understandable form. So it is one aspects by which java makes the programming language a little bit internationalized actually.

(Refer Slide Time: 24:38)



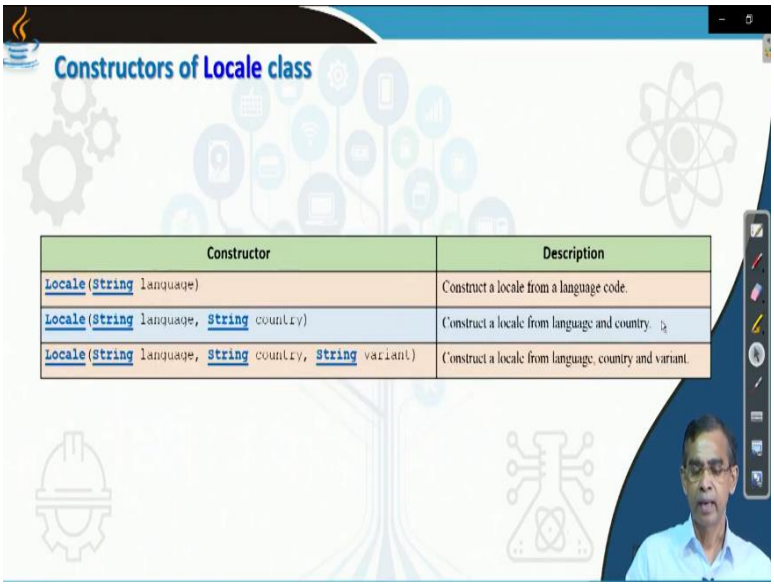

| Fields | Fields | Fields | Fields |
|---|---|---|---|
| CANADA | JAPAN | FRANCE | PRIVATE_USE_EXTENSION |
| CANADA_FRENCH | JAPANESE | FRENCH | ROOT |
| CHINA | KOREA | GERMAN | SIMPLIFIED_CHINESE |
| CHINESE | KOREAN | GERMANY | TAIWAN |
| ENGLISH | PRC | ITALIAN | TRADITIONAL_CHINESE |
| UNICODE_LOCALE_EXTENSION | US | ITALY | UK |

Now, there are many locales that it can support, as I have mentioned here. They are basically is a Canadian locale, Japanese locale, Italian locale, UK locale, US locale. This is the Republic China locale and so many other English locale. So there are many other different locales that you can.

So you can use a particular locale with which you want to print your text and then accordingly it will print. So in the print ln statement or print f statement here, the locale can be specified. For example, locale dot us, you can use it then it basically use statement locale dot PRC, for example; it will print in Chinese the text that you want to display.

Automatically, it will convert in a specification of a particular character set and then print it. So these are versatility that you can have it and you can check it writing program.

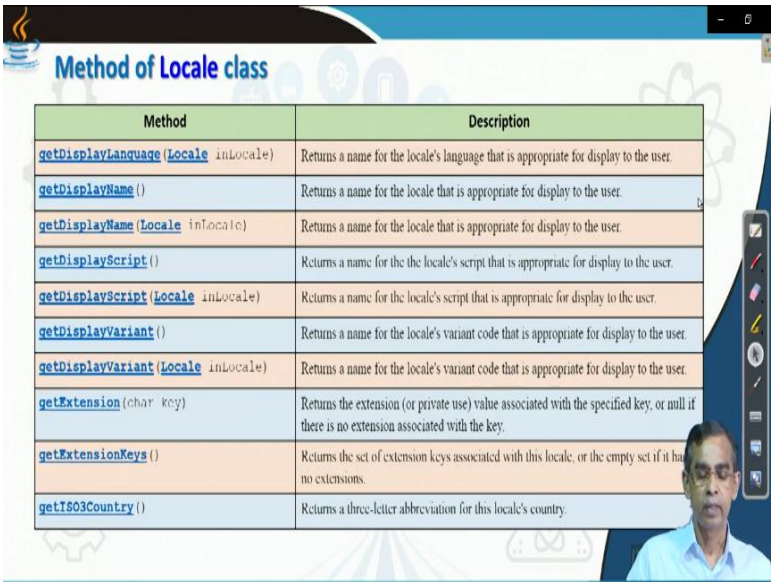
(Refer Slide Time: 25:30)




And there are many constructor that you can call for this locale class. Here is the different constructors, and this is basically the language that you can specify which locale; US or UK. So is a locale dot US, locale dot UK, locale dot Japan, these kind of things you can mention.

And then country also you can specify by giving the name and the locale also you can do it. So this is in many ways the locale object can be created and then you can customize your output to that purpose.

(Refer Slide Time: 26:03)



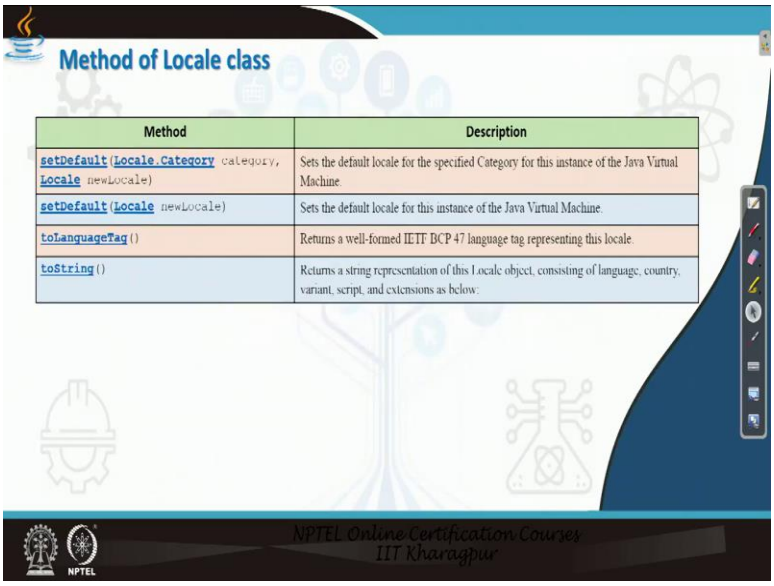
## Method of Locale class

| Method | Description |
|---|---|
| clone() | Overrides Cloneable. |
| equals(Object obj) | Returns true if this Locale is equal to another object. |
| forLanguageTag(String languageTag) | Returns a locale for the specified IETF BCP 47 language tag string. |
| getAvailableLocales() | Returns an array of all installed locales. |
| getCountry() | Returns the country/region code for this locale, which should either be the empty string, an uppercase ISO 3166 2-letter code, or a UN M.49 3-digit code. |
| getDefault() | Gets the current value of the default locale for this instance of the Java Virtual Machine. |
| getDefault(Locale.Category category) | Gets the current value of the default locale for the specified Category for this instance of the Java Virtual Machine. |
| getDisplayCountry() | Returns a name for the locale's country that is appropriate for display to the user. |
| getDisplayCountry(Locale inLocale) | Returns a name for the locale's country that is appropriate for display to the user. |
| getDisplayLanguage() | Returns a name for the locale's language that is appropriate for display to the user. |



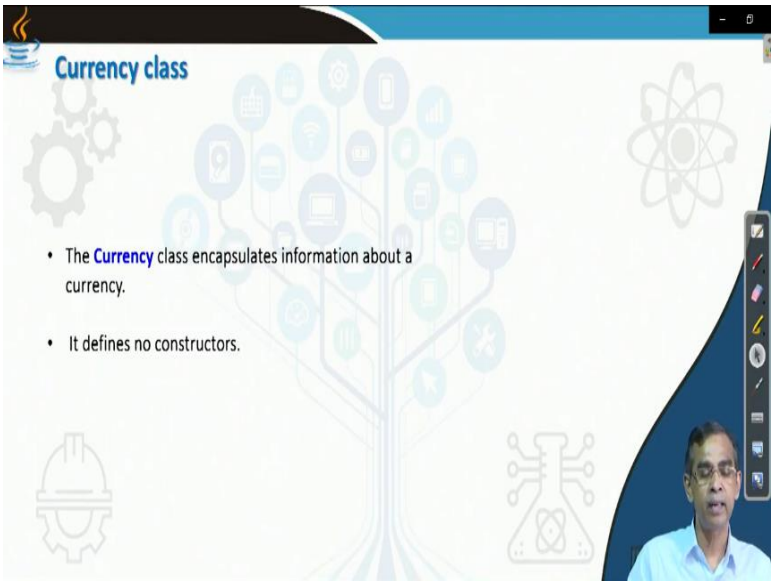
## Method of Locale class

| Method | Description |
|---|---|
| getISO3Language() | Returns a three-letter abbreviation of this locale's language. |
| getISOCountries() | Returns a list of all 2-letter country codes defined in ISO 3166. |
| getISOLanguages() | Returns a list of all 2-letter language codes defined in ISO 639. |
| getLanguage() | Returns the language code of this Locale. |
| getScript() | Returns the script for this locale, which should either be the empty string or an ISO 15924 4-letter script code. |
| getUnicodeLocaleAttributes() | Returns the set of unicode locale attributes associated with this locale, or the empty set if it has no attributes. |
| getUnicodeLocaleKeys() | Returns the set of Unicode locale keys defined by this locale, or the empty set if this locale has none. |
| getUnicodeLocaleType(String key) | Returns the Unicode locale type associated with the specified Unicode locale key for this locale. |
| getVariant() | Returns the variant code for this locale. |
| hashCode() | Override hashCode. |



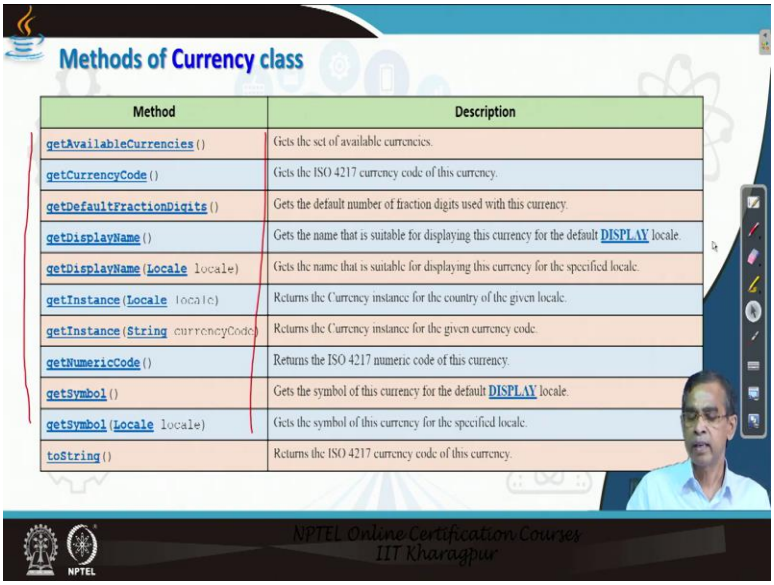

So these are the different methods, I have listed few methods here, these are the methods by which you can check that text comparison sort of thing. There are many more methods that you can go through and then learn it. And then you can write your program by calling those methods and creating objects and then you can see how they perform.

(Refer Slide Time: 26:30)




Now, currency utility, as you know, different currency are known, and those currency needs to be manipulated while you write the program. So the currency class is defined in java dot util packet.

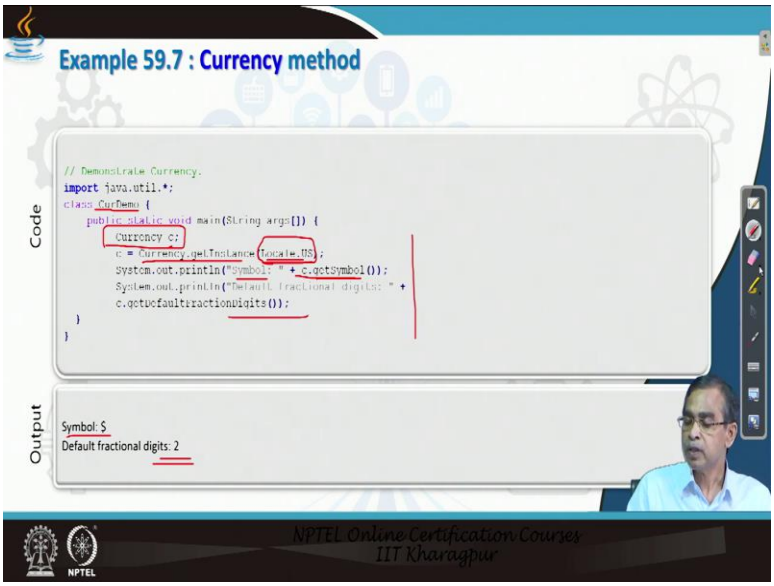
(Refer Slide Time: 26:43)




| Method | Description |
|---|---|
| getAvailableCurrencies() | Gets the set of available currencies. |
| getCurrencyCode() | Gets the ISO 4217 currency code of this currency. |
| getDefaultFractionDigits() | Gets the default number of fraction digits used with this currency. |
| getDisplayName() | Gets the name that is suitable for displaying this currency for the default DISPLAY locale. |
| getDisplayName(Locale locale) | Gets the name that is suitable for displaying this currency for the specified locale. |
| getInstance(Locale locale) | Returns the Currency instance for the country of the given locale. |
| getInstance(String currencyCode) | Returns the Currency instance for the given currency code. |
| getNumericCode() | Returns the ISO 4217 numeric code of this currency. |
| getSymbol() | Gets the symbol of this currency for the default DISPLAY locale. |
| getSymbol(Locale locale) | Gets the symbol of this currency for the specified locale. |
| toString() | Returns the ISO 4217 currency code of this currency. |

And there are many methods by which the currency can be manipulated. So these are the different methods. And these are the description that you can check and then you can learn about it.

(Refer Slide Time: 26:56)

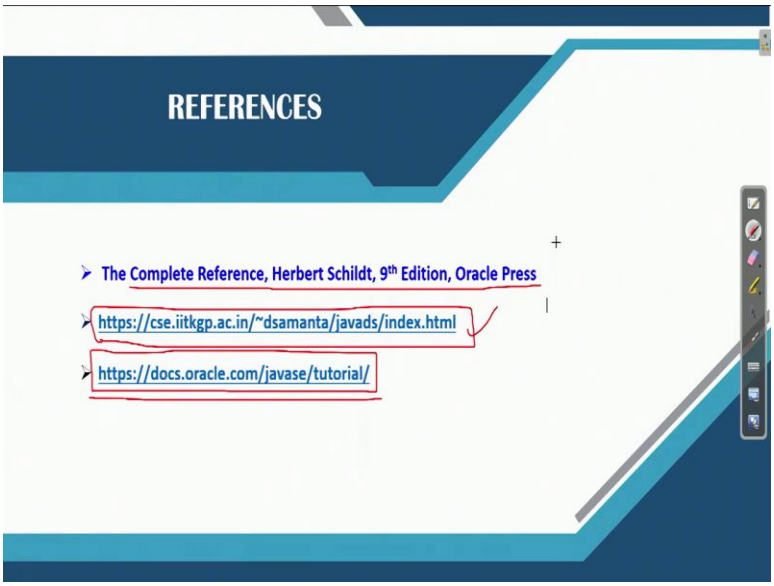



And then there are certain, and this is a method, this is the simple program that you can check that how this currency demo can work for some simple program. Here is a just we create a currency object, c. And this is the currency get instance locale dot US.

Now, it basically, we are creating the currency in the US locale and then symbol, we can print it. Now, get default fraction digits if in the US locale currency is there, there what is the fraction digit that you can obtain. So this is basically, this is a symbol that basically used this locale dot US currency and then fraction maximum allows two digits.

Now, here again, you can write that locale dot Japan and then get symbol. You can see Japan is Yen currency symbol, you will get it. And then the digit maximum it can allow it can see the digit that this currency allowed. For India it is basically digit 0. 1 paise, 2 paise, 3 paise like this one or maybe like this. So you can check it. Now so, this is the currency, the class that is there define in currency class.

(Refer Slide Time: 28:17)




And more discussion that you can obtain from these. This is basically exhaustive discussion of which you can find in this link. And these two; I have listed a few supplementary materials for your understanding in details which I could not covered in this presentation. So this also you can read so that you can get more details what I wanted to convey here.

Otherwise, if you want to learn in more details, then that is the link you can follow. There are some discussion in this book also available, but no book is available where you can find detailed discussion with more illustration. So that is the story of this one.

And that is why you can you can rely on some internet document also searching the document. You can find many more examples relating this calendars, current zone or currency, and then time zone and many more other utilities are there. So it require some, definitely, you have to put some effort to understand this one. And then time, of course, then you will be able to understand it. Thank you very much.