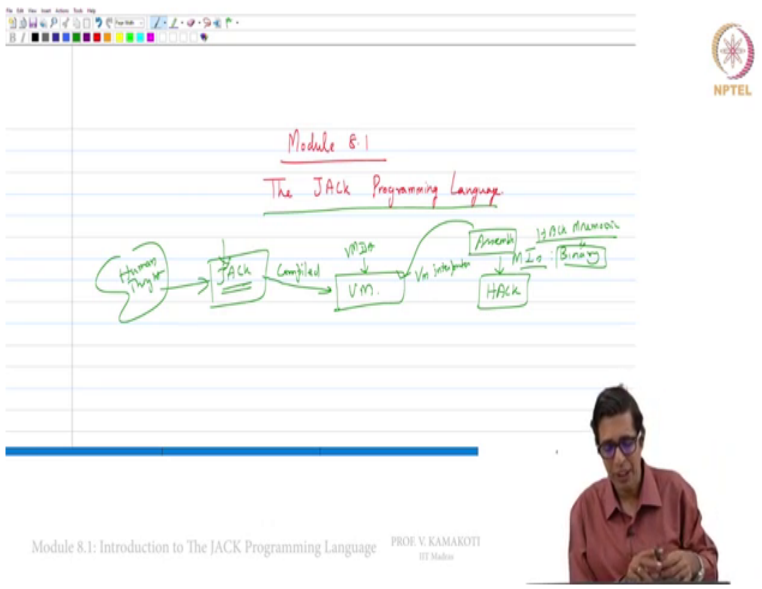
Foundations To Computer Systems Design
Professor V.Kamakoti
Department of Computer Science and Engineering
Indian Institute of Technology, Madras
Module 8.1
Introduction to The JACK Programing Language

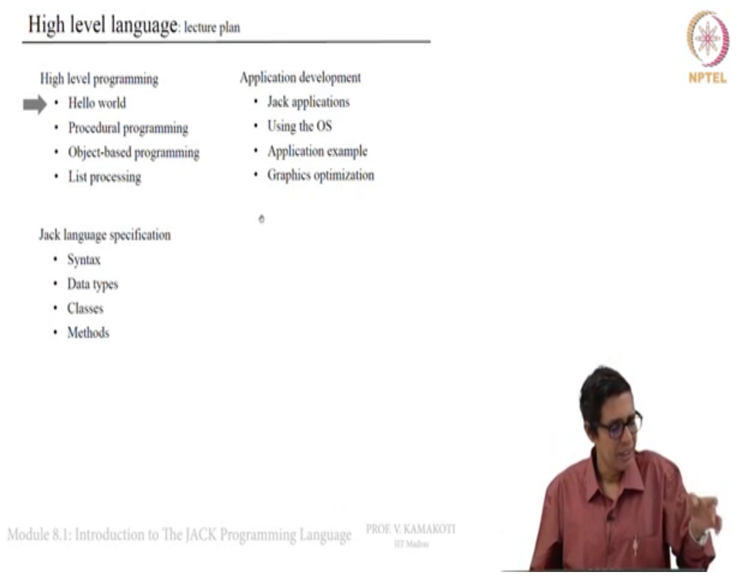
(Refer Slide Time: 00:17)



Welcome to module 8.1 and now we move on to the next phase which is basically the JACK Programing Language. So just to recap what we have done we have done we have created machine called HACK and this add machine instructions now this machine instructions were in binary so there was an hack mnemonic we wrote assembler which will convert from hack mnemonic to binary. When we define the virtual machine and we have written a virtual machine interpreter which will convert from this (())(01:07) virtual machine ISA from this virtual machine to hack mnemonic. Now we are defining the high level programing with this jack now this will be compiled so we will be writing programing in jack and that will be compiled into the virtual machine and that virtual machine interpreter will make it into assembly, assembly will assembler will convert that mnemonic to binary and then you can execute on hack.

So this is the entire sequence that we are going to follow now, now what we will be learning in this module 8.1 and 8.2 is to basically understand how do you program in this jack? So you have some human thought in mind and you want to just convert it into a jack program (())(01:53) jack is nothing but an (())(01:56) version of java. How do go and write that part in (())(02:00) so we will because before you start interpreting the (language) or compiling the language before you start writing the compiler for the language it is very important that you understand the constructs of the language and the best way to understand the construct of the language is to you know understand programing is in tat language. So will see a set of programs and show you how it works before we proceed on to how you write the compiler ok.

(Refer Slide Time: 02:32)



High level language: lecture plan

High level programming
- Hello world
- Procedural programming
- Object-based programming
- List processing

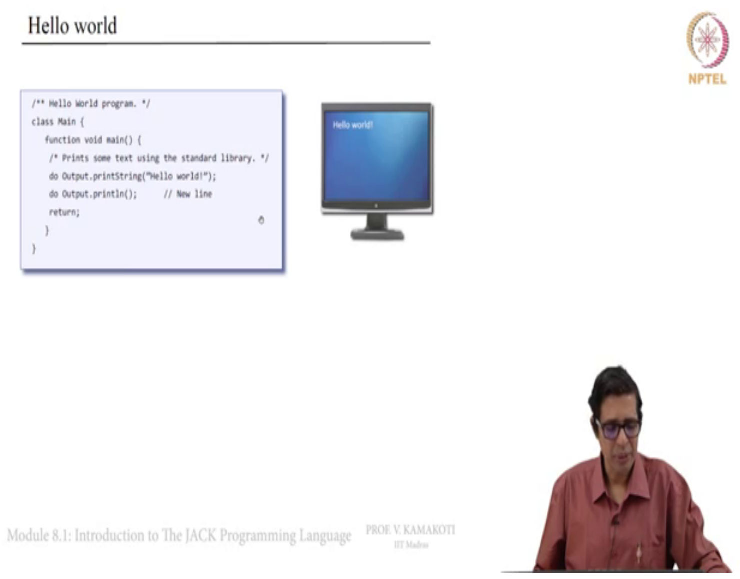
Application development
- Jack applications
- Using the OS
- Application example
- Graphics optimization

Jack language specification
- Syntax
- Data types
- Classes
- Methods

Module 8.1: Introduction to The JACK Programming Language PROF. V. KAMAKOTI IIT Madras

Now I am using the slides that are available in the (())(02:35) org website so you can download the slides from the project folder there in the (())(02:43) org for your reference. So the way we go about this entre lecture is that we will start looking at high level programing then we look at some amount of application development using this high level programing and then finally we end this module 8 with the specification of the jack language.

(Refer Slide Time: 03:12)



Hello world

```
/** Hello World program. */
class Main {
   function void main() {
    /* Prints some text using the standard library. */
    do Output.printString("Hello world!");
    do Output.println();      // New line
    return;
   }
}
```

Now let us now start looking at this the first programing anybody wants to write is the hello world program. Now in jack, like C there is a routine called main function called main and that will be the first function that you execute. If you are looking at you know other things like java or any other C++ you need not even go to this lecture you will understand this lecture very quickly but I am doing this lecture for people who have not studied objective inter programing languages like C++ or java but they are good in C ofcourse right so now as in C gets small main that function main will execute first now in jack right there is always a class called main it starts with capital M and inside that class there is going to be a small main function called small main, always this main dot main will execute first.

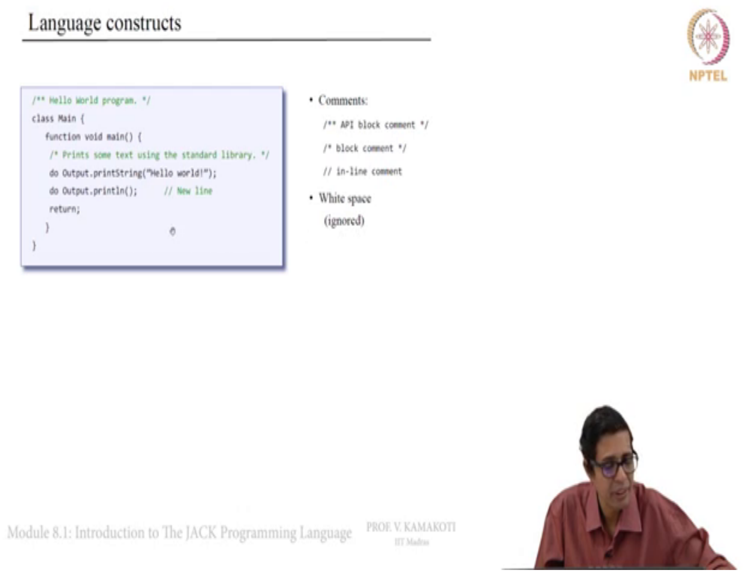
So this is the basic, now for a quick understanding of what a class is you would have studied struct in C right the struct in C has some equivalence to class though it is not fully equivalent but struct in C is having some equivalence to class. What is struct? Struct is basically a template and you instantiate the template again and again to create several objects right. Now what is stored inside the struct are only data items, class is also a template you instantiate that class again and again to create many objects. The difference between struct and glass is, in struct we only store the data items like integers, floating points etc.

In class we store the data items in addition we also store certain functions and methods by which you can manipulate those data so that is the difference between struct and class right. So the hello world program always has a any program always has a class called main with M capital as you see here and then inside that main you have another function void main small ok right so always you will execute main dot small main. So when you start executing main dot small main here when you compile this program and execute we will start executing main dot small main and what it will do is it does this two things do output dot print string hello world do output dot print l n and it will return.

Return back to whom? The operating system that's it and this is the end of this so this do output that (print screen) print string is equivalent to a print f statement and do output that print line is new line character so and so what we have understood so when what will be the output when you compile this using the compiler which is available as a part of your project the output will be just hello world here right so this is how this works. So what we have learned here, we have learned that here is a class (())(06:31) jack is based on classes there is always a class called main with capital M inside that class main there should be a function void small main and that will be the first function that will execute and that will call the other functions and other classes instantiate other classes etc and then the way I can output anything is I can use this so output that (())(06:53) print string I can print any string I want and I can also use new line output dot print n l and it should be do output dot print string do output dot print n l (())(07:06)

So who is going to provide me with this print string and print essentially this output, output class so this output is a class and print string is a function inside that class like main is a class capital main is a class and the small main is a function within that class inside this output is a class print string is a function within that class and similarly here also output print. So who is going to provide me this class called output which has all this functions inside, the operating system is going to provide me. Like how print f and scan f are provided by the operating system in the case of C programing here this output class which contains those functions print string and print (()) (07:54) etc here is going to be provide by the operating system. So this is an understanding of the hello world program.

(Refer Slide Time: 08:03)



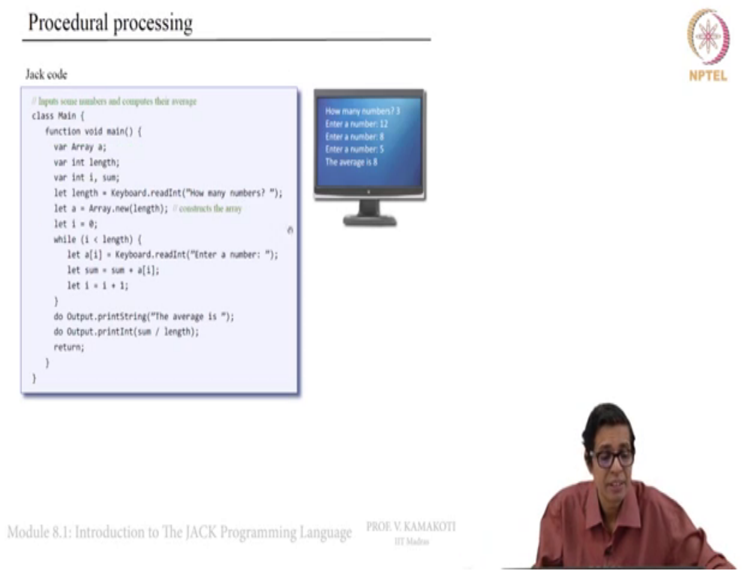
Language constructs

```
/** Hello World program. */
class Main {
   function void main() {
    /* Prints some text using the standard library. */
    do Output.printString("Hello world!");
    do Output.println();      // New line
    return;
   }
}
```

- Comments:
  - `/** API block comment */`
  - `/* block comment */`
  - `// in-line comment`
- White space (ignored)

So what are the things here? All the wide spaces are ignored and this are the different ways by which I can put a command right so there are three types of commands this called application API stands for Application Programmer Interface so slash star-star hello world program slash star this is an API block command then I could have just a simple block comment which is and I could have just inline comment like this right. So I could have three types of comments as you see here and then this wide space always all the wide spaces are ignored (())(08:51).

(Refer Slide Time: 08:54)



Procedural processing

Jack code

```
// Inputs some numbers and computes their average
class Main {
   function void main() {
     var Array a;
     var int length;
     var int i, sum;
     let length = Keyboard.readInt("How many numbers? ");
     let a = Array.new(length); // constructs the array
     let i = 0;
     while (i < length) {
        let a[i] = Keyboard.readInt("Enter a number: ");
        let sum = sum + a[i];
        let i = i + 1;
     }
     do Output.printString("The average is ");
     do Output.printInt(sum / length);
     return;
   }
}
```

```
How many numbers? 3
Enter a number: 12
Enter a number: 8
Enter a number: 5
The average is 8
```

Now let us take this next program right so this is function void main so again there is a class main function void main now we are trying to introduce more syntax here, more so I am declaring an array a small a where array a. Now in jack this array are not type so I can store anything in that array right but they are going to be 16 bit variables so I can store integers, I can store string whatever we want like 16 bit is the size of element here var array a var int length var int I, sum so I have declared an array a I have declared length, I have declared I, sum.

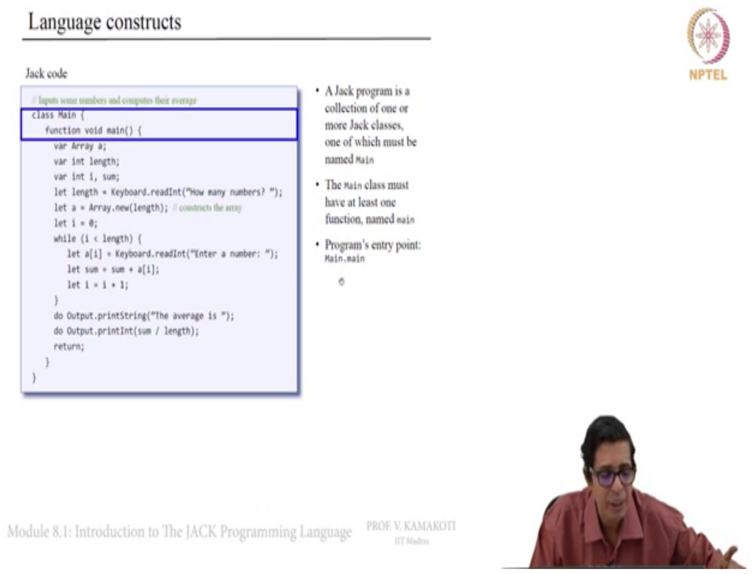
Now let length equal to what does (())(09:53) in C we just say length equal to something but in jack we say let length equal to keyboard dot read int how many numbers, so keyboard is a class inside that class there is something called read int which will be an integer from the keyboard. So who is providing this keyboard thing? The operating system is providing us right. So keyboard dot read int will read and how many numbers, so once this statement gets executed how many numbers, question will come on the screen then you press 3 and press enter right so 3 or whatever in this case. Now it will go and say let a equal to array dot new of length so it will create an array of length 3 right of so a0 array (a) there will be a0, a1 and a it will create array of size 3.

So this is equivalent to the malloc statement that you see malloc, malloc statement that you see in the C ok. Now what we do so we have then we do why I is less than length, let a equal to keyboard dot read int enter a number sum equal to sum plus a I let I equal to I plus 1 this goes on. So this loop will print this enter a number say 3 times in this case because you said length is 3 so it will be 12, 8 and 5 so that will be stored in a of 0, a of 1 and a of 2 as you see here I sum this up ok then after finishing this while loop we say the output dot print string the average is and do output dot print int sum by length return.

So sum this will be 25 by 3 which is the integer part is 8 so the average is 8. So this is how this particular function works and this is how the input output happens on the (screen) right and then you return back so this is basically. So what we have learnt here is the motion of what is array, how to declare an array, a while loop is there, how to use an array, how to read input from a keyboard and finally how the while loop statement that you have seen here and the malloc statement that you have seen here. How to declare an array, construct an array, read input from the keyboard and how to use the array and right using how to allocate space for that array equivalent to malloc, how to use the array and some statements like some arithmetic statement like some by length etc right.

So this is part of the procedure program so we are learning more and more syntax as we see here, you can type this programs and when compile in C I will show you how to go and do that right.

(Refer Slide Time: 13:10)



Language constructs

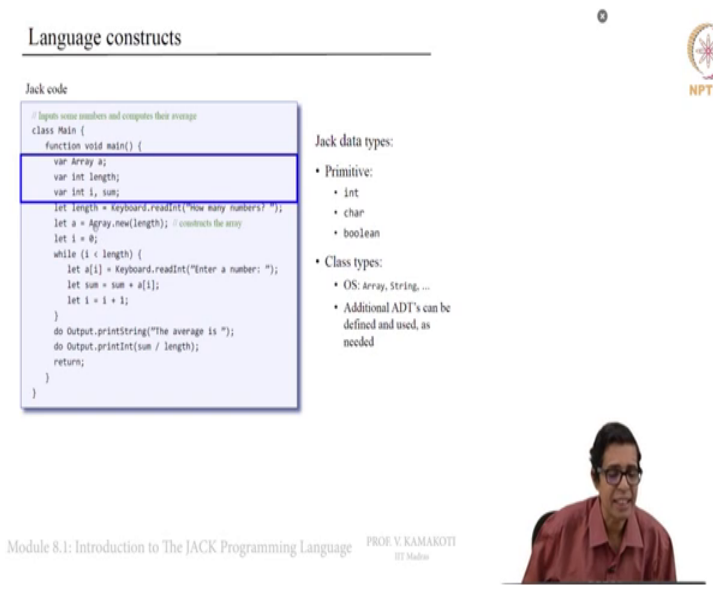
Jack code

```
// Inputs some numbers and computes their average
class Main {
   function void main() {
      var Array a;
      var int length;
      var int i, sum;
      let length = Keyboard.readInt("How many numbers? ");
      let a = Array.new(length); // constructs the array
      let i = 0;
      while (i < length) {
         let a[i] = Keyboard.readInt("Enter a number: ");
         let sum = sum + a[i];
         let i = i + 1;
      }
      do Output.printString("The average is ");
      do Output.printInt(sum / length);
      return;
   }
}
```

- A Jack program is a collection of one or more Jack classes, one of which must be named `Main`
- The `Main` class must have at least one function, named `main`
- Program's entry point: `Main.main`

So to sum up a jack program is a collection of one or more jack classes one of which must be main capital main and the main class must have atleast one function main it should have atleast one function and that function should be named main and the programs entry point is always main dot (())(13:32) right.

(Refer Slide Time: 13:36)



Language constructs

Jack code

```
// Inputs some numbers and computes their average
class Main {
   function void main() {
      var Array a;
      var int length;
      var int i, sum;
      let length = Keyboard.readInt("How many numbers? ");
      let a = Array.new(length); // constructs the array
      let i = 0;
      while (i < length) {
         let a[i] = Keyboard.readInt("Enter a number: ");
         let sum = sum + a[i];
         let i = i + 1;
      }
      do Output.printString("The average is ");
      do Output.printInt(sum / length);
      return;
   }
}
```

Jack data types:

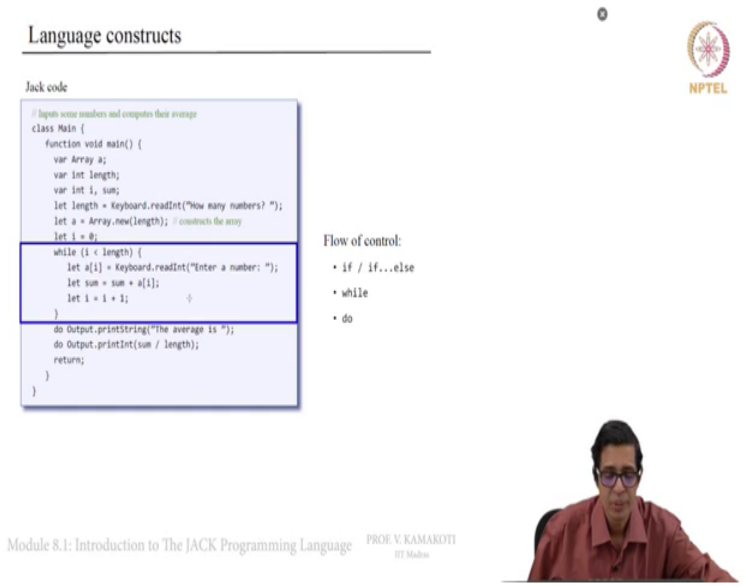
- Primitive:
  - int
  - char
  - boolean
- Class types:
  - OS: Array, String, ...
  - Additional ADT's can be defined and used, as needed

Module 8.1: Introduction to The JACK Programming Language PROF. V. KAMAKOTI IIT Madras

So there are different data types for jack there are primitive data types like int, char and Boolean so we have just var, int, length we could have var, char whatever it will be character and we could have var Boolean sum right it could be a Boolean here.

So this are the three primitive data types we have only seen integer here then there are certain class types which are basically arrays string etc and there can be additional abstract data types can be defined and used as needed, so as if now the basic data types are int, char and Boolean and then we have class types which are provided by the operating system which we have one of them we have used this is array, this array is a class and so the moment I declare an array I can have array dot new, new is a function inside the class this will create space for storing that many number of elements in that array, etc so similarly I could have also string ok.

(Refer Slide Time: 14:51)



Language constructs

Jack code

```
// Inputs some numbers and computes their average
class Main {
   function void main() {
     var Array a;
     var int length;
     var int i, sum;
     let length = Keyboard.readInt("How many numbers? ");
     let a = Array.new(length); // constructs the array
     let i = 0;
     while (i < length) {
        let a[i] = Keyboard.readInt("Enter a number: ");
        let sum = sum + a[i];
        let i = i + 1;
     }
     do Output.printString("The average is ");
     do Output.printInt(sum / length);
     return;
   }
}
```

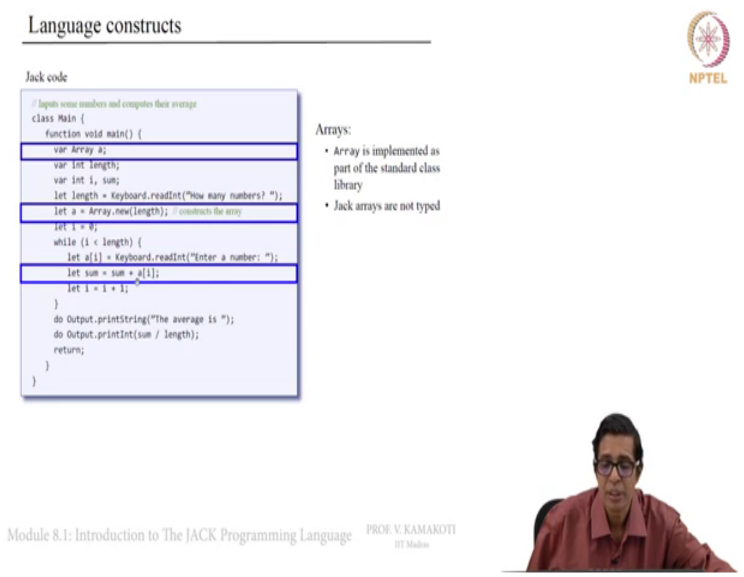
Flow of control:

- `if / if...else`
- `while`
- `do`

Module 8.1: Introduction to The JACK Programming Language PROF. V. KAMAKOTI IIT Madras

Then you have this, this is a flow control because I have while statement here so I have while I have do statement and I also have if and if else right we have not used if and if else so far but we can use if and if else.

(Refer Slide Time: 15:09)



Language constructs

Jack code

```
// Inputs some numbers and computes their average
class Main {
   function void main() {
     var Array a;
     var int length;
     var int i, sum;
     let length = Keyboard.readInt("How many numbers? ");
     let a = Array.new(length); // constructs the array
     let i = 0;
     while (i < length) {
        let a[i] = Keyboard.readInt("Enter a number: ");
        let sum = sum + a[i];
        let i = i + 1;
     }
     do Output.printString("The average is ");
     do Output.printInt(sum / length);
     return;
   }
}
```

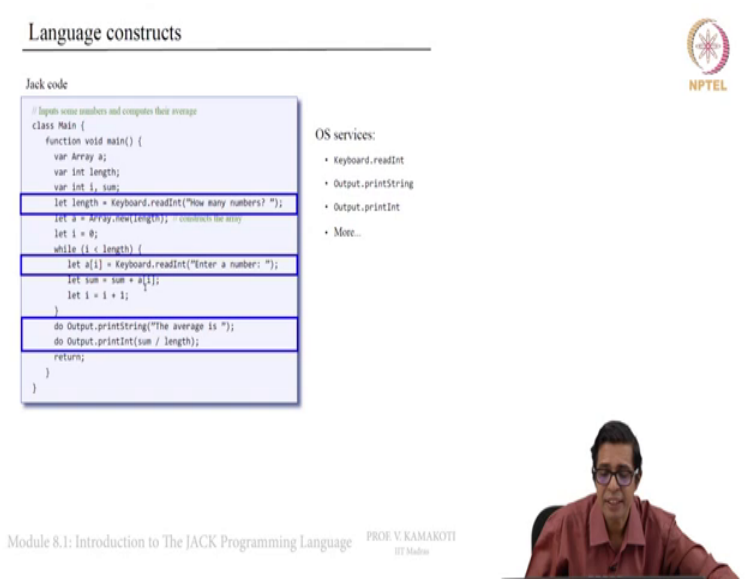
Arrays:

- `Array` is implemented as part of the standard class library
- Jack arrays are not typed

Module 8.1: Introduction to The JACK Programming Language PROF. V. KAMAKOTI IIT Madras

So let us go and look at the array, array is actually implemented as part of the standard class library and the jack arrays are actually not typed so this is a class and you can this are (())(15:26) equivalent and this is the way you can use an array.

(Refer Slide Time: 15:29)



Language constructs

Jack code

```
// Inputs some numbers and computes their average
class Main {
   function void main() {
     var Array a;
     var int length;
     var int i, sum;
     let length = Keyboard.readInt("How many numbers? ");
     let a = Array.new(length); // constructs the array
     let i = 0;
     while (i < length) {
        let a[i] = Keyboard.readInt("Enter a number: ");
        let sum = sum + a[i];
        let i = i + 1;
     }
     do Output.printString("The average is ");
     do Output.printInt(sum / length);
     return;
   }
}
```

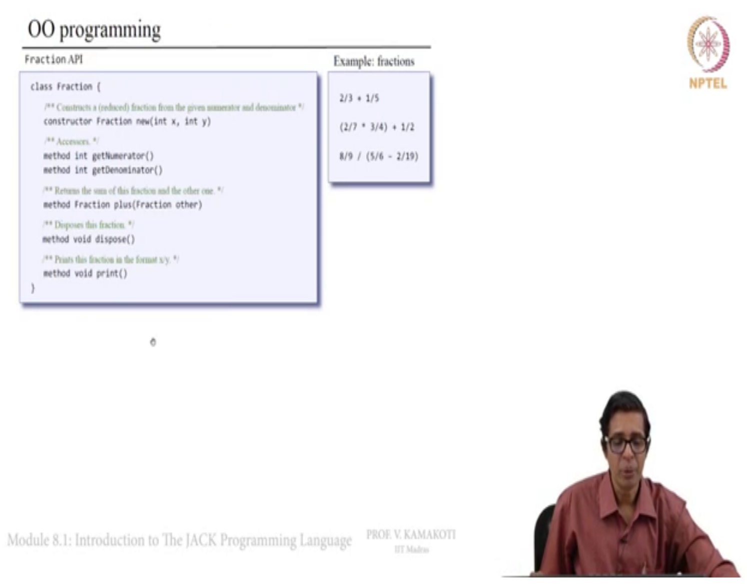
OS services:

- `Keyboard.readInt`
- `Output.printString`
- `Output.printInt`
- More...

Module 8.1: Introduction to The JACK Programming Language PROF. V. KAMAKOTI IIT Madras

Similarly keyboard the way services that you see are keyboard dot read int output dot print screen output dot print int and many more things these are all equivalent to the scan f and print f type of statements right.

(Refer Slide Time: 15:54)



OO programming

Fraction API

```
class Fraction {
   /** Constructs a (reduced) fraction from the given numerator and denominator */
   constructor Fraction new(int x, int y)
   /** Accessors. */
   method int getNumerator()
   method int getDenominator()
   /** Returns the sum of this fraction and the other one. */
   method Fraction plus(Fraction other)
   /** Disposes this fraction. */
   method void dispose()
   /** Prints this fraction in the format x/y. */
   method void print()
}
```

Example: fractions

```
2/3 + 1/5
(2/7 * 3/4) + 1/2
8/9 / (5/6 - 2/19)
```

Module 8.1: Introduction to The JACK Programming Language PROF. V. KAMAKOTI IIT Madras

Now with this we will stop looking at what we call as the object oriented programing we will now basically go into some of the details. Before proceeding further into this object oriented programing details we will quickly look at the project, part of your project 9 where we will show you show things are basically working right thank you.