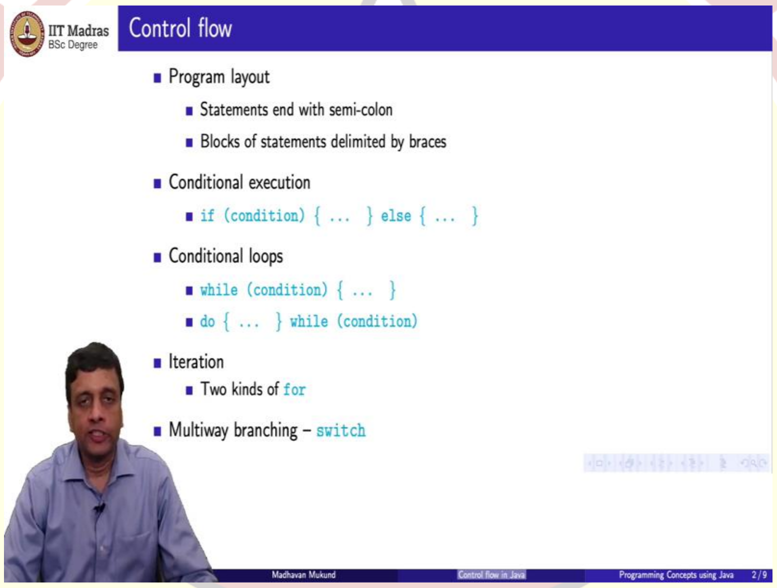
IIT Madras

ONLINE DEGREE

Programming Concepts Using Java
Online Degree Programme
B. Sc in Programming and Data Science
Diploma Level
Prof. Madhavan Mukund
Department of Computer Science
Chennai Mathematical Institute

Lecture 9
Control Flow in Java

(Refer Slide Time: 00:21)



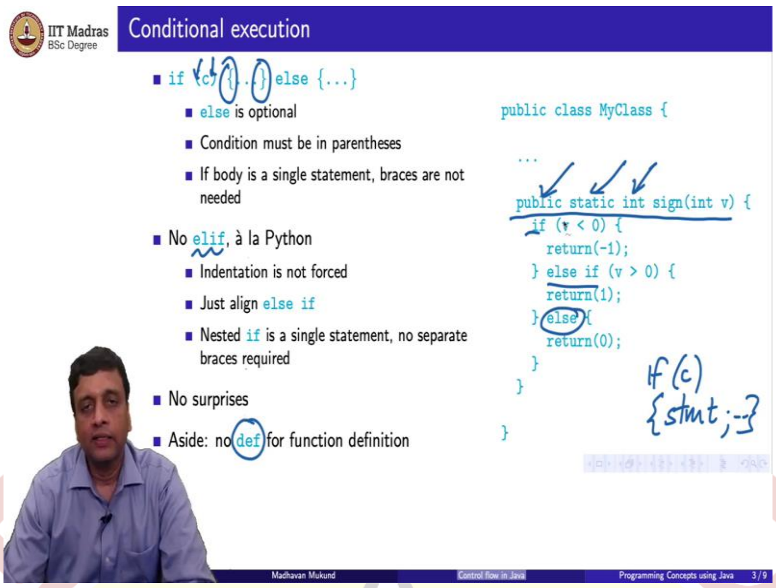
Control flow

- Program layout
  - Statements end with semi-colon
  - Blocks of statements delimited by braces
- Conditional execution
  - `if (condition) { ... } else { ... }`
- Conditional loops
  - `while (condition) { ... }`
  - `do { ... } while (condition)`
- Iteration
  - Two kinds of `for`
- Multiway branching – `switch`

So, we have seen the basic data types in java now let us look at control flow how the program actually executes. So, the first thing that we noted about java is that its text is laid out differently from a python program. So, in python we use remember indentation and layout to indicate the structure of a program whereas in java we are going to use statements ending with a semicolon and blocks of statements delimited by braces.

So, like in any conventional imperative programming language java has an if statement for conditional execution it has 2 different types of conditional loops. So, the while loop is something that we are familiar with from python but java also has something called a do while which you will see. It also has 2 kinds of iteration using for and finally it has a multi-way branching statement called switch.

(Refer Slide Time: 01:13)



IIT Madras
BSc Degree

Conditional execution

- if (c) {...} else {...}
  - else is optional
  - Condition must be in parentheses
  - If body is a single statement, braces are not needed
- No elif, à la Python
  - Indentation is not forced
  - Just align else if
  - Nested if is a single statement, no separate braces required
- No surprises
- Aside: no def for function definition

```
public class MyClass {

  ...

  public static int sign(int v) {
    if (v < 0) {
      return(-1);
    } else if (v > 0) {
      return(1);
    } else {
      return(0);
    }
  }

}
```

Madhavan Mukund Control flow in Java Programming Concepts using Java 3/9

So, let us look at the conditional statement first. So, the conditional statement is similar to python. So, you have an if condition then you have an if block and then you have an else block but remember that you have to put these braces around these blocks and as in any other programming language the else is optional. So, you do not have to have an else one thing is the condition must be in bracket.

So, you must put these brackets here. So, here is some simple code. So, for instance we write a function which is a sine function which takes an integer and if the integer is negative it returns a - 1 if the integer is positive it returns a + 1 and if the integer is 0 it returns 0. So, here we have this if v less than 0 return - 1 else. So, now we have a multi way condition.

So, if it is not less than 0 then it is either greater than 0 or it is 0. So, in python if we do this because of indentation this will go inside it goes one block inside. And so if we have a sequence of such conditions here it is three but if we add a four way choice for instance then the python code will keep traveling right. So, python has this thing called elif it allows us to write an if with multiple else clauses which span different cases for that variable and one reason to have this elif in python is because of the layout.

Because in python you would start with the first if then you would have to go and else and then else and. So, you will keep traveling to the and therefore you will run out of screen space or whatever space you are reading the programming. So, in python in java of course we do not have to indent. So, indentation is up to us. So, here for instance this is the else and now actually you should think of this whole if here as being nested inside that else.

So, the way it works in java is that this whole if is considered a single statement. So, the real statement is if c statement. So, this is the real if statement and if the single statement inside the if is more than one then you make it a block. So, the block is optional if you have a single statement. So, in this case this entire span from this if to this point is one single statement. So, there is an if sitting inside the else. So, it is if statement else statement. So, we do not have to put another brace and as a result we can just lay it out like this we write if this else if this.

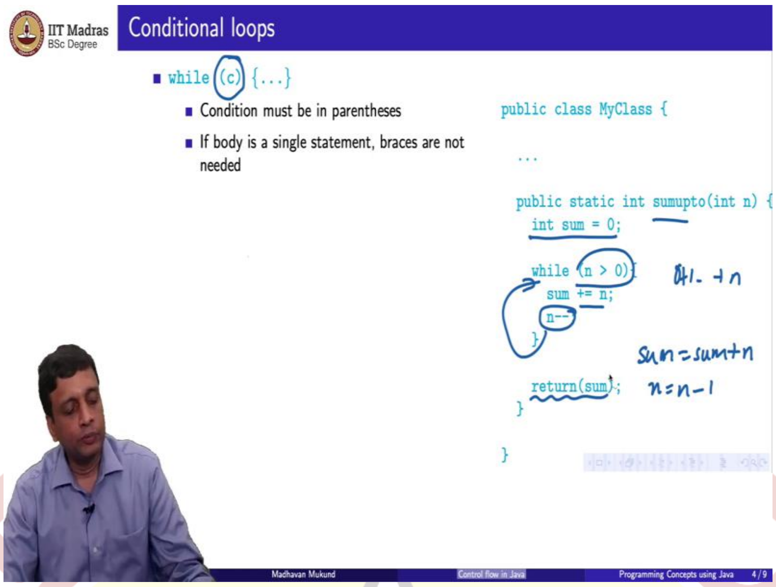
And then finally we have an s now if you add in between more cases we could just keep writing else if without going off the page. So, we can just align the else if's and have this multi way else and if just like the python elif but without having requiring a separate statement for that. So, basically the if statement is similar to what we are familiar with python except for this elif and this lack of indentation.

So, there are no surprises one thing is that just to remind you that we have defined a function here. So, if you remember from our first example of the main function for printing hello world. So, we have a return value we have this static indicating in this case that it is created with the class and does not require an object to be there it is publicly visible but notice that there is no separate statement saying that I am defining a function.

So, in python we have to write a def and then a function name. So, that python knows it is a definition of a function and not a class or something. So, in general in java anytime you have a block like this it is believed to be a function name unless you say otherwise. So, when we define classes we use the word class but when we define functions we do not have to write a separate word to indicate that we are defining a function.

So, that is as far as conditional execution goes is pretty straight forward just some minor syntactic deviation from what we know in python.

(Refer Slide Time: 05:02)



Conditional loops

- `while (c) {...}`
  - Condition must be in parentheses
  - If body is a single statement, braces are not needed

```
public class MyClass {

  ...

  public static int sumupto(int n) {
    int sum = 0;

    while (n > 0){
      sum += n;
      n--;
    }

    return(sum);
  }

}
```

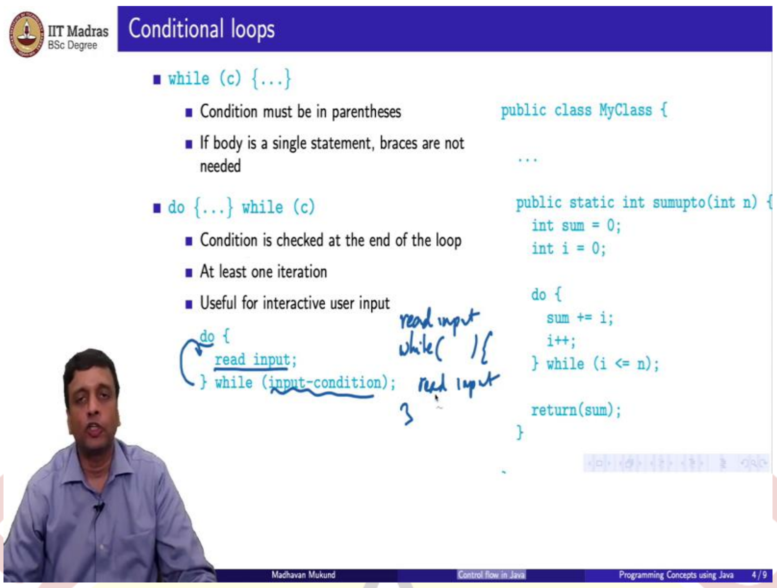
Again while is very similar to python except for the fact that we have again this braces and we do not have indentation as a requirement. So, once again the condition must be in parentheses. So, the usual semantics of while. So, you test the condition if it is true you enter the while finish what you do come back. So, here is a typical example of a function which in this case adds up the numbers 0 + 1 up to n.

So, it is sum up to n. So, you start by initializing sum to be 0 and now what we are going to do is we are going to start from n and count down to 0. So, while n is greater than 0 we add the current value of n to some. So, remember this shortcut this is the same as sum equal to sum plus n. So, we mentioned this last time. So, it is convenient to save some space also when writing these things and then this is the decrement operator this is n is equal to n - 1.

So, I have written it deliberately just to remind you. So, that you get familiar with these shortcuts which are quite common when you read java code. So, you take the sum update it with the current value of n and you decrement n and then n will be tested again and since we are starting with some value of n which is positive it will come down to 0. And if n was negative to start with you would already fail this test and you would return the sum to be 0.

So, finally like any other function you return sum. So, the while loop is fairly simple nothing to be nothing new there.

(Refer Slide Time: 06:40)



Conditional loops

- `while (c) {...}`
  - Condition must be in parentheses
  - If body is a single statement, braces are not needed
- `do {...} while (c)`
  - Condition is checked at the end of the loop
  - At least one iteration
  - Useful for interactive user input

```
do {
  read input;
} while (input-condition);
```

```
read input
while( ){
  read input
}
```

```
public class MyClass {

  ...

  public static int sumupto(int n) {
    int sum = 0;
    int i = 0;

    do {
      sum += i;
      i++;
    } while (i <= n);

    return(sum);
  }
```

Madhavan Mukund   Control flow in Java   Programming Concepts using Java   4 / 9

So, what is new in java compared to python is that you have a while loop in which the while is tested at the end. So, here is a different way of doing this same function. So, you start at 0 and then you go up to n. So, you start with I equal to 0. So, initially you add i 0 then you add i + 1 and if i is less than or equal to n then you continue so that you will add the next value.

So, until the time that it becomes equal to n you will be incrementing i. When it becomes strictly greater than n this while will fail and you will come out. So, the while is tested at the bottom of the loop rather than at the top of the loop. So, the implication therefore is that when you enter for the first time it is not taking anything. So, you will always go through this loop at least once.

So, in the while loop you could come out of the while loop without doing anything. So, in the previous example if we started here and n was already negative or n was already 0 we would never enter the while whereas here we will always enter the while loop but it does not matter in case we check you can check it if n is 0 for instance you will start by adding 0 then i will become 1 and it will come out.

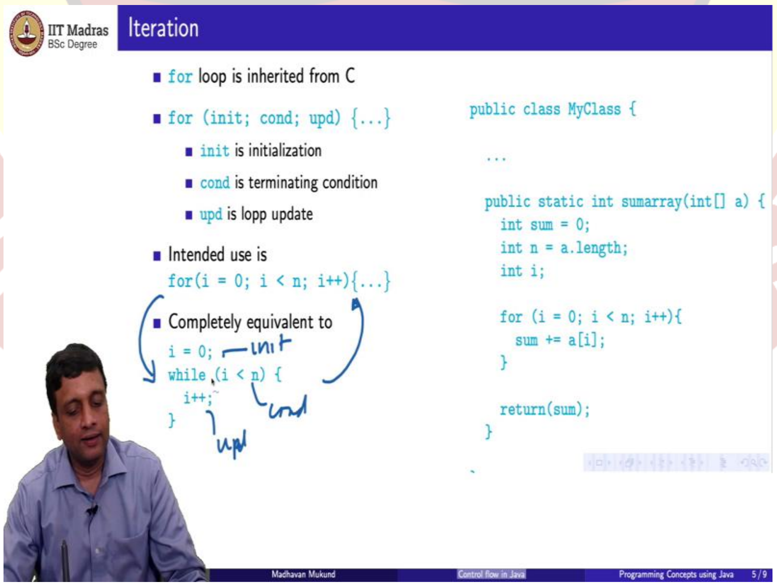
If it is negative again you will add 0 i will become 1 and you will come out. So, you it will still be the same. So, there is no real difference between entering it in this way or entering it in the other way. So, why would you want to do something like this well a typical example where this kind of a loop is useful is when you are actually asking for input and checking whether the input is something sensible.

For instance you might say keep typing something and when you end finish type a blank line. So, you are expecting the user to enter something and then the end of input is signified by entering a blank line. So, you would like to always start by reading a line. So, here is a simple do while loop. So, the do says first read an input line and then check whether the input line satisfies some condition or not.

So, for example if the input line is not empty then the reader is not the right user is not finished. So, you will go back and read one more line and if the input line is empty you will exit. Now if you did this in the conventional way you would have to first have an isolated read input outside and then you will have to say while something read input. So, you will have this duplication of this read input once for the first time and once for the second time onwards.

So, this is a little cumbersome when you when you definitely want to do something at least once the do while is more natural than the while do. So, I would say that in ninety nine percent of the cases we end up using the normal while but there are situations where the do while is useful to have.

(Refer Slide Time: 09:18)



IIT Madras
BSc Degree

Iteration

- for loop is inherited from C
- for (init; cond; upd) {...}
  - init is initialization
  - cond is terminating condition
  - upd is lopp update
- Intended use is
  for(i = 0; i < n; i++){...}
- Completely equivalent to

```
i = 0;  — init
while (i < n) {
  i++;
}
```

cond
upd

```
public class MyClass {

  ...

  public static int sumarray(int[] a) {
    int sum = 0;
    int n = a.length;
    int i;

    for (i = 0; i < n; i++){
      sum += a[i];
    }

    return(sum);
  }
```

Madhavan Mukund   Control flow in Java   Programming Concepts using Java   5 / 9

So, the do while and the while do are conditional loops. So, they keep iterating. So, long as a condition is true. So, it is up to you to make sure that the condition becomes false because if the condition never becomes false and the loop will not terminate and you will have an infinite execution and the program will not end. So, for loop or the iteration is intended to have the behavior where you go through a fixed sequence.

So, it is unlike a while you are supposed to go through a number of values of a fixed thing. So, in python for instance you will go through a list of values. So, typically in python you say for x in l. So, l is a fixed list and you go x goes to the first 0th element the first element and so on or if you want numbers you will say for x in range n then it will go from 0 to n - 1.

So, you have a number of iterations which is prescribed in advance. So, that is the usual interpretation of iteration or a for loop. So, it is in some sense it is determinant it determined in advance how many times the loop is going to execute unlike a while loop which will execute. So, long as the condition remains true. Now strangely the basic for loop in java is actually derived from the for loop in languages like C and it does not have this determinate behaviour.

So, the for loop normally you would expect a starting point and an ending point you want to say go from here to there go from 1 to n go from 0 to 75. But in general what a for loop allows is you initialize something then you update it and then you test the condition. So, it is very much actually like a while loop. So, let us first look at an example of how this would work.

So, a typical example of an iteration is to run through for example an array and add up the elements. So, let us look at that. So, here we have sum array which takes as input and array. Now remember that in java you can extract the length of the array. So, you do not have to know how many elements are there in advance. So, you create an internal variable n which is the length of the array and now you want to run through the loop.

So, you use a loop variable i and this is how the for loop works you initialize the i to 0 you increment at the end of every iteration the value of i by 1. So, remember this is i equal to i + 1 and you check whether i has reached n or not. So, so long as i is smaller than n. So, you will go from 0 to n - 1. So, for each value of i in this thing you look at the ith element in the array and you add it to sum which initially was 0.

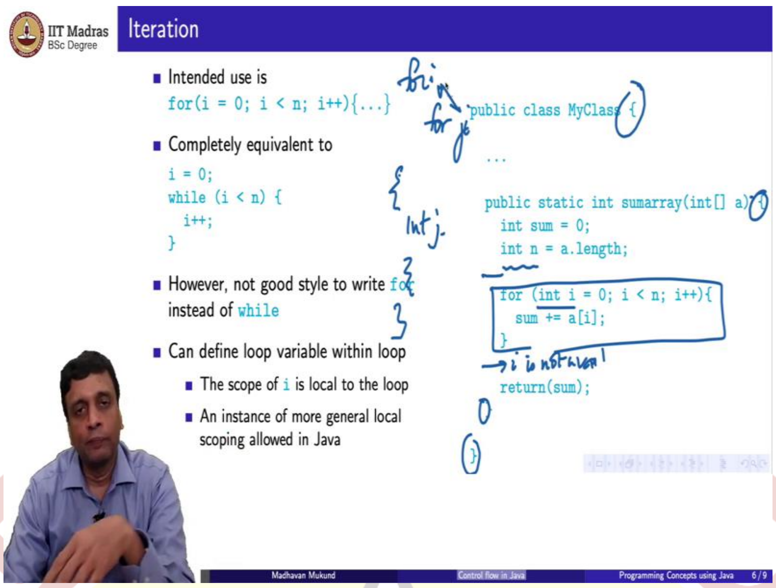
So, we start with sum equal to 0 add a 0 add a 1 add a 2 and so on. So, once again remember this plus equal to operator. So, this is the intended use, I do not know why this is lop update ok. So, this is the intended use the intended use is to say you start from some starting point and increment up to some ending point but of course now this is flexible. So, you can increment for instance in steps of 2.

You can say instead of i++ you can say i + equal to 2. So, you can skip over alternate elements you can do anything you want. So, formally if you look at this loop actually it is exactly this loop. So, this is the initialization this is the condition and this is the update. So, in this case there is literally no difference between a for loop and a while. So, normally we can say in a general language that you can always take a for loop and write it as a while loop.

So, while is able to express for but it is not necessarily the other way around. So, you cannot take a while loop in python and write a for loop in python with it because you cannot express in that for statement in python when you write for x and l you cannot write conditions and stuff like that which you need for a while. But here there is absolutely just syntactically the same.

I can take a while loop and I can go back to a for loop. So, I showed you that you can take a for loop and come to a while loop but you can check that the same thing will happen if I write any kind of a while loop which has an initialization and then a condition and then goes on to an update it will behave exactly like a for loop and the for loop will behave exactly like a while loop.

(Refer Slide Time: 13:30)


IIT Madras
BSc Degree

Iteration

- Intended use is
  `for(i = 0; i < n; i++){...}`
- Completely equivalent to

```
i = 0;
while (i < n) {
  i++;
}
```

- However, not good style to write `for` instead of `while`
- Can define loop variable within loop
  - The scope of `i` is local to the loop
  - An instance of more general local scoping allowed in Java

```
public class MyClass {
  ...
  public static int sumarray(int[] a) {
    int sum = 0;
    int n = a.length;
    for (int i = 0; i < n; i++){
      sum += a[i];
    }
    return(sum);
  }
}
```

So, this is a little bit strange and this is not really something that you should try to do. So, the intended use is this and you are expected to use for in this intended way. So, if you mean four you use for if you mean while you use while even though for and while are really very similar in expressiveness you do not confuse the 2. So, one more variation on this which is allowed in java is that.

So, this i that we are using here is it is very common to use a variable like this i inside a loop. So, if we when you come to this i here you really want to make sure that is being used for only one purpose which is to iterate over the positions in the array you do not want it outside. So, java actually allows you to move the definition here. So, you do not have any declaration here you just directly have a local declaration within a loop.

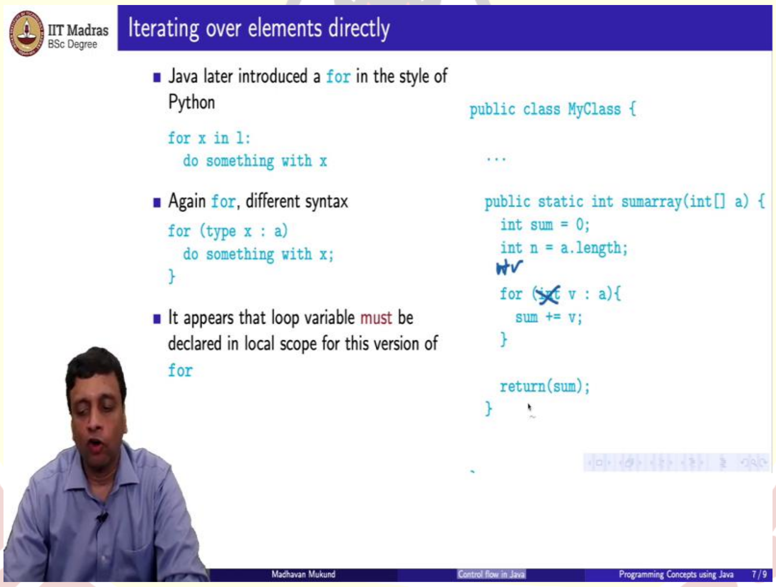
So, this means that outside here i is out of scope. So, i is not available here and i is not available here. So, this guarantees that you are not kind of mixing up things. So, sometimes it is a problem that you might have an i inside which is actually accidentally referring an i outside and there might be some confusion as to what the loop is doing. So, by declaring i within this for loop it actually defines the scope of this i to be this for.

So, this is actually not peculiar only to the for loop. So, java in general allows you to create a block anywhere in your code. So, normally the whole of your code has a block like this or like this. But you can create these nested local blocks and inside this you can write int j and then use j here and that j will not be available outside. So, you can have some local scoping in java we will not get into it too much because we will not need it.

But this particular instance of local scoping is important for us the fact that you can take a loop variable and declare it and use it inside the loop that means I can keep on using i everywhere with no fear that the i's are interfering with each other. Of course you cannot if I have a nested loop I cannot use the same i outside and inside. So, if I write for i outside and inside error at for i this will be confusion.

So, I have to write for j here. So, I cannot have a nested loop where the inside variable is the same as the outside variable but still if I declare the int j or the int i inside the for it makes it very clear that this i is used only inside here. So, the local scoping is preserved. So, this is a convenience and also a desirable feature to use. So, that you make sure that your loop variables are always used in this very specific way.

(Refer Slide Time: 16:01)



Iterating over elements directly

- Java later introduced a for in the style of Python

```
for x in l:
   do something with x
```

- Again for, different syntax

```
for (type x : a)
   do something with x;
}
```

- It appears that loop variable must be declared in local scope for this version of for

```
public class MyClass {

  ...

  public static int sumarray(int[] a) {
    int sum = 0;
    int n = a.length;
   int
    for (~~int~~ v : a){
      sum += v;
    }

    return(sum);
  }
```

So, this was the original for in java which is from the origins are actually in C. So, C has a for loop like that. So, java retained that for loop but as I said for is not intended to be used that way that is not how one normally thinks. So, for normally when think so, for s iterating over a sequence so, the sequence could be given in the form of a list or an array or it could just be a sequence of numbers generated by a function like range in python.

So, this is what you normally write in python for x and l do something with x. So, x goes from the first element of l to the second element of l to the last minute and each time you get an x you do something with it move to the next one. So, moving to the next one is implicit you do not have to say i equal to i + 1 for example. And it would be nice to have that and now java has introduced that.

May not now but it was not there in the original definition of java but obviously enough people felt it would be nice to have this version of for which is it is specific to this case but this is a very important case and you do not want to confuse this by running through the indices. So, it is much nicer to run through for x in l rather than say in python you would say for i in range length of l and then you talk about l i.

So, this is equivalent you can run i from 0 to n - 1 and talk about l i but it is so much nicer to just say for x and l if you are interested in the values. So the equivalent thing is there in java now. So, java has this kind of for it is again called for because they didn't want to add a new word to the language. So, this is a different version of four and it is very different it does not.

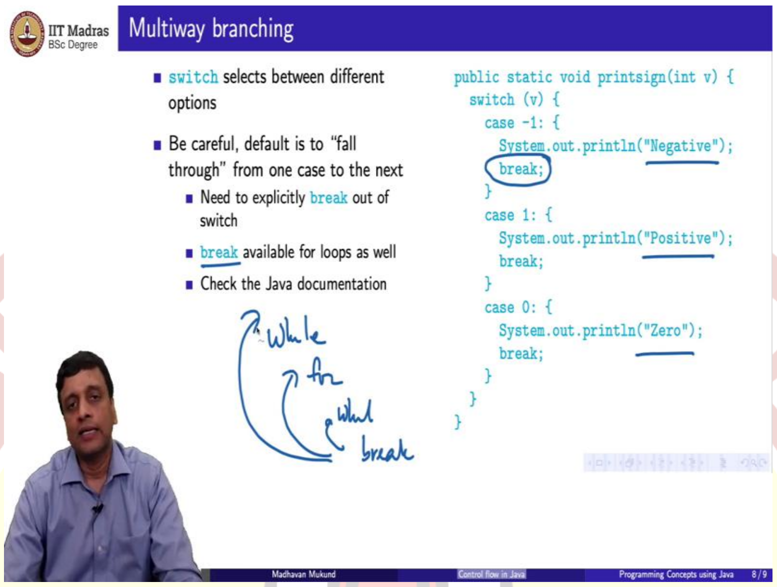
So, the previous one has these 2 semicolons the initialization semicolon the condition semicolon the update. So, in this for loop you have a colon first of all what is on the side is in this case an array but as we will see later on as we get into this later part of the course it can be any kind of a sequence. So, java has other types of sequences. So, any kind of a sequence and this is the variable that runs over that sequence.

So, what this is saying is this is another version of that sum array. So, it is saying let v range over a and keep adding v to sum. So, for every v if. So, it is just like saying for x and l sum is equal to sum + x. So, here I have said for v in a sum is equal to sum plus v. So, this is like a for each for each v in the sequence do something. Now there is one subtle point here which is that remember we said that this idea of local scoping is permitted by java.

So, you can take a loop and we can put the description or the definition of the loop variable inside the loop and therefore it becomes local to that loop and is not visible outside. Now it turns out that in this particular form of this loop you know this for each loop you might call it the one that goes over each element in a sequence this is not an option. So, actually if you do not put this here if you try to say int v here and then just use v here you will get a compiler error.

So, for this version of for java actually forces you to put this local declaration inside the loop otherwise it will not compile.

(Refer Slide Time: 19:08)



IIT Madras
BSc Degree

Multiway branching

- switch selects between different options
- Be careful, default is to "fall through" from one case to the next
  - Need to explicitly break out of switch
  - break available for loops as well
  - Check the Java documentation

```
public static void printsign(int v) {
  switch (v) {
    case -1: {
      System.out.println("Negative");
      break;
    }
    case 1: {
      System.out.println("Positive");
      break;
    }
    case 0: {
      System.out.println("Zero");
      break;
    }
  }
}
```

Madhavan Mukund Control flow in Java Programming Concepts using Java 8 / 9

So, we saw earlier that there is this if else if way of going through multiple conditions but quite often you have a situation where you have a variable which takes one of n values I mean you have all seen these kind of interactive voice menus you pick up the phone and it says press one if you want international press 2 if you want domestic and press three if you want something else.

So, you have these kinds of situations where you want to choose between a fixed number of alternatives and you could of course write it as if if v equal to one do something else it be equal to 2 something and so on. But there is a switch statement which is inherited again from c which allows you to do this switching between different options. So, here is the syntax of the switch statement.

So, I am here I have inverted that remember we wrote a sine function which took a number and returned back - 1 0 or + 1 depending on whether the input was negative 0 or positive now here is a function which takes the output of that. So, it takes as input a - 1 0 or 1 and tells you whether to interpret it is negative. So, it is just a diagnostics weight. So, if you get a if you get -1 here I want to print negative if you get +1 there I want to print positive if you get 0 then I want to print 0.

So, this is the statement. So, you write switch and then you have this case. So, these are the various cases. So, for each case this indicates the value that the case must match. So, if the value of v matches -1 it goes to that one if the value of v matches +1 goes to second one if it matches 0 it goes to the third one. And then you execute this line but there is one again peculiarity which is inherited from C there is no reason why it should be like this but this is inherent from C which is that it does not automatically come out of this case.

It is not like an if remember in an if you go into the if you will not go into the else either you go into this or you go into the edge you can you will not do both whereas in the switch statement if you go into case -1 and you do not say anything it will continue into next case. So, it will go into all succeeding cases and execute. So, it will only the case only describes the entry point.

So, for instance I will say one I will start here but then I will go all the way through so to get out of this situation you need to introduce explicitly this statement called break. So, break as the name suggests breaks out of the statement it says we are in some inside some nested block and get out of it and you have to break out of each case because the default is to fall through to the next case.

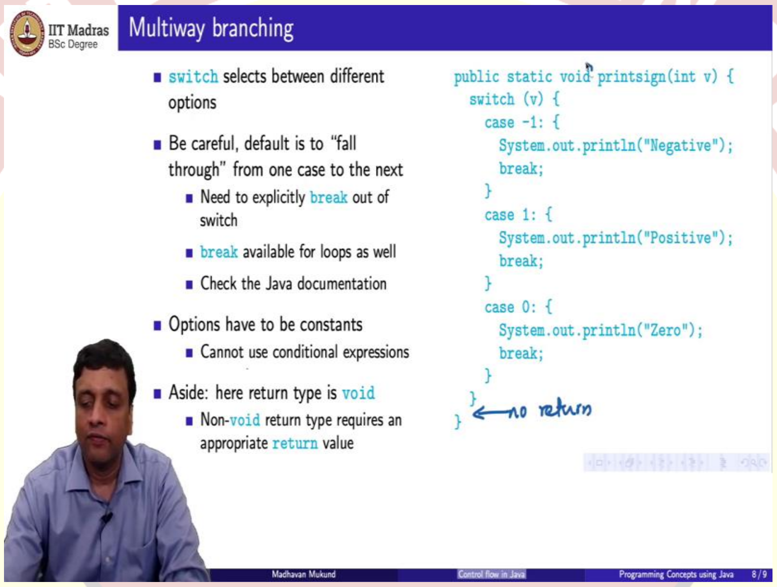
So, if I did not put breaks and if I gave -1 as an input to this function it will first print negative then print positive then print 0. If I gave it +1 it will start print positive then print 0 and if I give it 0 it will print only 0 it will not skip the subsequent cases. So, I have put a break. Now you can also use a break for loops. So, you can check the java documentation. So, python also has a break statement.

So, it is not very different only thing in java is you have what are called labeled breaks. So, you can describe because if I have a nested loop if I have a while then I have a for. And then I have a while then I have a break over here the question is do I want to get exit this loop or exit this loop or exit this loop I might want to get out of a whole sequence of loops.

So, you can put a label and say get out till that level you can break out multiple levels. So, you can look at the java documentation because break is not something very common that we are going to worry about but if you are writing practical code for instance if you are searching for an element in an array. So, if you are scanning an unsorted array from beginning to end the moment you find it you want to get out.

So, you can say break and get out. So, that is a typical situation where you want to break. So, exactly how you use break. So, break is just this statement but you can break and go out to a specific place if you are doing a break from inside a loop and that you can look it up.

(Refer Slide Time: 22:53)



Multiway branching

- switch selects between different options
- Be careful, default is to "fall through" from one case to the next
  - Need to explicitly break out of switch
  - break available for loops as well
  - Check the Java documentation
- Options have to be constants
  - Cannot use conditional expressions
- Aside: here return type is void
  - Non-void return type requires an appropriate return value

```
public static void printsign(int v) {
  switch (v) {
    case -1: {
      System.out.println("Negative");
      break;
    }
    case 1: {
      System.out.println("Positive");
      break;
    }
    case 0: {
      System.out.println("Zero");
      break;
    }
  }
}
```

← no return

Now one thing to remember in this switch statement is that I can only write constants here I cannot say case v less than 0 case v equal to 0 case v greater than 0. So, I cannot write conditional expressions. So, the switch is a very specific statement which allows me to choose between constant options. Now these constants can be numbers like this it also allows a string.

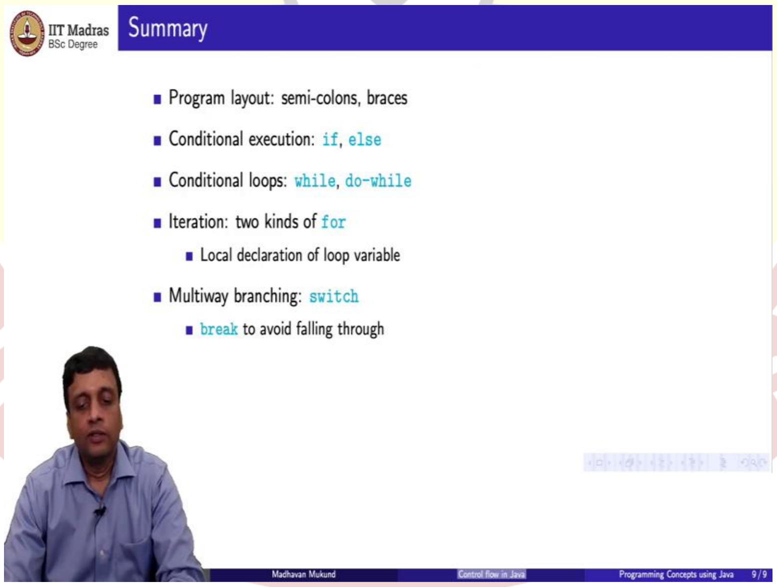
So, you can say if I have a response you can check case yes case no for example. So, these are but they have to be very specific values you cannot switch between expressions it is not like an multi way if conditions. If you want a multi wave condition you should use an if now one minor point just to tell you about how strict java is about a compiler. So, notice that there is no return here.

And the fact that there is no return is possible only because I have explicitly said that the return type of this function is void. For every function you must provide of course it could have a function with no inputs there is no reason why you must have an input but every function you must specify what it returns and if you do not expect it to return something you would better say void.

Because if you for example by mistake say int here then the compiler will insist on a return with some value of type int you cannot get away with just putting a return with no value. So, python is very flexible that way. So, at the end of a python function you can return with a value normally you can return without a value you can turn with different types of values at different places in the in the function or you could omit the return statement altogether it will just automatically exit at the end.

And because python does not expect anything to be known about the return value all of these are valid. Whereas in java you must give the return type and unless you specify the return type as void as I have done here you must provide a return statement with the correct type and the correct a definite value when you cut out. So, just be careful about that if you have a function which does nothing useful make its return type void.

(Refer Slide Time: 24:57)



Summary

- Program layout: semi-colons, braces
- Conditional execution: `if`, `else`
- Conditional loops: `while`, `do-while`
- Iteration: two kinds of `for`
  - Local declaration of loop variable
- Multiway branching: `switch`
  - `break` to avoid falling through

So, what we have seen now is that in java instead of the python's type indentation we use semicolons and braces to demarcate where statements end and where blocks begin and end if and else are similar to java similar to python the only thing is that we do not need an elif because we can write nested ifs without having to force the indentation across the page.

We have 2 types of conditional loop the normal while which tests at the top of the loop and a do while we test at the bottom of the loop. The bottom of the loop testing is much rarer to use but we would use it for instance we explained when you are taking input. If you are expecting the user to type something and the input terminates with some special character you need to read at least one line.

In such cases where you have to do at least one iteration do while is more natural than while do. For also there are 2 types there is the C style for which is very flexible and it is really a disguised while loop but you are intended to use it by setting a variable to be the initial value final value I mean test the condition for the final value and have an increment. But you can also write a form which runs through the elements of an array for example but then you must use local declaration.

So, for loops allow local declarations in general java allows you to locally scope a variable by creating a nested block. But in particular this nested block can be a for loop and if you are using this for v in a form then you must put a type here. You cannot use this for a variable which is not locally scoped within the following. And finally we saw that you can do a multi way branch on among a set of constant options like 0 1 2 or yes no or so on with a switch statement.

But the switch statement again inherits for no particular reason except that it is familiar to people it inherits the C option of falling through. So, each case automatically leads to the next case. So, you only get an entry point from the switch not an exit point. So, you must explicitly exit by using the break otherwise you fall through and you will unnecessarily execute multiple cases.

So, it is not like an if else in that sense it does not automatically choose only one segment it will choose a starting point and will keep going until you break.