**Introduction To Java - Day02**

**Read Input :**

Reading inputs can be done in different ways in Java, one of the classes that is used to read inputs from Keyboard is Scanner class .

Scanner is present in java.util package

Scanner class can be used to read the data from the keyboard.The syntax that is used to read data from the user through keyboard is.

**Scanner scan1 = new Scanner(System.in)**

**System.in** represents an Object containing a stream of data that flows into the application,The class that is responsible for this activity is InputStream, **in** is an Object of InputStream class

You will be learning more on STreams during the **“FileHandling”** session

Methods used to read the data through KeyBoard are

scan1.next() reads data as String without spaces,which means Scanner cannot read spaces when this function is used, which has the following syntax

**String employeeName = scan.next(); // accepting the string and storing it in employeeName**

Other functions include:

String employeeName = scan1.next() —------------->Reads data without spaces for eg. only Harish

String employeeFullName = scan1.nextLine() —------->Reads data with space for eg. Harish Kumar

int score = scan1.nextInt() —----- >Reads integer type of data for eg. 87

Double salary = scan1.nextDouble() —------> Reads double type of Data for eg. 9999.99

float tax = scan1.nextFloat() —--------> Reads float type of Data for eg. 23.34

For a complete list of functions please visit this link:

<https://docs.oracle.com/javase/7/docs/api/java/util/Scanner.html>

**Loops & Constructs** Contd…

Loops are the techniques used to repeat a task continuously as long as a condition is satisfied, for example, the Reservation Clerk at the counter would like to continue the process of reservation for the n number of times as many persons are in the queue.

In such circumstances, Looping techniques are used.

Major Looping Mechanisms available in Java are as follows:

‘ for Loop

‘ while Loop

‘ do while Loop

**For Loop :**

For Loop can be used in the situations, where we would like to repeat a particular activity for a specific number of times, which is known in advance.

The syntax to use **for loop** will be as follows:

for(initialization ; check condition ; operation)

{

Statements

}

**Initialization** takes place only once , whereas the **condition** will be checked, every time before entering the block of statements, if the condition is satisfied, it performs the functionality mentioned in the form of statements, else breaks out of the loop,

If the condition is satisfied, it performs the functionality and then computes the **operation** which could be either incrementation or decrementation.. After which , again the condition is checked and the functionality is executed based on the satisfaction of the condition and this process continues as long as the condition is satisfied.

Eg.

for(int i=0; i< 5 ; i++)

{

System.out.println(“The Step is “+ (i+1));

}

In the above snippet , int i = 0 is initialized only once

Then the condition is checked which is i < 5 ; 0 < 5 is satisfied, therefore it enters the statements block

Displays “The Step is 1” because i+1 is 1, but please note i is 0, i+1 is 1

After which the operation has to take place i.e i++ which is 0++ that makes i as 1

Condition gets checked after increment i.e i < 5 i.e 1 < 5 which is true, therefore it enters the statements block Displays “The Step is 2 “ i.e i+1 is 2

Therefore it displays

The Step is 1

The Step is 2

The Step is 3

The Step is 4

The Step is 5 on the Console

After this display, i gets incremented but the condition i<5 i.e 5<5 is false, so stops the loop

In loops , every set of statements’ execution within the block is called ITERATION.

‘**while Loop :**

**“**While” loops are also used when a particular task is to be executed n number of times , but this n might not necessarily be known in advance, as in the case of **for** loop. Here the condition need not be purely based on numerical values but it could be based on some conditions like - while ans == “yes” for example , where “ans” is a String variable.

But even the “for” loop kind of situation can be simulated using “while” loops.

The syntax for the while loop is, as given below:

while (conditional statement)

{

statements

}

The conditional statement is checked and if it is satisfied, executes the statements within the block and if the condition fails it breaks the loop.

Eg.

import java.util.Scanner;

public class WhileSample

{

String ans = “yes”

Int count = 0

Scanner scan = new Scanner(System.in);

while ( ans.equals(“yes”))

{

count++;

System.out.println(“ Loop “+count);

System.out.println(“Do You Wish to Continue yes/no”);

ans = scan.next();

}

}

In the above snippet, initially in the condition : ans.equals(“yes”)

equals method of String class is used to compare the string value present in **ans** and if it is same as **“yes”** the condition is satisfied and it enters the loop, count is incremented from 0 to 1

Then it displays “Loop 1”

After which it displays the text **“Do You Wish to Continue yes/no”** on the console , for which if the user enters “yes”, then loop continues again from the **conditional statement**, then the condition is satisfied and continues displaying “Loop 2”, “Loop 3” and so on as long user does not type something other than “yes”.

**Do..while** loop

. do… while loop works in the same way as that of the while loop, but in the while loop the condition is checked first and then based on the satisfaction of the condition, the statements are either executed or avoided, but in do..while loop it executes the statements without checking the condition at least once and then at the end, checks the condition and if it satisfies the condition, it continues the loops and so on.

Syntax for do..while loop is as follows

do

{

Statements

} while (condition);

Eg.

Import java.util.Scanner

public class DoWhiler

{

public static void main(String[] args)

{

Scanner scan = new Scanner(System.in);

String reply = “yes”;

int i = 1;

do

{  
 System.out.println(“ loop executed “ + i + ” time “;

i++;

System.out.println(“Do You wish to continue yes/no”);

reply = scan1.next();

} while (reply.equals(“yes”));

}

}

In the above example,

initially it prints the message **“loop executed 1 time”**, the **i** which is containing 1 is incremented to 2, then it prints the message **“Do You wish to continue yes/no”** , accepts the reply from the user into the variable **“reply” ,** then the condition **reply.equals(“yes”)** is checked, if it is satisfied, then the loop continues back to **do block** and continues to print

**loop executed x times** x being incremented value of i , as long as the user does not key in anything other than **yes**.

**continue & break**

**“**continue” and “break” are 2 clauses which are handy in controlling the loops, for eg. in the following snippet,

‘ int square;

‘ int number;

Scanner scan = new Scanner(System.in);

String reply = “yes”;

while( reply.equals(“yes”))

{

System.out.println(“Enter a Number”);

number = scan.nextInt();

If (number > 100)

{

System.out.println(“Sorry Number cant be greater than 100”);

Continue; // or break

}

square = number \* number;

System.out.println(“The Square of the Number “+number+” is “+square);

System.out. println(“Do You wish to continue yes/no”);

reply = scan.next();

}

The user can enter a number and find the square of the number as long as he enters “yes” when asked “Do You wish to continue yes/no”, but once he enters the number for which square is to be found, it checks if the number entered is > 100, if so, it prints a message “Sorry Number cant be greater than 100” and goes to the beginning of the loop , skipping all the subsequent lines’ execution and gives the chance to keep entering numbers as long as the user enters a number < 100.

That is the usage of continue, instead if one uses **“break”** when the user enters a number > 100 it breaks out of the loop, without giving the user the option to continue.

—------------------------------------------------------