DeepSeek Chat

```
//Lecture 18

Iterative Statements , WHile , Do While , For, Nested Loops ...

SYNTAX:
//Test condition must be boolean in every case always always always
while (test_cond ){
    ...
}

in C valid but java not valid :
while (1){
    ...
}

In java for infinite loops :
//this works in java this is the way
while (true ){
    ....
}
//runs upto 10
class WhileDemo {

    public static void main(String [] args ){
        int i = 1;
        while (i<=10){
            System .out .println (i);
            i++;
        }
    }
}
//runs infinitely stop it using ctrl C
```

```
//runs infinitely stop it using ctrl C.
class WhileDemo {

    public static void main(String[] args){
        int i = 1;
        while(true){
            System.out.println(i);
            i++;
        }
    }
}

//break can also be used in while loops as well.
class WhileDemo {

    public static void main(String[] args){
        int i = 1;
        while(true){
            System.out.println(i);
            i++;
            if(i==10){
                break;
            }
        }
    }
}

//WAP take in a number by user and give the factorial of that.

//Initial dumbass code see what you thought of later now...
import java.util.Scanner;

class WhileDemo {

    public static void main(String[] args){
        int i;
```

```
        Scanner scan = new Scanner(System.in);
        i = scan.nextInt();
        int number = 1, count = 1;
        while(count<=i){
            number*=number;
            count++;
        }
        System.out.println(number);
    }
}

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        int i;
        Scanner scan = new Scanner(System.in);
        int i = scan.nextInt();
        int sum = 1;
        if(i==0){
            System.out.println(1);
        }
        while(i>0){
            sum*=i; //condition was so simple lol
            i--;
        }
        System.out.println(sum);
    }
}

SYNTAX OF FOR Loops
======================

for(int ; test cond ; stmt){
    ...
```

```
}
```

Point to remember about for loop:

in C++ Turbo

```
for(int i = 1; i<=10; i++){

    cout<<i<<endl;
}
cout << "out of loop i ="<<i;
```

gives no error.

GThe output is 11in Turbo C++ IDE

In java

==========

```
for(int i = 1; i<=10; i++){
    SOP(i);
}
SOP("Out of the loop i = " + i);
```

This gives an ERRRORin JAVA -> variable initiralized inside the for loop are local to the for loop.

Gives error cannot find symbol.

If you want to use it outside of for, you gotta declare it above the for loop.

you can initialize it inside but yea gotta declare outside the for loop.

2. In C

==========

```
int i, j;

for(i = 1, j = 10; i<=10, j>=5; i++, j++){
    printf ("\n%d",i);
}
```

The loop will ONLY WORK FOR THE 2ND condition , the last condition , other

conditions are ignored !

BUT NOT IN JAVA , this gives ERROR

YOU CAN WRITE MULTIPLE CONDITIONS BUT MUST USE LOGICAL OPERATORSIN THAT .

```
for (i = 1, j = 10; i<=10 || j>=5; i++, j++){
    System .out .println (i + " " + j);
}

//THis works therefore . But gives error

import java.util .Scanner ;

class WhileDemo {

    public static void main (String [] args ){
        Scanner scan = new Scanner (System .in);
        int i = scan .nextInt ();
        for (int fact = 1; i>1; i--){
            fact *=i; //works just fine , however fact is not accessible outside the
loop so gives error
        }
        System .out .println (fact );
    }
}

import java.util .Scanner ;

class WhileDemo {

    public static void main (String [] args ){
        Scanner scan = new Scanner (System .in);
        int i = scan .nextInt ();
        int fact ;
```

```
        int fact;
        for(fact = 1; i>1; i--){
            fact*=i; //works just fin, fact was declared outside so no errors.
        }
        System.out.println(fact);
    }
}

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int i = scan.nextInt();
        int fact;
        for(fact = 1; i>1; i--){
            fact*=i; //works just fin, fact was declared outside so no errors.
        }
        System.out.println(fact);
    }
}

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int i = scan.nextInt();
        int fact = 1;
        for(fact; i>1; i--){
            fact*=i; //also works just fine, no need to initialize inside for loop, can
be outside.
        }
        System.out.println(fact);
    }
```

```
    }
}

import java.util.Scanner;

class WhileDemo {

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int i = scan.nextInt();
        int fact = 1;
        for(fact; i>1; ){
            fact*=i; //also works just fine, you can even do ++, -- inside the loops.
            i--; // like here
        }
        System.out.println(fact);
    }
}

//Also works fine the empty for loop (;;){}, runs infinitely unless broken
though ...
//SO INSIDE FOR, everything is optional !

import java.util.Scanner;

class WhileDemo {

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        int i = scan.nextInt();
        int fact = 1;
        for( ; ; ){
            if(i<=1){
                break;
            }
            fact*=i; //also works just fine, you can even do ++, -- inside the loops.
```

```
            i--; // like here
        }
        System.out.println(fact);
    }
}

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        for(int i = 0 i <= 10; ; i++ ); // This also works, Java Assumes the Loop
body is empty.
        System.out.println(i); // i prints 11
    }
}
```

Do WHile

==========

Syntax:

```
do{
....
....
}while(); //semicolon last important btw.
```

//WAP to accept 2 integers from the user and display their sum, now ask the user
whether he or she wants to continue or not, if the answer is Yes, then again repeat,
otherwise terminate the program displaying the message "thank you user"

```
//My Solution
```

```
import java.util.Scanner;

class WhileDemo {

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);

        while(true){
                System.out.println("Enter 2 Integers :");
                int one = scan.nextInt();
                int two = scan.nextInt();
                System.out.println("The Sum is: " + (one+two));
                System.out.print("Run this Program Again?(Y/N): ");
                char yesOrNo = scan.next().charAt(0);
                if(yesOrNo == 'Y' || yesOrNo == 'y'){

                }else{
                        break;
                }
        }
    }
}

//my solution 2 more compressed using continue and break

import java.util.Scanner;

class WhileDemo {

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);

        while(true){
                System.out.println("Enter 2 Integers :");
                int one = scan.nextInt();
                int two = scan.nextInt();
```

```
            System.out.println("The Sum is: " + (one+two));
            System.out.print("Run this Program Again?(Y/N): ");
            char yesOrNo = scan.next().charAt(0);
            if(yesOrNo == 'Y' || yesOrNo == 'y')
        continue ; // works for if since its a single statement
        break; // works for the while pretty cool
        }
    }
}

//Solution 3 it needs to be done using do while and also it should be Yes or No not Y or N

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String yes;

        do{
            System.out.println("Enter 2 Integers : ");
            int a = scan.nextInt();
            int b = scan.nextInt();
            System.out.println("Sum is " + (a+b));
            System.out.println("Do you want to continue ? Type \"Yes\" if you do without any quotes : ");
            yes = scan.next();
        }while(yes.equals("Yes"));
    }
}

//This is wrong , ends the program because of nextLine() But why? nextLine scans whole strings with spaces
//There is a problem . When asking 2 numbers , the user strikes enter after the
```

```
//There is a problem : When asking 2 numbers, the user enters enter after the
2 numbers are asked and done
//the memory took in int 1 int 2 and also took ENTER inside the memory buffer
//nextLine() sees the Enter as another input and scans it, it won't wait for the
user input
//The solution is to call nextLine twice.
//Before the scanning of the string, take in input as the nextLine() above it. So
the enter is taken
// and now the buffer has space for the string thats the solution

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String yes;

        do{
                System.out.println("Enter 2 Integers : ");
                int a = scan.nextInt();
                int b = scan.nextInt();
                System.out.println("Sum is " + (a+b));
                System.out.println("Do you want to continue ? Type \"Yes\" if you do
without any quotes : ");
                yes = scan.nextLine();
        }while(yes.equals("Yes"));
    }
}

//SOLVED nextLine() buffer problem

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
```

```
    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String yes;

        do{
            System.out.println("Enter 2 Integers : ");
            int a = scan.nextInt();
            int b = scan.nextInt();
            System.out.println("Sum is " + (a+b));
            System.out.println("Do you want to continue ? Type \"Yes\" if you do
without any quotes : ");
        scan.nextLine();//This one takes the enter away as the input from buffer
            yes = scan.nextLine(); // This one now takes in the String for the
input via nextLine() * * * !VVIMP
        }while(yes.equals("Yes"));
    }
}



//This also won't work since it compares addresses

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        String yes;

        do{
            System.out.println("Enter 2 Integers : ");
            int a = scan.nextInt();
            int b = scan.nextInt();
            System.out.println("Sum is " + (a+b));
            System.out.println("Do you want to continue ? Type \"Yes\" if you do
```

```
without any quotes : ");
            yes = scan.next(); // input taken correctly
        }while(yes=="Yes");
    }
}

//This also won't work correctly since you cannot declare the variable inside because then its
a local variable of the do while loop body you gotta declare the condition variable outside.

import java.util.Scanner;

class WhileDemo{

    public static void main(String[] args){
        Scanner scan = new Scanner(System.in);
        do{
            System.out.println("Enter 2 Integers : ");
            int a = scan.nextInt();
            int b = scan.nextInt();
            System.out.println("Sum is " + (a+b));
            System.out.println("Do you want to continue ? Type \"Yes\" if you do
without any quotes : ");
            String yes = scan.next();
        }while(yes.equals("Yes"));
    }

Nested Loop
===========
//THis will break out of only once in the nested loops. Q.What if I wanna
break out of all?
for(init; test; stmt){
    for(init; test; stmt){
        for(init; test; stmt){
            ....
```

```
            //this only breaks out to the for loop outside this one. 1 nest broken
            if(testcond ){
                break;
            }

        }
    }
}

//JAVA has LABELS for this. you can break out to the label.
Labels can have any name just not predefined  java classes, keywords ,
methods , etc.



//works  like GOTO but java has no goto  so we use labels, either  labelled
break  or labelled  continue .

bhopal :
for(init; test; stmt){
    for(init; test; stmt){
        for(init; test; stmt){
            ....
            //this only breaks out to the for loop outside this one. 1 nest broken
            if(testcond ){
                break bhopal; // This makes the control  flow  go out of all the loops
and runs down  the program  normally
            }

        }
    }
}
//break  bhopal  sent us here.

bhopal :
//continue  bhopal  led us here
for(init; test; stmt){
```

```
    for(init; test; stmt){
        for(init; test; stmt){
            ....
            //this only breaks out to the for loop outside this one. 1 nest broken
            if(testcond ){
                continue bhopal; // This makes the control flow go to the code
under bhopal, this means at the start
                //of the outermost for loop
            }

        }
    }
}

//Bonus you can also initialize a string into var and then use method,
//or you can directly use it on a string like this
//System.out.println("Sachin".length());

//NEXT WE WILL USE NETBEANS AND ECLIPSE IDES in JAVA
```

now I want it for lecture 18. This is my practice for Lecture 18 in Java, I want you to make me simple notes based on these notes at first. second I want you to collect a list of questions and answers that I can put into kahoot to practice the answers and not forget the quirks of Lecture 18 in Java.
I want you to not miss out on including any code that I have written by the way, mention it all, with relevency

Simple Notes for Lecture 18: Iterative Statements (Loops)

Iterative Statements in Java

Iterative statements (loops) are used to execute a block of code repeatedly based on a condition. Java supports three types of loops:

1. `while` Loop

2. `do-while` Loop
3. `for` Loop

---

1. `while` Loop

- **Syntax**:

```java
while (condition) {
    // Code to execute
}
```

- **Key Points**:
  - The condition must be a **boolean expression**.
  - If the condition is `true`, the loop body is executed.
  - If the condition is `false`, the loop terminates.
- **Example**:

```java
int i = 1;
while (i <= 10) {
    System.out.println(i);
    i++;
}
```

Infinite `while` Loop

- Use `while(true)` for an infinite loop.
- Example:

```java
while (true) {
    System.out.println("Infinite Loop");
}
```

Breaking a Loop

- Use the `break` statement to exit a loop.
- Example:

```java
```

```java
int i = 1;
while (true) {
    System.out.println(i);
    i++;
    if (i == 10) {
        break;
    }
}
```

---

2. `for` Loop

- **Syntax**:

```java
for (initialization; condition; update) {
    // Code to execute
}
```

- **Key Points**:
  - Variables declared in the initialization block are **local to the loop**.
  - All parts of the `for` loop are optional (e.g., `for(;;)` is valid).
- **Example**:

```java
for (int i = 1; i <= 10; i++) {
    System.out.println(i);
}
```

Multiple Conditions in `for` Loop

- Use logical operators (`&&`, `||`) for multiple conditions.
- Example:

```java
for (int i = 1, j = 10; i <= 10 || j >= 5; i++, j--) {
    System.out.println(i + " " + j);
}
```

Factorial Using `for` Loop

- Example:

```java
```

```java
Scanner scan = new Scanner(System.in);
int num = scan.nextInt();
int fact = 1;
for (; num > 1; num--) {
    fact *= num;
}
System.out.println("Factorial: " + fact);
```

---

3. `do-while` Loop

- **Syntax**:

```java
do {
    // Code to execute
} while (condition);
```

- **Key Points**:
  - The loop body is executed **at least once**.
  - The condition is checked after the loop body.
- **Example**:

```java
int i = 1;
do {
    System.out.println(i);
    i++;
} while (i <= 10);
```

Sum of Two Numbers with User Choice

- Example:

```java
Scanner scan = new Scanner(System.in);
String choice;
do {
    System.out.println("Enter two numbers:");
    int a = scan.nextInt();
    int b = scan.nextInt();
    System.out.println("Sum: " + (a + b));
    System.out.println("Do you want to continue? (Yes/No)");
    scan.nextLine(); // Clear buffer
```

```
    choice = scan.nextLine();
} while (choice.equalsIgnoreCase("Yes"));
```

Nested Loops

- A loop inside another loop is called a **nested loop**.
- **Example**:

```java
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        System.out.println(i + " " + j);
    }
}
```

Breaking Out of Nested Loops

- Use **labels** to break out of multiple loops.
- Example:

```java
outerLoop:
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        if (i == 2 && j == 2) {
            break outerLoop; // Breaks out of both loops
        }
        System.out.println(i + " " + j);
    }
}
```

Common Mistakes and Fixes

1. `nextLine()` **Issue**:
    - When using `nextInt()` followed by `nextLine()`, the `nextLine()` may consume the newline character.
    - Fix: Add an extra `nextLine()` to clear the buffer.
    - Example:

```java
```

```
Scanner scan = new Scanner(System.in);
int num = scan.nextInt();
scan.nextLine(); // Clear buffer
String str = scan.nextLine();
```

2. **Comparing Strings**:
   - Use `.equals()` instead of `==` to compare strings.
   - Example:

```java
String choice = scan.next();
if (choice.equals("Yes")) {
    // Code
}
```

Kahoot Questions and Answers

`while` Loop

1. **Q**: What is the output of the following code?

```java
int i = 1;
while (i <= 5) {
    System.out.println(i);
    i++;
}
```

   - **A**: `1 2 3 4 5`.

2. **Q**: How do you create an infinite `while` loop?
   - **A**: `while(true)`.
3. **Q**: What is the purpose of the `break` statement in a loop?
   - **A**: To exit the loop immediately.

`for` Loop

4. **Q**: What is the output of the following code?

```java
for (int i = 0; i < 3; i++) {
    System.out.println(i);
```

```
}
```

- **A**: `0 1 2`.

5. **Q**: What happens if you omit the condition in a `for` loop?
   - **A**: The loop runs infinitely.
6. **Q**: What is the output of the following code?

```java
for (int i = 1; i <= 3; i++) {
    for (int j = 1; j <= 3; j++) {
        System.out.println(i + " " + j);
    }
}
```

   - **A**: Prints all combinations of `i` and `j` from `1 1` to `3 3`.

`do-while` Loop

7. **Q**: What is the output of the following code?

```java
int i = 1;
do {
    System.out.println(i);
    i++;
} while (i <= 3);
```

   - **A**: `1 2 3`.
8. **Q**: How many times is the body of a `do-while` loop executed?
   - **A**: At least once.

Nested Loops

9. **Q**: What is the output of the following code?

```java
outerLoop:
for (int i = 1; i <= 2; i++) {
    for (int j = 1; j <= 2; j++) {
        if (i == 2 && j == 2) {
            break outerLoop;
        }
```

```
            System.out.println(i + " " + j);
        }
    }
```

- **A**: `1 1`, `1 2`, `2 1`.

10. **Q**: What is the purpose of a label in nested loops?
    - **A**: To break out of multiple loops.

General Questions

11. **Q**: What is the output of the following code?

```java
for (int i = 1; i <= 5; i++) {
    if (i == 3) {
        continue;
    }
    System.out.println(i);
}
```

- **A**: `1 2 4 5`.

12. **Q**: What is the output of the following code?

```java
int i = 1;
while (i <= 3) {
    System.out.println(i);
    i++;
}
```

- **A**: `1 2 3`.

13. **Q**: What is the output of the following code?

```java
int i = 1;
do {
    System.out.println(i);
    i++;
} while (i <= 0);
```

- **A**: `1`.

14. **Q**: What is the output of the following code?

```java
for (int i = 1; i <= 3; i++) {
    if (i == 2) {
        break;
    }
    System.out.println(i);
}
```

- **A**: `1`.

15. **Q**: What is the output of the following code?

```java
for (int i = 1; i <= 3; i++) {
    if (i == 2) {
        continue;
    }
    System.out.println(i);
}
```

- **A**: `1 3`.

These notes and questions are designed to help you understand and retain the concepts of iterative statements in Java. Use the Kahoot questions to test your knowledge! 😊