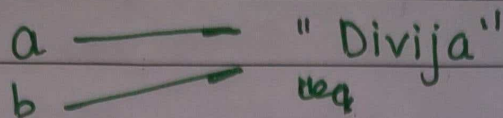


* LECTURE 5 SWITCH CASES / NESTED CASE

if :-



ie a, b are both pointing towards same object

$a == b \Rightarrow \text{TRUE}$

if :-

a — "Divija"

b — "Divija"

ie a, b are pointing to diff. objects of same value

$a == b \Rightarrow \text{FALSE}$

Thus, to compare strings we use .equals.

NOTE :-

- cases have to be the same type as expressions

- must be a constant or literal

- duplicate case values are not allowed

- break is used to terminate the sequence

- if break is not used, it will continue to next

- default will execute when no cond. is satisfied

- if default is not at the end, but break after it.

(*)

Switch Cases

In switch statements, you can jump to various cases based on your expression.

SYNTAX

```
switch (expression) {
```

```
    // cases
```

```
    case one:
```

```
        // do something
```

```
        break;
```

```
    case two:
```

```
        // do something
```

```
        break;
```

```
    default:
```

```
        // do something
```

```
}
```

NOTE :-

- cases have to be the same type as expressions, must be a constant or literal
- duplicate case values are not allowed.
- break is used to terminate the sequence
- if break is not used, it will continue to next
- default will execute when no cond. is satisfied
- if default is not ~~xx~~ at the end, put break after it

ENHANCED SWITCH SYNTAX :-

```
switch (expression) {  
    case one → do this ;  
    case two → do this ;  
    case three → do this ;  
    default → do this ;  
}
```

* Nested switch case.

switch inside switch.

```
switch (exp) {  
    case one :  
        switch (exp2) {  
            do this  
        }  
        break ;  
    case two :  
        do this ;  
        break ;  
    default :  
        do this ;  
        break ;  
}
```