# Domain 3: Flow Control Implementation Videos

## If - Else If - Else

```
- if (statement) {
      block;
  } else if (statement 2) {
      block;
  } else {
      block;
  }
```

## Terinary

```
statement ? true : false;
```

## Logical Operators

&& → and

|| → or

## Relational Operators

== , != , >= , <= , > , <

## Switch

```
- switch (var) {
      case 1:
        block;
        break;
      case 2:
        block;
        break;
      default:
        block;
        break;
  }
```

If there werent breaks and case 2 was true, then it would be a diff case. ALL cases below the true case will occur for example case 2 and default would happen.

- Modern v.s. Traditional

OLD

```
switch (op) {
  case "+":
    SOP (v1 + v2);
    break;
  case "-":
    SOP (v1 - v2);
    break;
  default:
  } SOP (v1 * v2);
    break;
}
```

NEW

```
switch (op) {
    case "+" -> SOP (v1 + v2);
    case "-" -> {
        SOP (v1 - v2);
        ____________
        ________
    }
    default -> SOP(v1 * v2);
}
```

## Loops

### While

```
- while (statement){
      block;
  }
```

### Do-While

- a do-while loop is a type of loop that runs like a while loop, however it will run atleast once
- ex.

```
do {
    block;
} while (statement);
```

### For

- runs a certain amount of times
- essentially a speacil while loop
- ex.

```
                                        → ex. var++;
for (type var = init; statement; change in var){
    block;
}
```

- For-Each

ex.

```
type [] arrName = new type[];

for( type var: arrName){
    block;
}
```

### Break/Continue

break → exit the loop

continue → acts like a skip

## Unary Operators

ex.

SOP (x++); { print → change by +1

SOP (++x); } change by +1 → print

- ofc, +1 could be anything