Lecture 2

Part A

Selections -
Motivation of Conditionals

Why Selective Actions

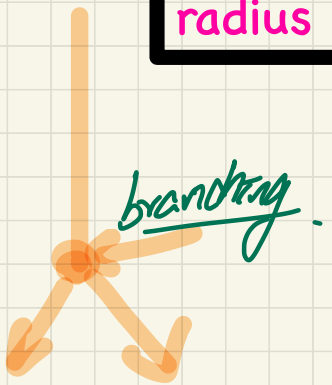
Test Inputs:

radius = 3

radius = -3

```
1  import java.util.Scanner;
2  public class ComputeArea {
3    public static void main(String[] args) {
4      Scanner input = new Scanner(System.in);
5      System.out.println("Enter the radius of a circle:");
6      double radiusFromUser = input.nextDouble();
7      final double PI = 3.14;
8      double area = radiusFromUser * radiusFromUser * PI;
9      System.out.print("Circle with radius " + radiusFromUser);
10     System.out.println(" has an area of " + area);
11     input.close();
12   }
13 }
```

executed despite that input radius < 0.



in this case, an alternative block of code should be executed.

If the user enters a positive radius value as expected:

```
Enter the radius of a circle:
3
Circle with radius 3.0 has an area of 28.26
```

However, if the user enters a negative radius value:

```
Enter the radius of a circle:
-3
Circle with radius -3.0 has an area of 28.26
```

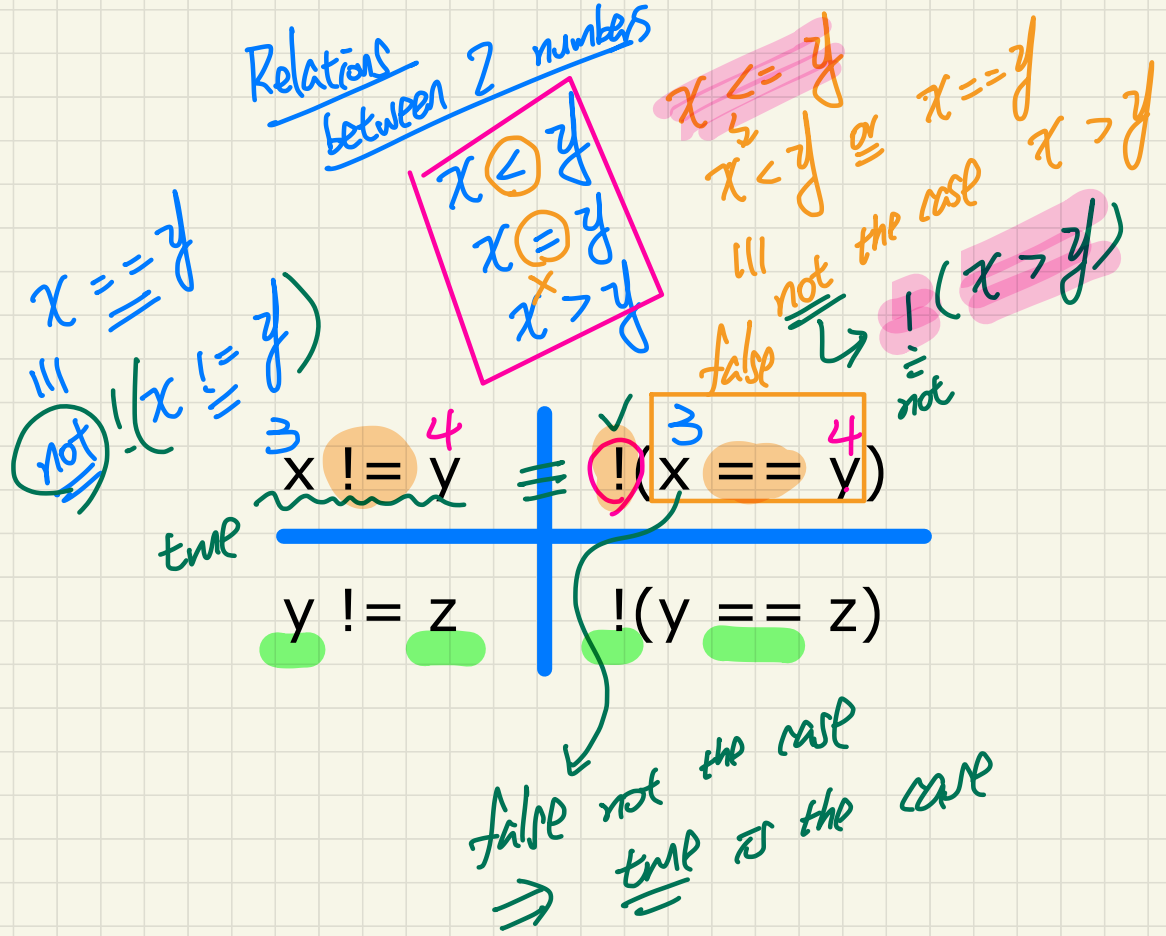
Lecture 2

Part B

Selections - Boolean Data Type

Not Equal To

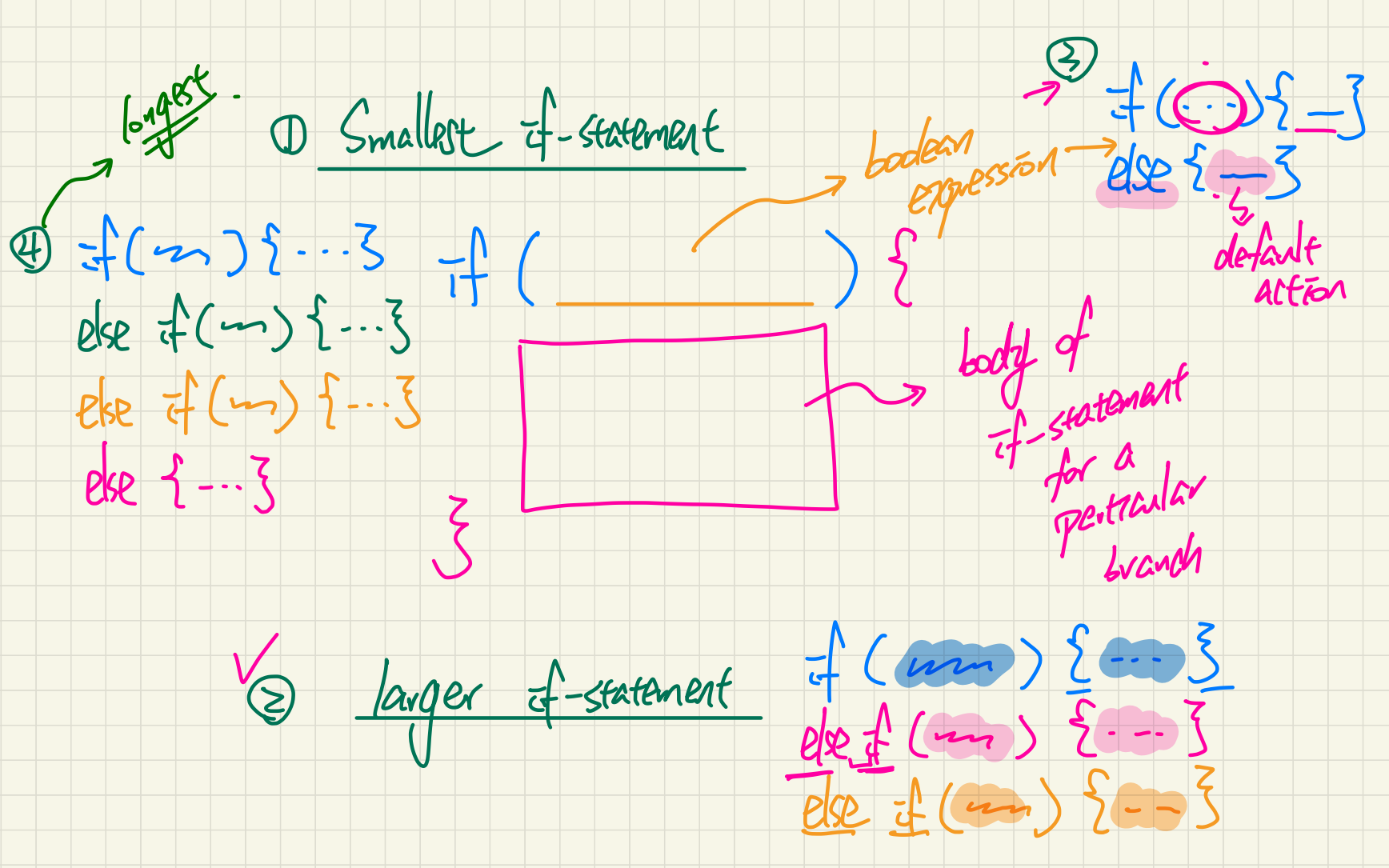
```
int x = 3;
int y = 4;
int z = 4;
```



Lecture 2

Part C

Selections - If-Statement: Syntax and Semantics

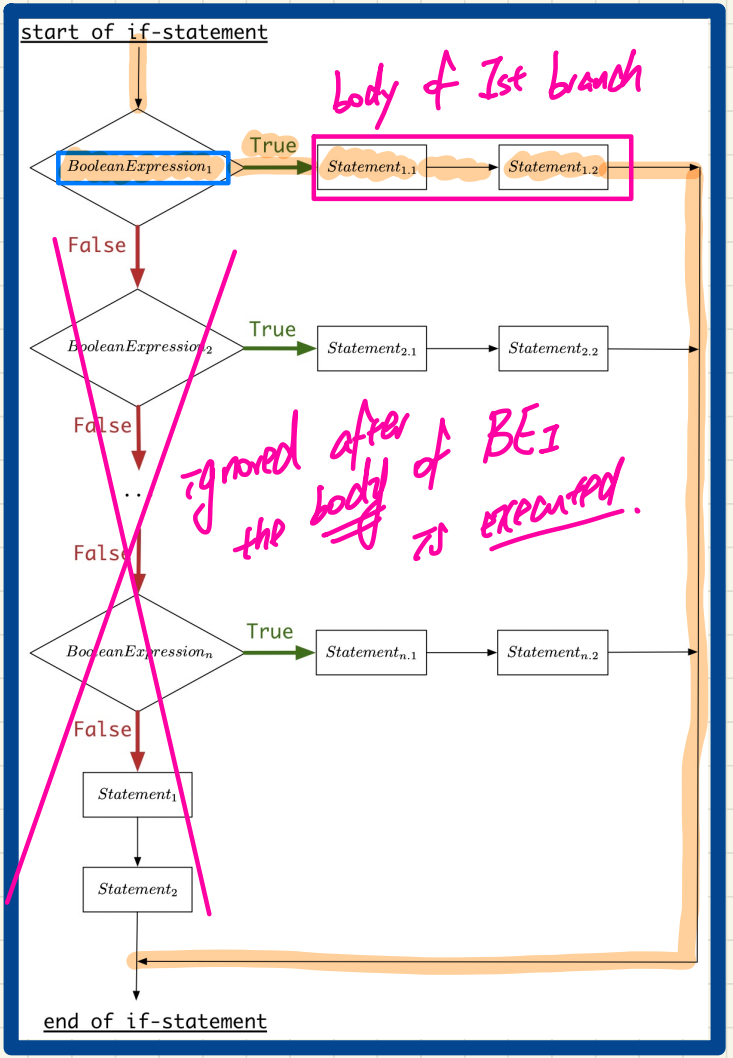

longest
① Smallest if-statement
boolean expression
if ( ) {
}
body of if-statement for a particular branch
③
if ( ) { }
else { }
default action
④
if ( ) { ... }
else if ( ) { ... }
else if ( ) { ... }
else { ... }
② Larger if-statement
if ( ) { }
else if ( ) { }
else if ( ) { }

A Single If-Statement

Syntax

```
if ( BooleanExpression1 ) { /* Mandatory */
  Statement1.1; Statement2.1;
}
else if ( BooleanExpression2 ) { /* Optional */
  Statement2.1; Statement2.2;
}
... /* as many else-if branches as you like */
else if ( BooleanExpressionn ) { /* Optional */
  Statementn.1; Statementn.2;
}
else { /* Optional */
  /* when all previous branching conditions are
  Statement1; Statement2;
}
```

Semantics/ Meaning



Case 1

BooleanExpression1 evaluares to **true**

If-Statement Case 1: Example

Only **first** satisfying branch *executed*; later branches *ignored*.

```
int i = -4;
if(i < 0) {
  System.out.println("i is negative");
}
else if(i < 10) {
  System.out.println("i is less than than 10");
}
else if(i == 10) {
  System.out.println("i is equal to 10");
}
else {
  System.out.println("i is greater than 10");
}
```

-4 < 0. (T)

ignored/bypassed.

Console

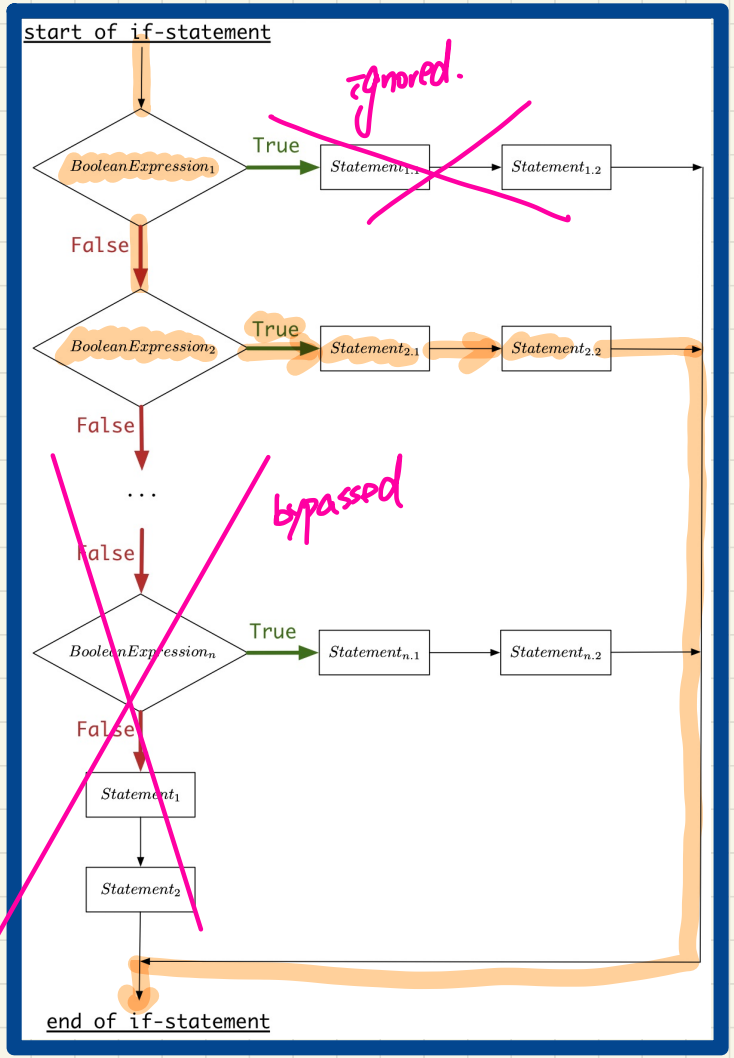
```
i is negative
```

A Single If-Statement

Syntax

```
if ( BooleanExpression1 ) {  /* Mandatory */
  Statement1.1; Statement2.1;
}
else if ( BooleanExpression2 ) {  /* Optional */
  Statement2.1; Statement2.2;
}
... /* as many else-if branches as you like */
else if ( BooleanExpressionn ) {  /* Optional */
  Statementn.1; Statementn.2;
}
else {  /* Optional */
  /* when all previous branching conditions are
  Statement1; Statement2;
}
```

Semantics/ Meaning



Case 2

BooleanExpression1 evaluares to **false**

BooleanExpression2 evaluares to **true**

If-Statement Case 2: Example

Only **first** satisfying branch *executed*; later branches *ignored*.

```
int i = 5;
if(i < 0) {
  System.out.println("i is negative");
}
else if(i < 10) {
  System.out.println("i is less than 10");
}
else if(i == 10) {
  System.out.println("i is equal to 10");
}
else {
  System.out.println("i is greater than 10");
}
```

5 < 0 (F)

5 < 10 (T).

bypassed.

Console

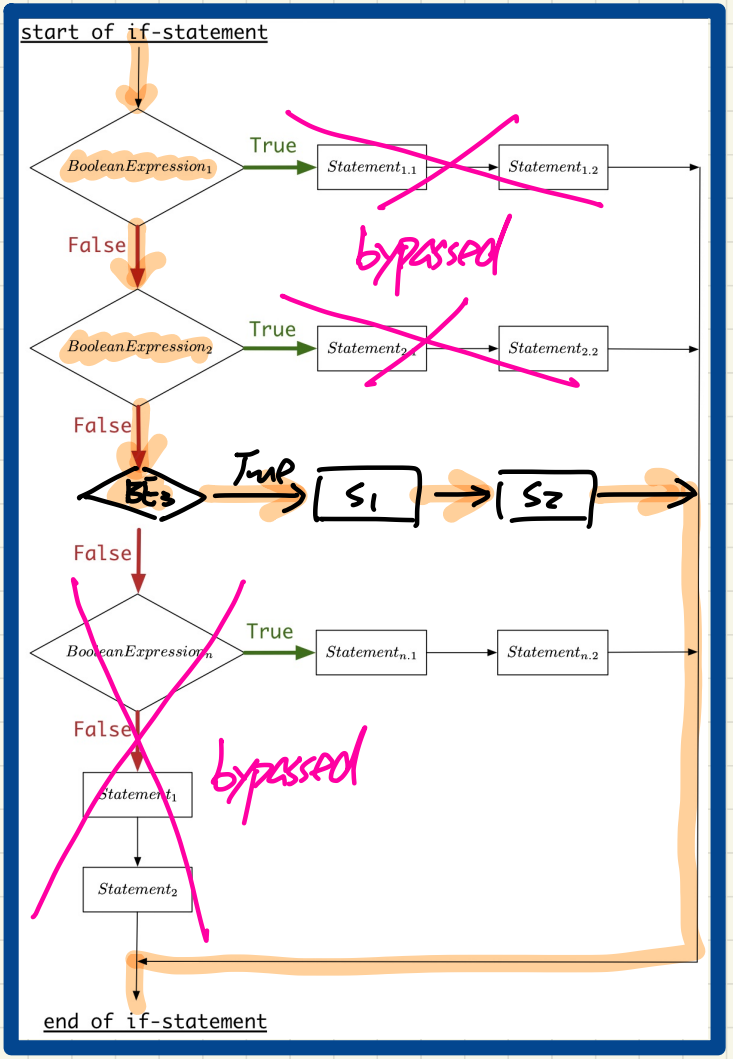
```
i is less than 10
```

A Single If-Statement

Syntax

```
if ( BooleanExpression1 ) {  /* Mandatory */
  Statement1.1; Statement2.1;
}
else if ( BooleanExpression2 ) {  /* Optional */
  Statement2.1; Statement2.2;
}
... /* as many else-if branches as you like */
else if ( BooleanExpressionn ) {  /* Optional */
  Statementn.1; Statementn.2;
}
else {  /* Optional */
  /* when all previous branching conditions are
  Statement1; Statement2;
}
```

Semantics/ Meaning



Case 3

BooleanExpression1 evaluates to **false**

BooleanExpression2 evaluates to **false**

BooleanExpression3 evaluates to **true**

If-Statement Case 3: Example

Only **first** satisfying branch *executed*; later branches *ignored*.

```
int i = 10;
if(i < 0) {
System.out.println("i is negative");
}
else if(i < 10) {
System.out.println("i is less than than 10");
}
else if(i == 10) {
System.out.println("i is equal to 10");
}
else {
System.out.println("i is greater than 10");
}
```

10 < 0 (F)

10 < 10 (F)

10 == 10 (T)

bypassed.

Exercise: Run debugger on Eclipse for Case 3!

Console

```
I is equal to 10
```

A Single If-Statement

Syntax

```
if ( BooleanExpression1 ) {  /* Mandatory */
  Statement1.1; Statement2.1;
}
else if ( BooleanExpression2 ) {  /* Optional */
  Statement2.1; Statement2.2;
}
... /* as many else-if branches as you like */
else if ( BooleanExpressionn ) {  /* Optional */
  Statementn.1; Statementn.2;
}
else {  /* Optional */
  /* when all previous branching conditions are
  Statement1; Statement2;
}
```

Semantics/ Meaning

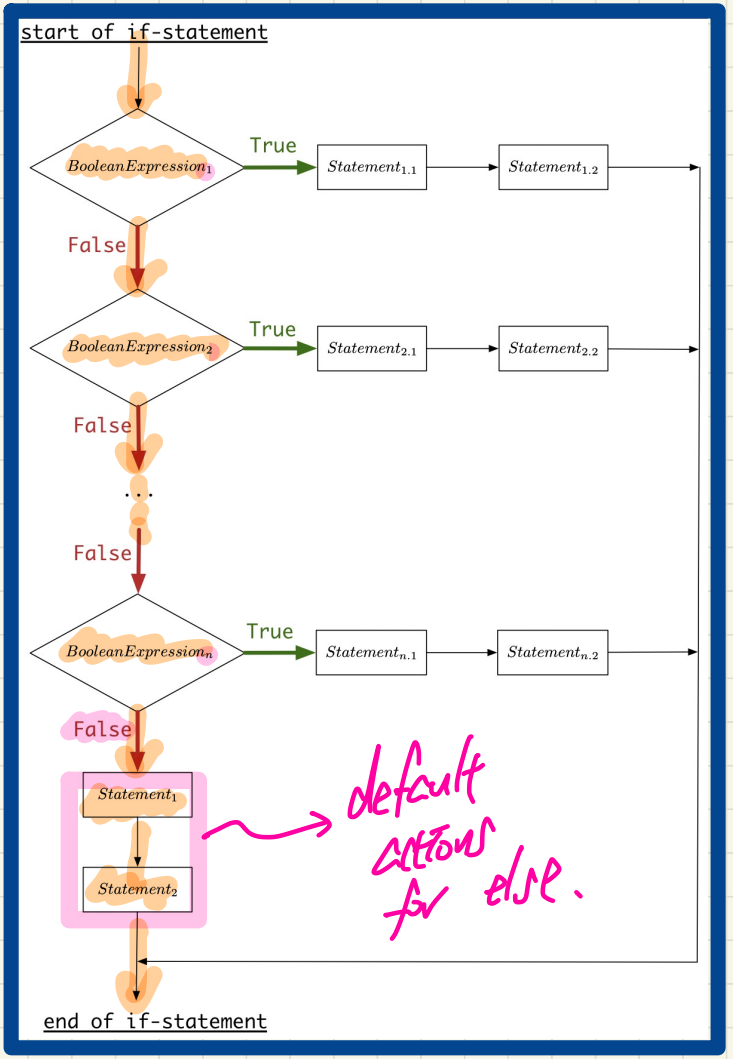
Case 4 An **else** statement is present

BooleanExpression$_1$ evaluares to **false**

BooleanExpression$_2$ evaluares to **false**

...

BooleanExpression$_n$ evaluares to **false**



If-Statement Case 4: Example

No satisfying branches, and an `else` part is present, then the *default action* is executed.

```
int i = 12;
if(i < 0) {
  System.out.println("i is negative");
}
else if(i < 10) {
  System.out.println("i is less than than 10");
}
else if(i == 10) {
  System.out.println("i is equal to 10");
}
else {
  System.out.println("i is greater than 10");
}
```

12 < 0 (F)

12 < 10 (F)

12 == 10 (F)

Console

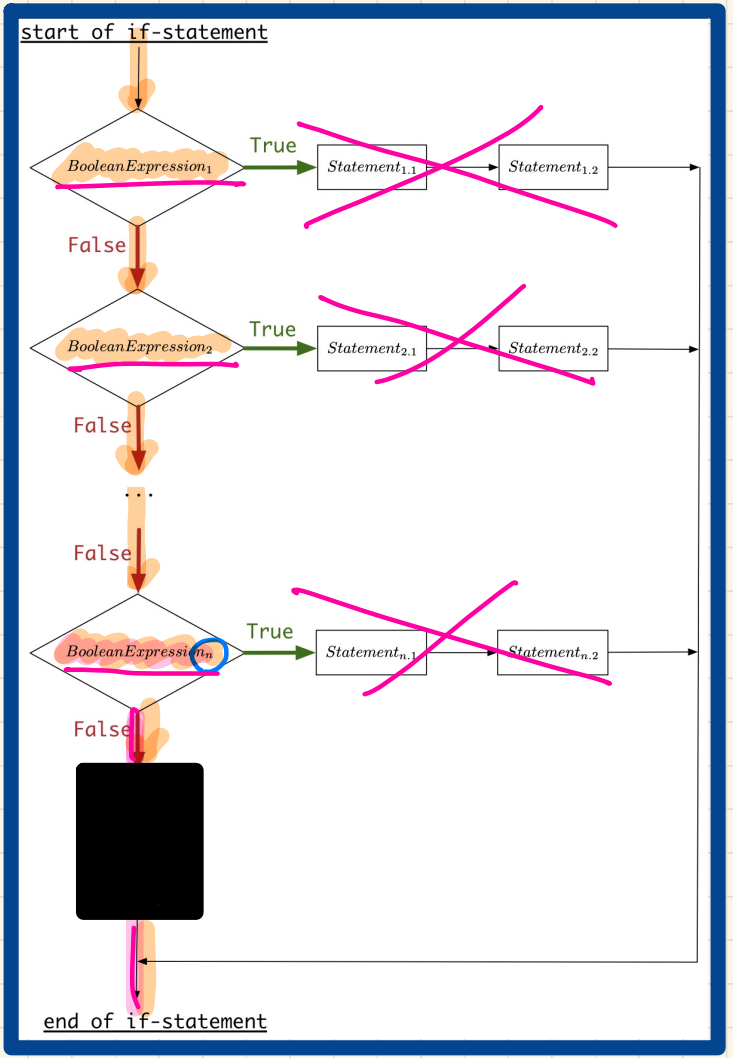
```
I is greater than 10.
```

A Single If-Statement

Syntax

```
if ( BooleanExpression1 ) { /* Mandatory */
  Statement1.1; Statement2.1;
}
else if ( BooleanExpression2 ) { /* Optional */
  Statement2.1; Statement2.2;
}
... /* as many else-if branches as you like */
else if ( BooleanExpressionn ) { /* Optional */
  Statementn.1; Statementn.2;
}
else { /* Optional */
  /* when all previous branching conditions are
  Statement1; Statement2;
}
```

Semantics/ Meaning



Case 5 An **else** statement is **absent**

BooleanExpression1 evaluares to **false**

BooleanExpression2 evaluares to **false**

...

BooleanExpressionn evaluares to **false**

If-Statement Case 5: Example

No satisfying branches, and an `else` part is absent,

then *nothing* is executed.

```
int i = 12;
if(i < 0) {
System.out.println("i is negative");
}
else if(i < 10) {
System.out.println("i is less than than 10");
}
else if(i == 10) {
System.out.println("i is equal to 10");
}
```

12 < 0 (F)

12 < 10 (F)

12 == 10 (F)

Console

Lecture 2

Part D

Selections - Logical Operators

Defining Logical Operators: Truth Tables

Negation (¬, not)

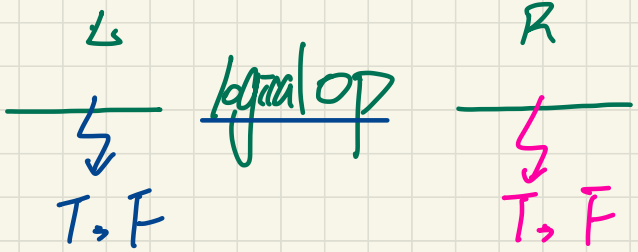
P	! P
false	
true	

Conjunction (∧, and)

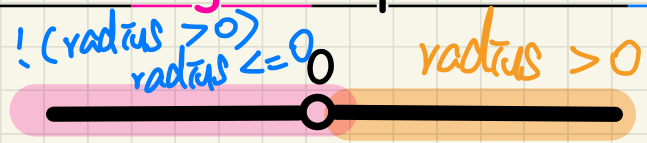
P	Q	P && Q
false	false	
false	true	
true	false	
true	true	

Disjunction (∨, or)

P	Q	P || Q
false	false	
false	true	
true	false	
true	true	



Example of Logical Operation: Negation



Exercise: Run in Debugger.

Test Inputs:

radius = 5

radius = 0

radius = -3

The result is the "negated" value of its operand.

Operand op	!op
true	false
false	true

data type

!F → T

isPositive.

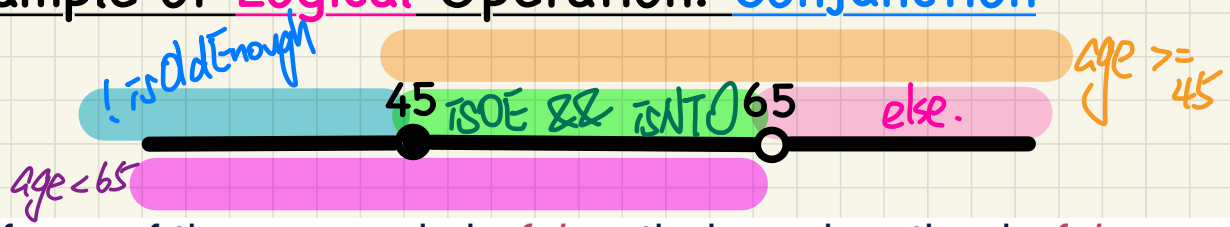
! isPositive → false

! (! isPositive) → True

```
double radius = input.nextDouble();
final double PI = 3.14;
boolean isPositive = radius > 0;
if (!isPositive) {
System.out.println("Error: radius value must be positive.");
}
else {
System.out.println("Area is " + radius * radius * PI);
}
```

relational expression → evaluates to a Boolean value (T, F)

Example of Logical Operation: Conjunction



Test Inputs:

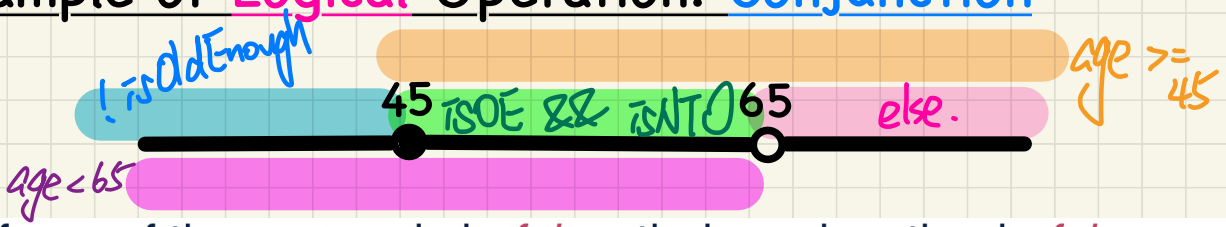
- age = 30
- age = 50
- age = 70

If one of the operands is *false*, their conjunction is *false*.

Left Operand `op1`	Right Operand `op2`	`op1 && op2`
true	*true*	*true*
true	*false*	*false*
false	*true*	*false*
false	*false*	*false*

```
int age = input.nextInt();
boolean isOldEnough = age >= 45;
boolean isNotTooOld = age < 65;
if (!isOldEnough) { /* young */ }
else if (isOldEnough && isNotTooOld) { /* middle-aged */ }
else { /* senior */ }
```

Example of Logical Operation: Conjunction



Test Inputs:

- age = 30
- age = 50
- age = 70

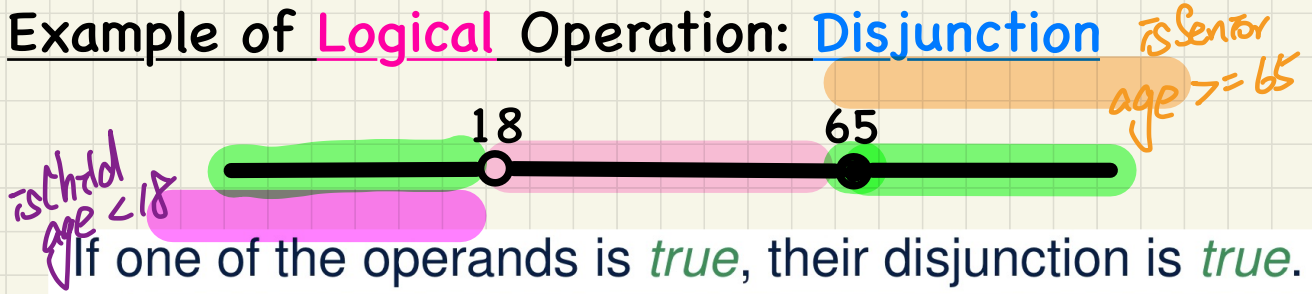
If one of the operands is *false*, their conjunction is *false*.

Left Operand `op1`	Right Operand `op2`	`op1 && op2`
true	true	true
true	false	false
false	true	false
false	false	false

Exercise: Try 30, 70 on Debugger.

```
int age = input.nextInt();
boolean isOldEnough = age >= 45;
boolean isNotTooOld = age < 65;
if (!isOldEnough) { /* young */ }
else if (isOldEnough && isNotTooOld) { /* middle-aged */ }
else { /* senior */ }
```

Example of Logical Operation: Disjunction



If one of the operands is *true*, their disjunction is *true*.

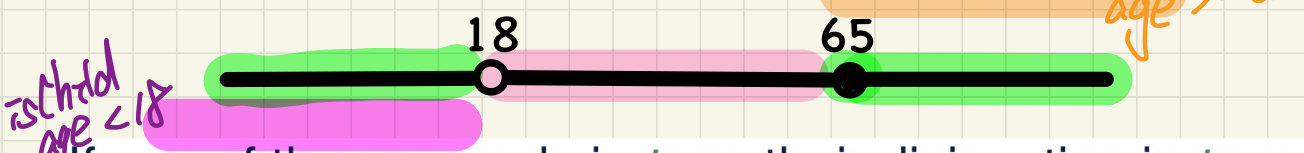
Test Inputs:

- age = 70
- age = 15
- age = 40

Left Operand `op1`	Right Operand `op2`	`op1 || op2`
false	*false*	*false*
true	*false*	*true*
false	*true*	*true*
true	*true*	*true*

```
int age = input.nextInt();
boolean isSenior = age >= 65;
boolean isChild = age < 18;
if (isSenior || isChild) { /* discount */ }
else { /* no discount */ }
```

Example of Logical Operation: Disjunction



If one of the operands is *true*, their disjunction is *true*.

Test Inputs:

- age = 70
- age = 15
- age = 40

Left Operand `op1`	Right Operand `op2`	`op1 || op2`
false	false	false
true	false	true
false	true	true
true	true	true

Exercise: Try all values in Debugger.

```
int age = input.nextInt();
boolean isSenior = age >= 65;
boolean isChild = age < 18;
if (isSenior || isChild) { /* discount */ }
else { /* no discount */ }
```