Lecture 2

Part E

Selections -
Laws of Logical Operators,
Precedence of Logical Operators

Logical Law: Negation of Relational Operation

Relation	Negation	Equivalence
`i > j`	`!(i > j)`	`i <= j`
`i >= j`	`!(i >= j)`	`i < j`
`i < j`	`!(i < j)`	`i >= j`
`i <= j`	`!(i <= j)`	`i > j`

Test Inputs:

i = 17, j = 3

i = -4, j = 13

$17 \leq 3$

$!(i<=j) \equiv i>j$

$!(i>j) \equiv i<=j$

$i>j$

T

```
if (i > j) {
  /* Action 1 */
}
else { /* !(i > j) */
  /* Action 2 */
}
```

$i<=j$

equivalent to

F

```
if (i <= j) {
  /* Action 2 */
}
else { /* !(i <= j) */
  /* Action 1 */
}
```

$i<=j$

$i>j$

$i>j$

Two-Way If-Stmt: Handling Errors

Test Inputs:

radius = 9

radius = -5

Trace on both sides

```
public class ComputeArea {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("Enter a radius value:");
    double radius = input.nextDouble();
    final double PI = 3.14159;
    if (radius < 0) {  /* condition of invalid inputs */
      System.out.println("Error: Negative radius value!");
    }
    else {  /* implicit:  !(radius < 0), or radius >= 0 */
      double area = radius * radius * PI;
      System.out.println("Area is " + area);
    }
    input.close();
  }
}
```

-5

x

-5 >= 0 (F)

!(radius < 0)

≡ radius >= 0

!(radius >= 0)

||

radius < 0

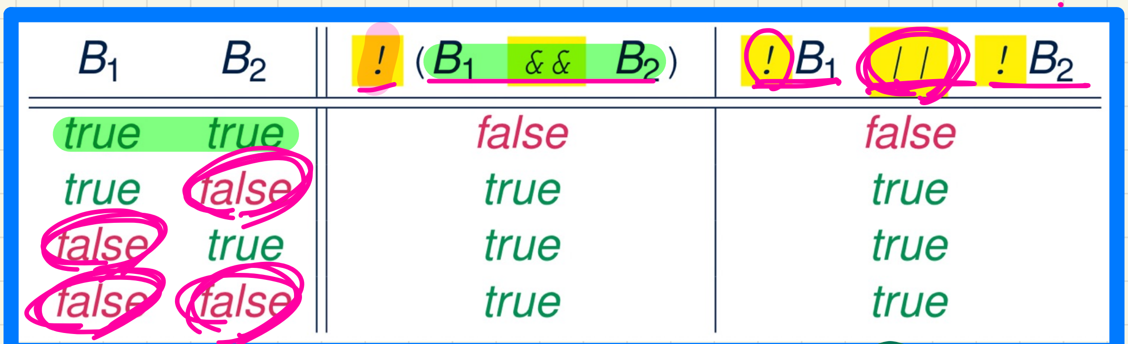
```
public class ComputeArea2 {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    System.out.println("Enter a radius value:");
    double radius = input.nextDouble();
    final double PI = 3.14159;
    if (radius >= 0) {  /* condition of valid inputs */
      double area = radius * radius * PI;
      System.out.println("Area is " + area);
    }
    else {  /* implicit:  !(radius >= 0), or radius < 0 */
      System.out.println("Error: Negative radius value!");
    }
    input.close();
  }
}
```

F

x

Logical Laws: DeMorgan

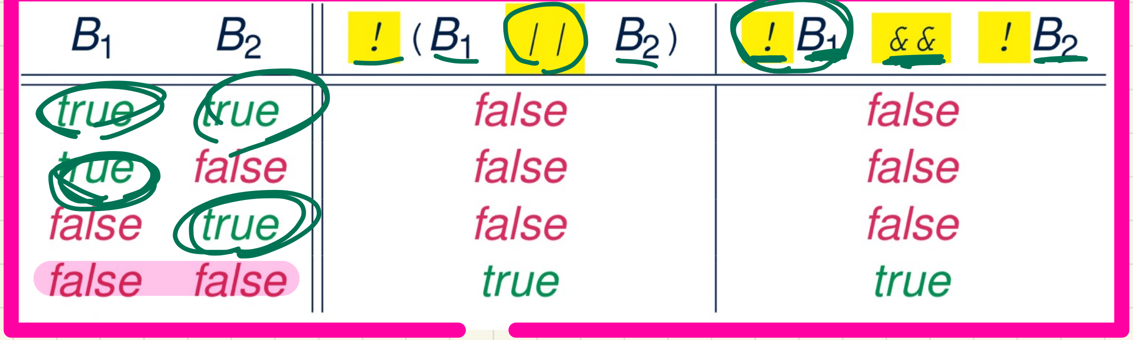
T



B_1	B_2	$!(B_1 \ \&\& \ B_2)$	$!B_1 \ \vert\vert \ !B_2$
true	true	false	false
true	false	true	true
false	true	true	true
false	false	true	true

DeMorgan for Conjunction

!T F



B_1	B_2	$!(B_1 \ \vert\vert \ B_2)$	$!B_1 \ \&\& \ !B_2$
true	true	false	false
true	false	false	false
false	true	false	false
false	false	true	true

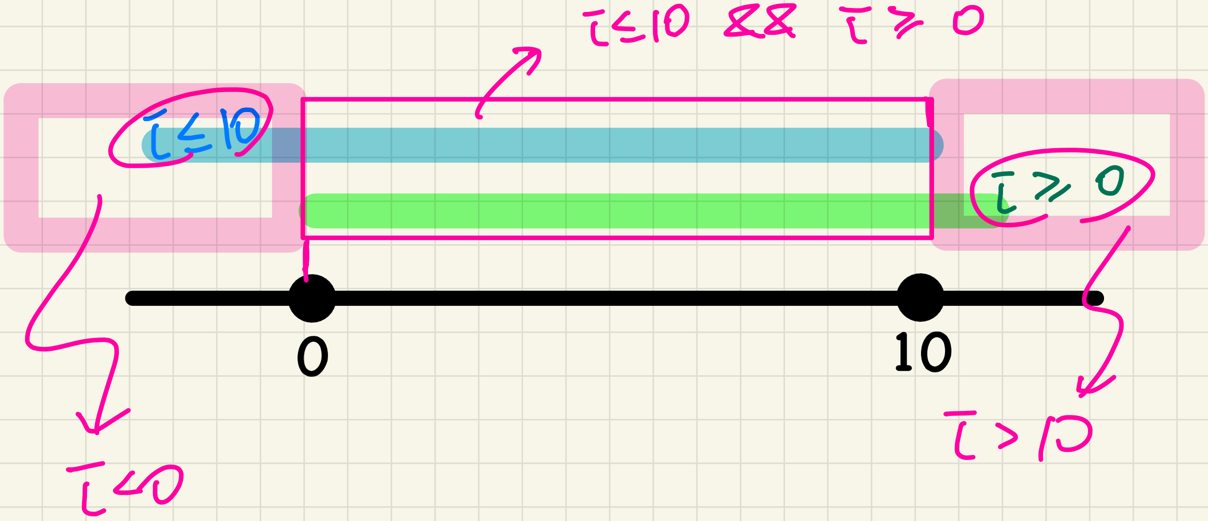
DeMorgan for Disjunction

DeMorgan Law of Conjunction: Example (1)

```
if (0 <= i && i <= 10) { /* Action 1 */ }
else { /* Action 2 */ }
```

- When is *Action 2* executed? `i < 0 || i > 10`

$$!(0 <= i \;\&\&\; i <= 10) \equiv\; !(0 <= i) \;||\; !(i <= 10) \equiv \boxed{0 > i \;||\; i > 10}$$



DeMorgan Law of Conjunction: Example (2)

F.

never executed

```
if(i < 0 && false) { /* Action 1 */ }
else { /* Action 2 */ }
```

T → always executed.

- When is *Action 1* executed? *false*
- When is *Action 2* executed? *true* (i.e., `i >= 0 || true`)

$$!(i < 0 \;\&\&\; false)$$

$$\equiv\ !(i < 0) \;||\; !(false)$$

$$\equiv\ i \geq 0 \;||\; T$$

$$\equiv\ T.$$

$$\boxed{i < 0 \;\&\&\; false}$$

$$\leadsto F.$$

DeMorgan Law of Conjunction: Example (3)

never executed.

```
if (i < 0 && i > 10) { /* Action 1 */ }
else { /* Action 2 */ }
```

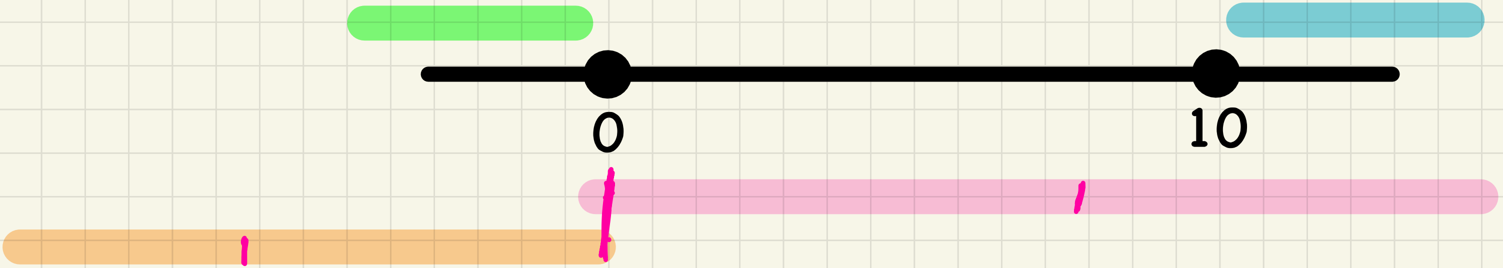
always executed.

- **When** is *Action 1* executed? *false*
- **When** is *Action 2* executed? *true* (i.e., `i >= 0 || i <= 10`)

$$!(i<0 \ \&\& \ i>10)$$

$$\equiv \ !(i<0) \ || \ !(i>10) \ \equiv \ i>=0 \ || \ i<=0$$

T.



DeMorgan Law of Disjunction: Example (1)

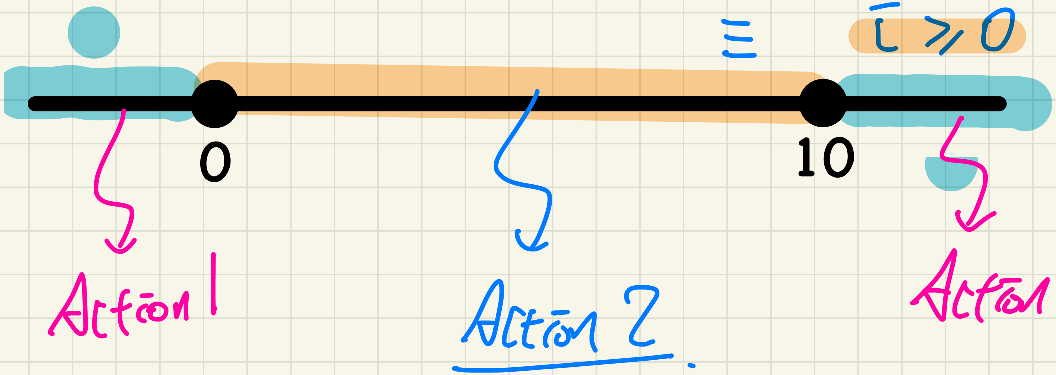
```
if (i < 0 || i > 10) { /* Action 1 */ }
else { /* Action 2 */ }
```

- **When is *Action 2* executed?**

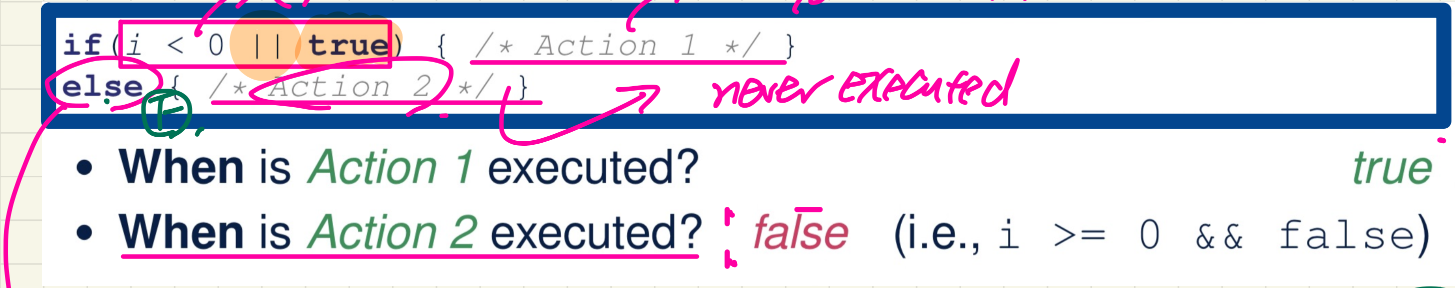
```
0 <= i && i <= 10
```

$$!(i < 0 \boxed{||} i > 10) \equiv !(i<0) \ \&\& \ !(i>10)$$

$$\equiv i \geq 0 \ \&\& \ i \leq 10$$



DeMorgan Law of Disjunction: Example (2)



T

always executed

```
if (i < 0 || true) { /* Action 1 */ }
else { /* Action 2 */ }
```

F

never executed

- When is *Action 1* executed? *true*
- When is *Action 2* executed? *false* (i.e., `i >= 0 && false`)

$!\,(i < 0 \;||\; true)$

↓ DeMorgan

?   Exercise

F

$$\boxed{i < 0 \;||\; \underline{\underline{true}}}$$

T?

DeMorgan Law of Disjunction: Example (3)

T

always executed

```
if (i < 10 || i >= 10) { /* Action 1 */ }
else { /* Action 2 */ }
```

never executed

- **When is** *Action 1* **executed?** *true*
- **When is** *Action 2* **executed?** *false* (i.e., `i >= 10 && i < 10`)

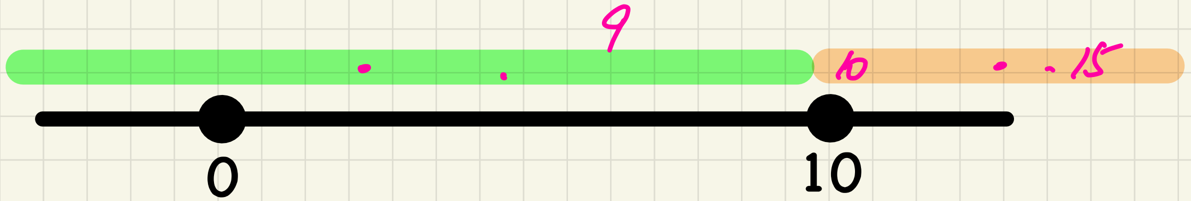
DeMorgan.

$!(i<10 \;||\; i>=10) \equiv \;!(i<10) \;\&\&\; !(i>=10)$

||| Exercise

?

F.



Precedence of Logical Operators

```
boolean p = true;
boolean q = true;
boolean r = false;
```

!
&&
||

&& has higher precedence than ||

evaluated first

p || (q && r)

(p || q) && r

p || (q && r)

$T \;||\; (T \;\&\&\; F)$ → T

F

$(T \;||\; T) \;\&\&\; F$ → F

T

p || q && r

① = ② ≠ ③

③ ((p || q) && r)

T

Exercise

Find p, q, r showing ② and ③ may evaluate to different results.

① ! p || q && r = ② (!p) || (q && r)

Lecture 2

Part F

Selections -
Two-Way vs. Multi-Ways If-Statements,
Nested If-Statements

Two-Way If-Statement without else Part

Test Inputs:

radius = 10

radius = -23

-23 10 T F

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("Area for the circle of is " + area);
}
```

Console

Area for circle is . –

Console

-23 10 T

```
if (radius >= 0) {
  area = radius * radius * PI;
  System.out.println("Area for the circle of is " + area);
}
else {
  /* Do nothing. */
}
```

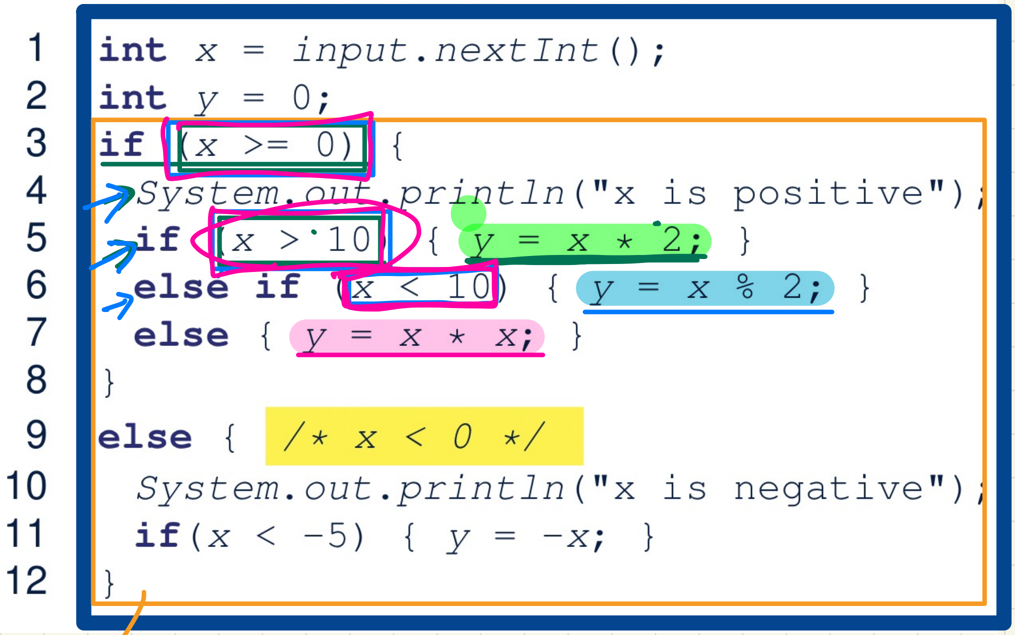
→ ! (radius >= 0).

Console

Area for circle is - -

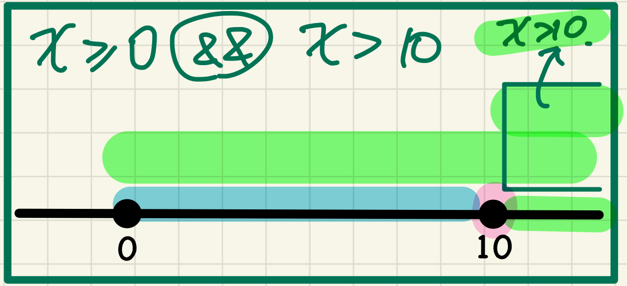
Console

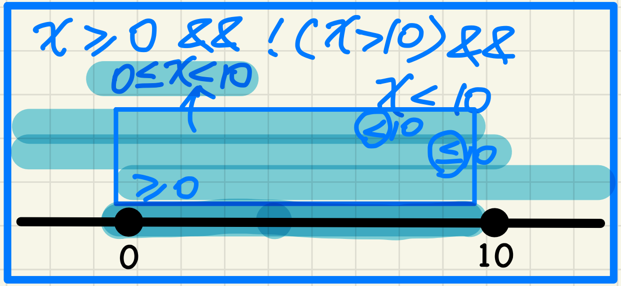
Compound If-Statement: Implicit Conditions

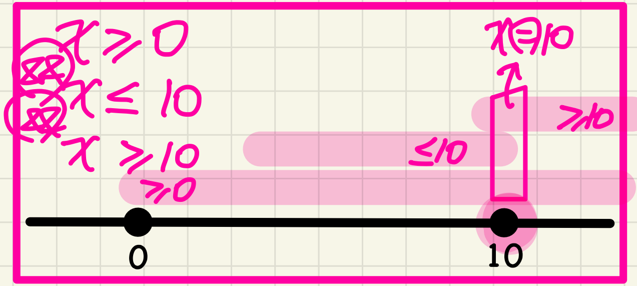


```
int x = input.nextInt();
int y = 0;
if (x >= 0) {
  System.out.println("x is positive");
  if (x > 10) { y = x * 2; }
  else if (x < 10) { y = x % 2; }
  else { y = x * x; }
}
else { /* x < 0 */
  System.out.println("x is negative");
  if(x < -5) { y = -x; }
}
```

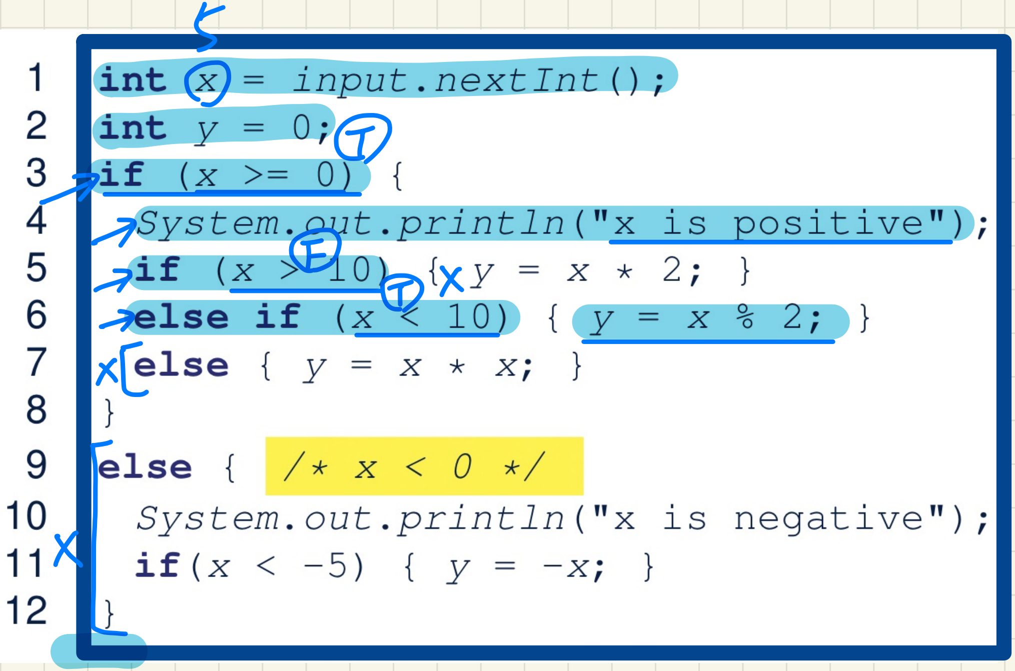
single if-statement







Compound If-Statement: Tracing



```
int x = input.nextInt();
int y = 0;
if (x >= 0) {
  System.out.println("x is positive");
  if (x > 10) { y = x * 2; }
  else if (x < 10) { y = x % 2; }
  else { y = x * x; }
}
else { /* x < 0 */
  System.out.println("x is negative");
  if(x < -5) { y = -x; }
}
```

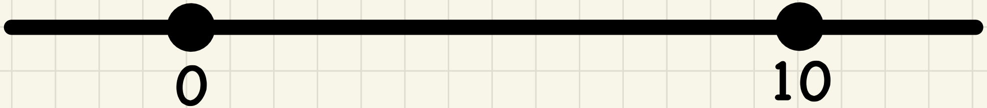
Test Inputs:

x = 5

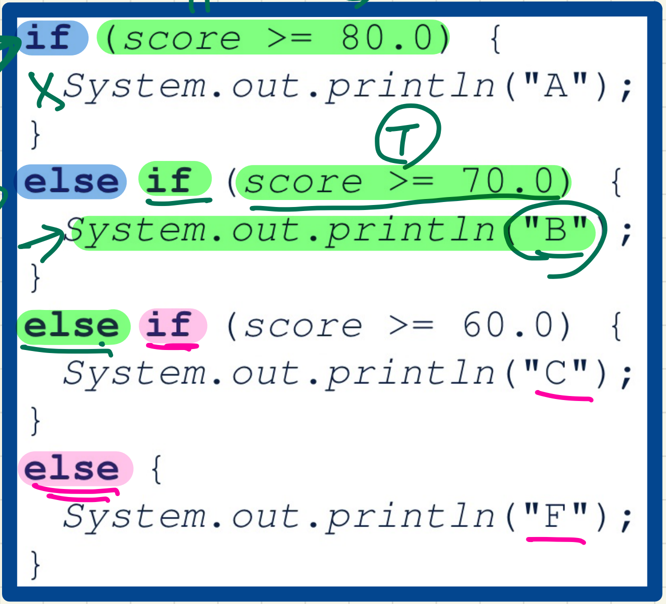
x = 10

x = -2

Exercise:
Trace on paper and Debugger.



Multi-Way If-Statement with else Part



```
           71        F
if (score >= 80.0) {
X System.out.println("A");
}
                    T
else if (score >= 70.0) {
  System.out.println("B");
}
else if (score >= 60.0) {
  System.out.println("C");
}
else {
  System.out.println("F");
}
```

≡

```
           71
if (score >= 80.0) {
  X        "A"
}
else {
           71           T
  if (score >= 70.0) {
     →  "B"
  }
  else {
     if (score >= 60.0) {
          "C"
     }
     else { "F" }
  }
}
```

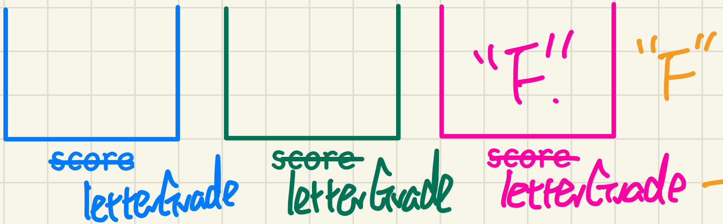
Test Inputs:

score = 83

score = 71

score = 59

Multi-Way If-Statement without else Part



```
String letterGrade = "F";
if (score >= 80.0) F {
X letterGrade = "A";
}
else if (score >= 70.0) F {
X letterGrade = "B";
}
else if (score >= 60.0) F {
X letterGrade = "C";
}
```

≡

```
String letterGrade = "F";
if (score >= 80.0) {
X letterGrade = "A";
}
else {
  if (score >= 70.0) F {
  X letterGrade = "B";
  }
  else {
    if (score >= 60.0) F {
    X letterGrade = "C";
    }
    else {
      /* do nothing */
    }
  }
}
```

Test Inputs:

score = 83

score = 71

score = 59

Lecture 2

Part G

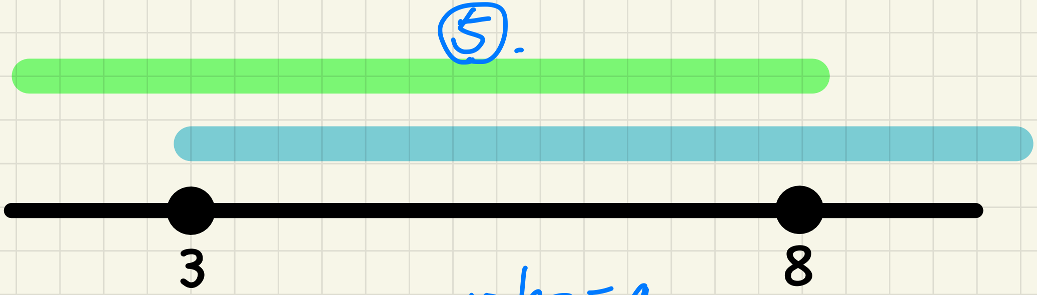
Selections -
Overlapping vs. Disjoint Conditions,
Single If-Stmt vs. Multiple If-Stmts

Overlapping vs. Non-Overlapping Intervals

disjoint.

i >= 3

i <= 8

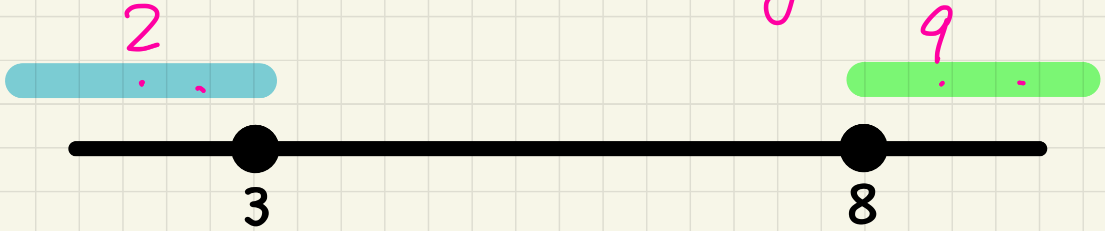


overlapping

i <= 3

i >= 8

non-overlapping.



Single If-Stmt vs. Multiple If-Stmts: Overlapping Conditions

- Single vs. multiple
- overlapping

```
int i = 5;
if (i >= 3) {System.out.println("i is >= 3");}
else if (i <= 8) {System.out.println("i is <= 8");}
```

T

Console

```
i is >= 3
```

independent if-stmts.

```
int i = 5;
if (i >= 3) {System.out.println("i is >= 3");}
if (i <= 8) {System.out.println("i is <= 8");}
```

Console

```
i is >= 3
i is <= 8
```

Single If-Stmt vs. Multiple If-Stmts: Non-Overlapping Conditions

```
int i = 2;
if(i <= 3) {System.out.println("i is <= 3");}
else if(i >= 8) {System.out.println("i is >= 8");}
```

Console

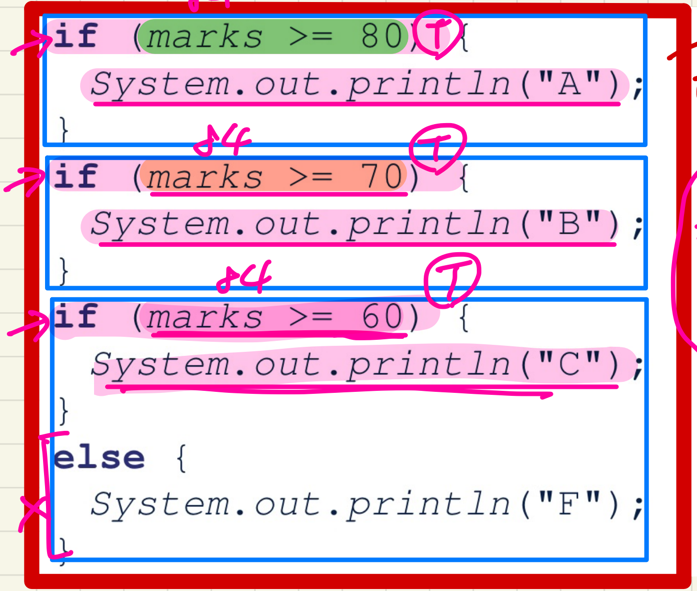
```
i is <=3
```

```
int i = 2;
if(i <= 3) {System.out.println("i is <= 3");}
if(i >= 8) {System.out.println("i is >= 8");}
```

Console

```
i is <=3
```

Common Error: Multiple If-Statements with Overlapping Conditions



```
if (marks >= 80) {
  System.out.println("A");
}
if (marks >= 70) {
  System.out.println("B");
}
if (marks >= 60) {
  System.out.println("C");
}
else {
  System.out.println("F");
}
```

84 T

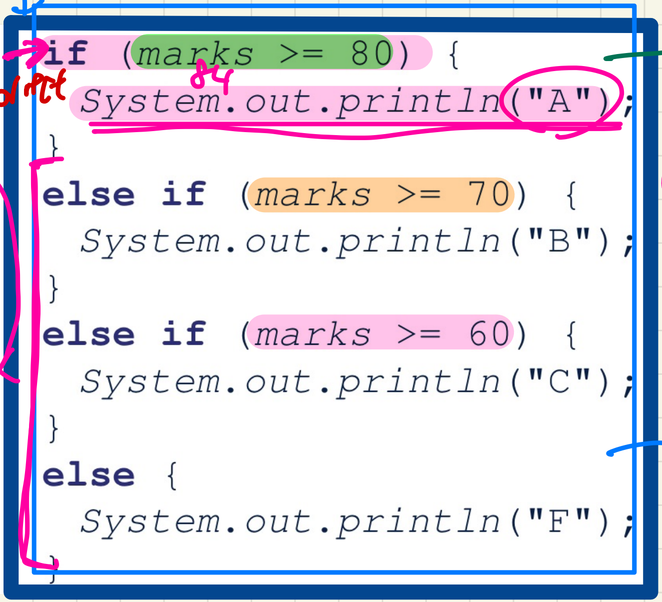
84 T

84 T

A
B
C

Incorrect

3 if-statements.



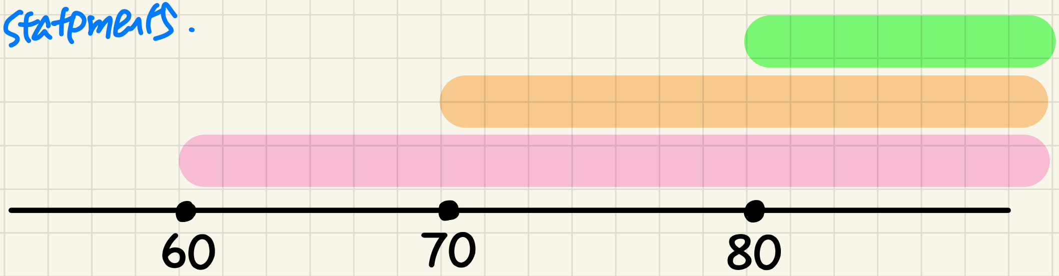
```
if (marks >= 80) {
  System.out.println("A");
}
else if (marks >= 70) {
  System.out.println("B");
}
else if (marks >= 60) {
  System.out.println("C");
}
else {
  System.out.println("F");
}
```

84

Correct

A

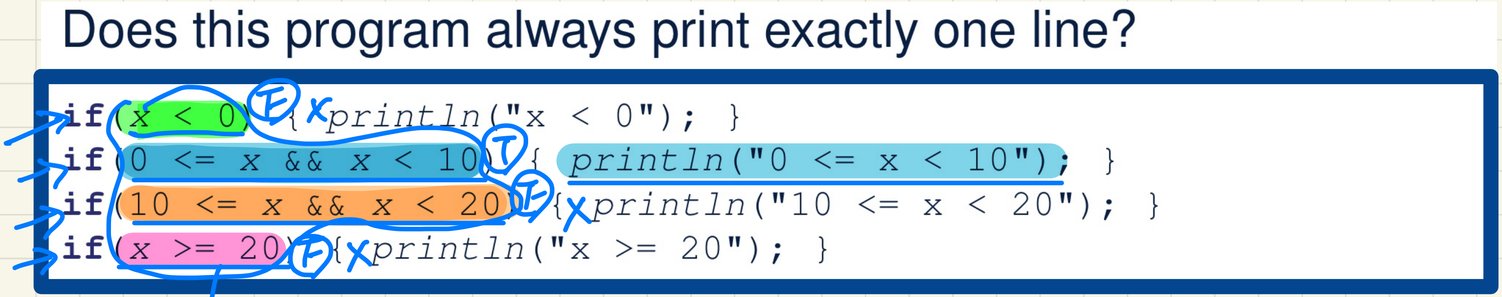
single if-statement.



Test Inputs:

marks = 84

Overlapping Conditions: Exercise (1)



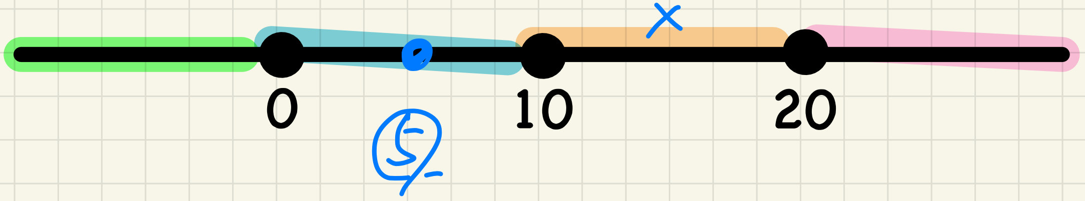
Does this program always print exactly one line?

```
if (x < 0) { println("x < 0"); }
if (0 <= x && x < 10) { println("0 <= x < 10"); }
if (10 <= x && x < 20) { println("10 <= x < 20"); }
if (x >= 20) { println("x >= 20"); }
```

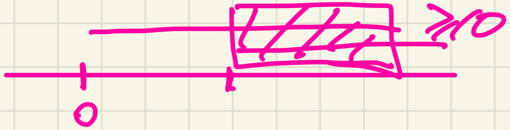
disjoint.

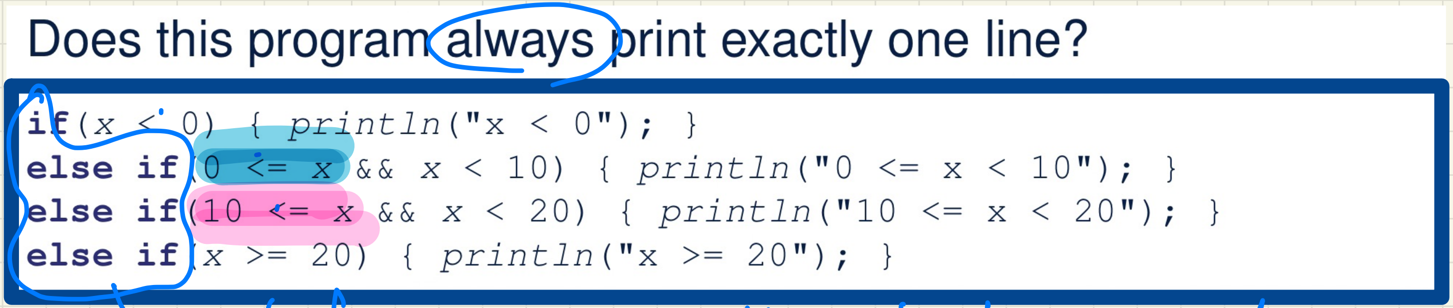
no value can satisfy more than one of them

⇒ only one if-stmt's body of code is executed.



Overlapping Conditions: Exercises (2, 3)

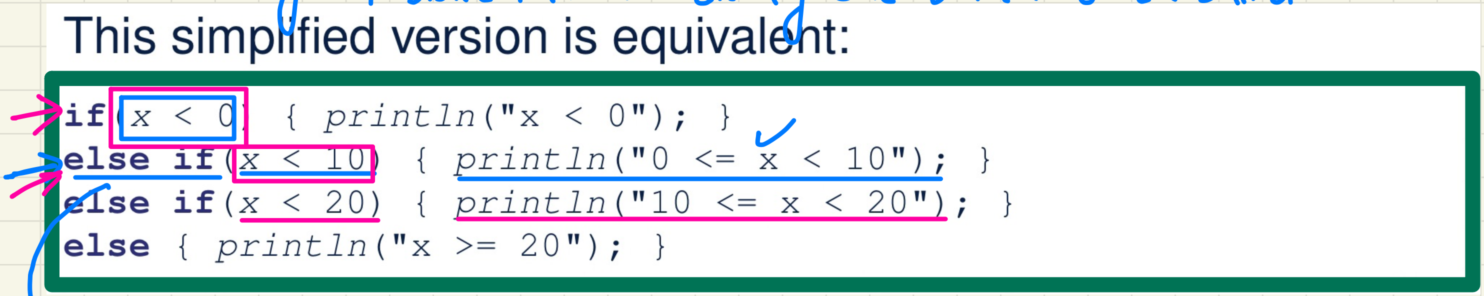




Does this program always print exactly one line?

```
if(x < 0) { println("x < 0"); }
else if(0 <= x && x < 10) { println("0 <= x < 10"); }
else if(10 <= x && x < 20) { println("10 <= x < 20"); }
else if(x >= 20) { println("x >= 20"); }
```

single if statement ⇒ exactly one branch is executed



This simplified version is equivalent:

```
if(x < 0) { println("x < 0"); }
else if(x < 10) { println("0 <= x < 10"); }
else if(x < 20) { println("10 <= x < 20"); }
else { println("x >= 20"); }
```

$!(x<0)\ \&\&\ x<10$

$\equiv\ x \geq 0\ \&\&\ x < 10$

$!(x<0)\ \&\&\ !(x<10)\ \&\&\ x<20$

$x \geq 10$

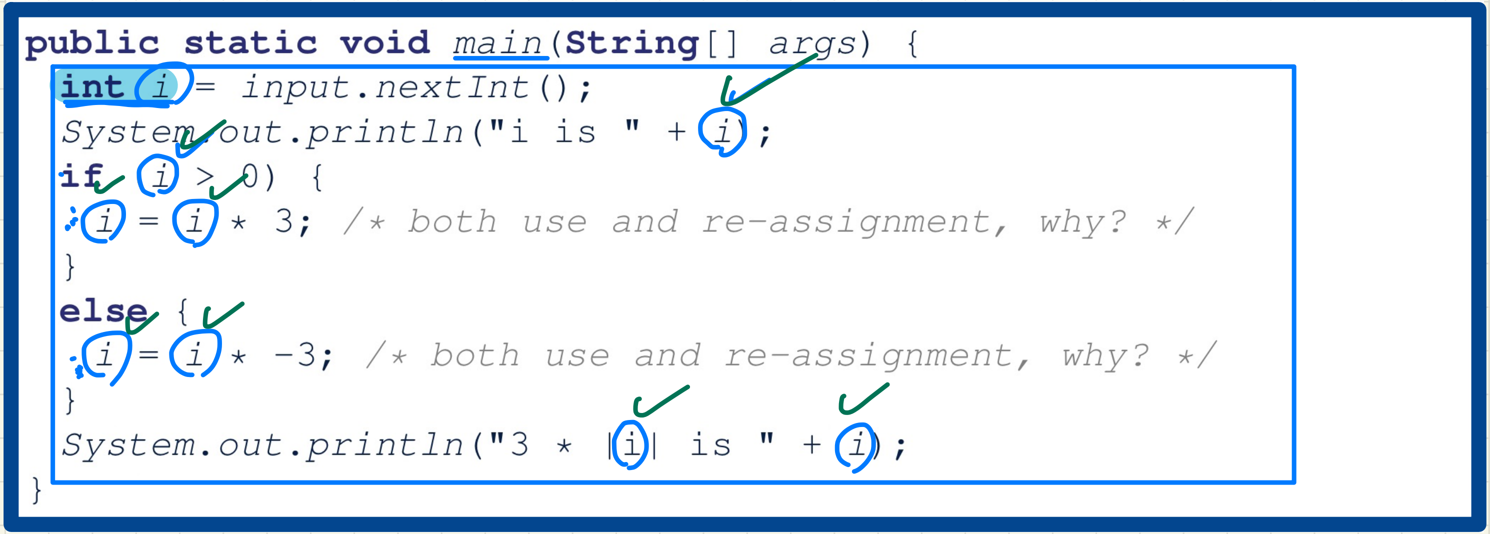
$\equiv\ \boxed{x \geq 0\ \&\&\ x \geq 10}\ \&\&\ x<20$

Lecture 2

Part H

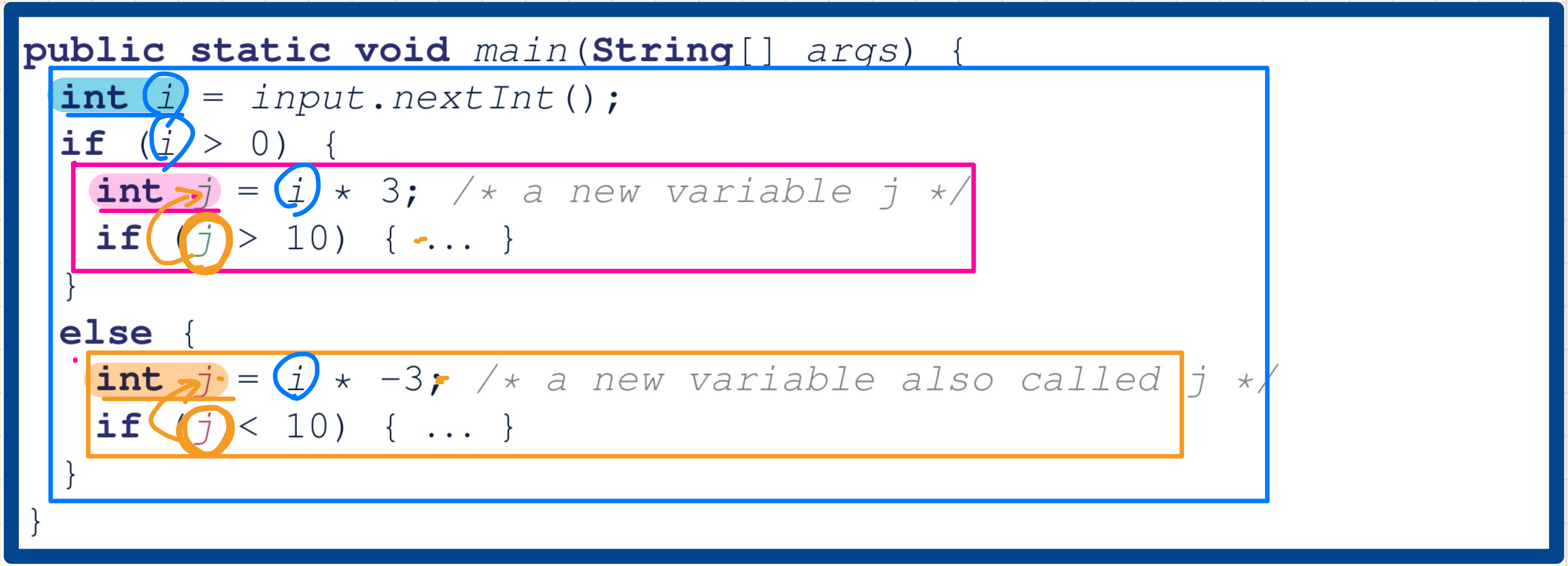
Selections -
Scope of Variables

Scope of Variables: Method



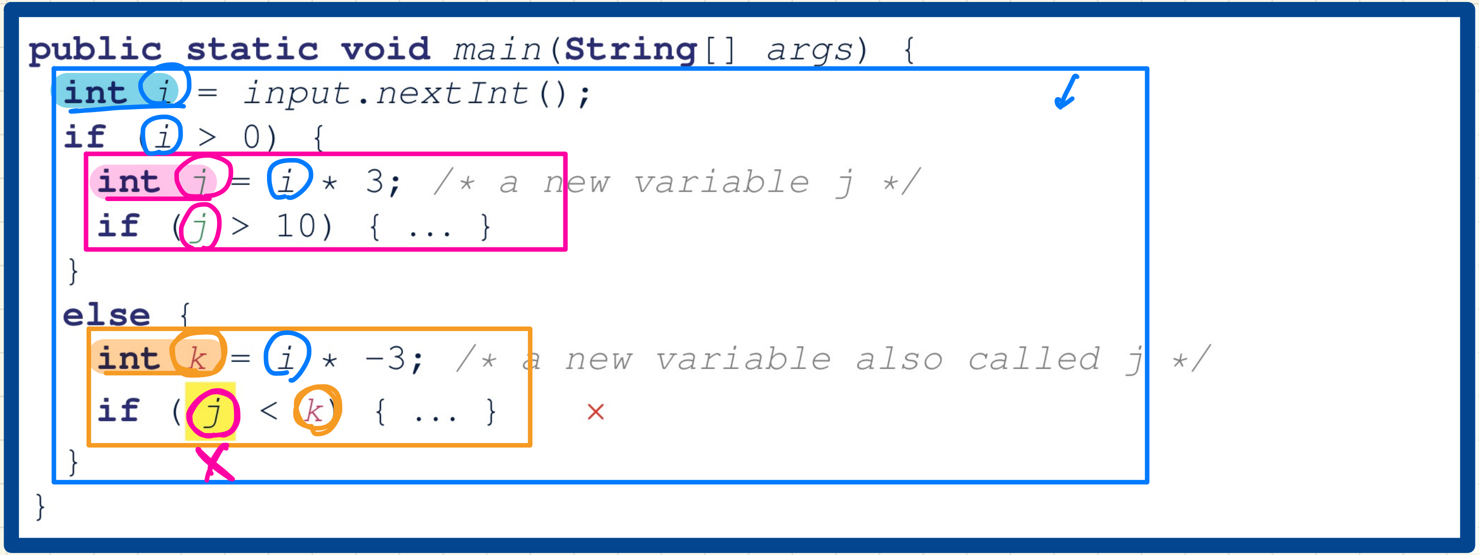
```
public static void main(String[] args) {
  int i = input.nextInt();
  System.out.println("i is " + i);
  if (i > 0) {
    i = i * 3; /* both use and re-assignment, why? */
  }
  else {
    i = i * -3; /* both use and re-assignment, why? */
  }
  System.out.println("3 * |i| is " + i);
}
```

Scope of Variables: Branches



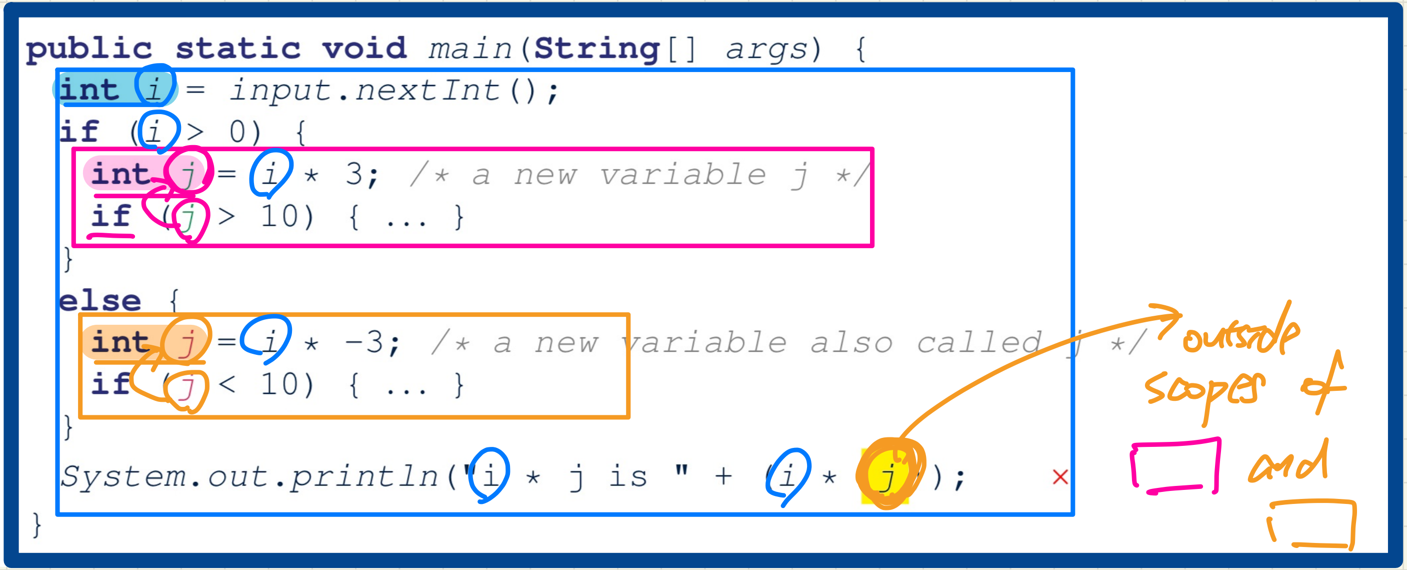
```java
public static void main(String[] args) {
  int i = input.nextInt();
  if (i > 0) {
    int j = i * 3; /* a new variable j */
    if (j > 10) { ... }
  }
  else {
    int j = i * -3; /* a new variable also called j */
    if (j < 10) { ... }
  }
}
```

Scope of Variables: Use of Variables from Other Branches



```java
public static void main(String[] args) {
  int i = input.nextInt();                                     ✓
  if (i > 0) {
    int j = i * 3; /* a new variable j */
    if (j > 10) { ... }
  }
  else {
    int k = i * -3; /* a new variable also called j */
    if (j < k) { ... }     ×
  }
}
```

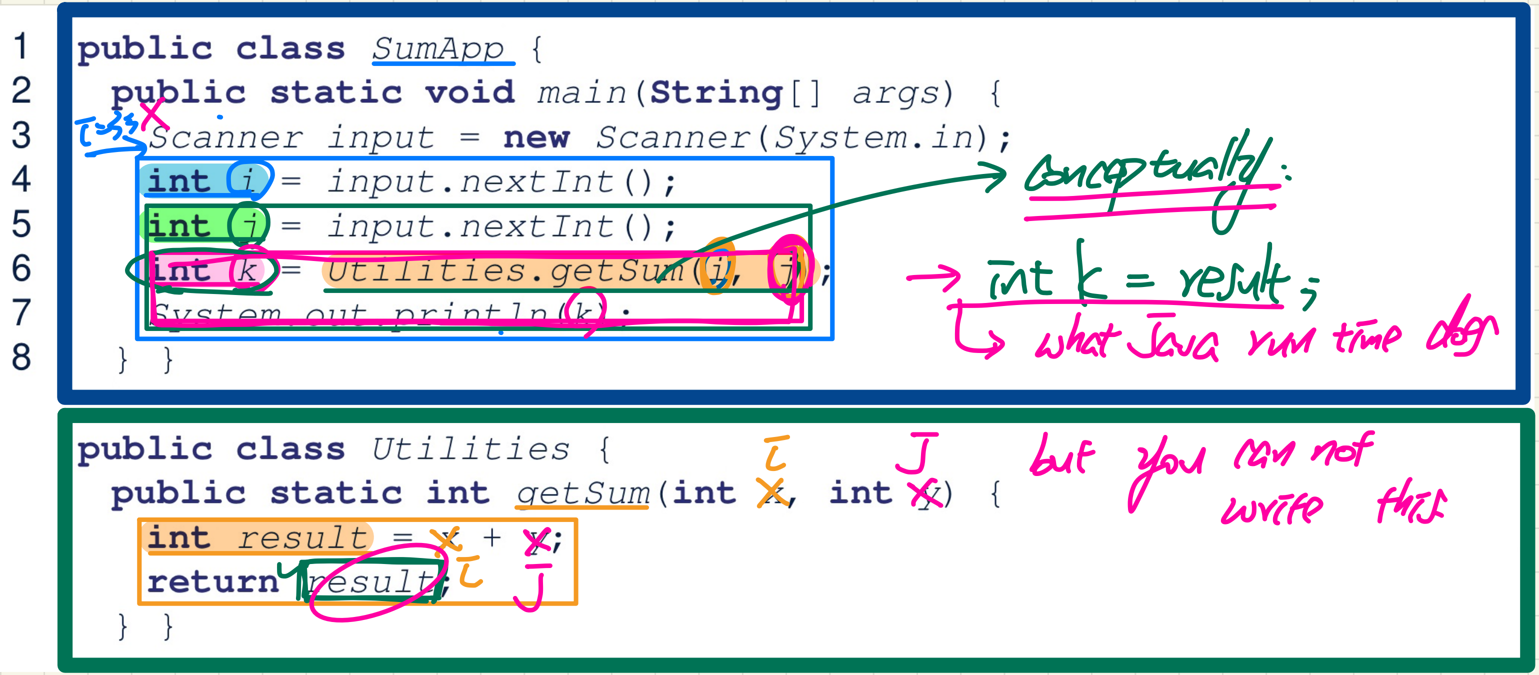
Scope of Variables: Use of Variables Outside If-Stmt



```
public static void main(String[] args) {
  int i = input.nextInt();
  if (i > 0) {
    int j = i * 3; /* a new variable j */
    if (j > 10) { ... }
  }
  else {
    int j = i * -3; /* a new variable also called j */
    if (j < 10) { ... }
  }
  System.out.println("i * j is " + (i * j));          ×
}
```

outside scopes of □ and □

Scope of Variables: Method Parameters & Return Values



```
public class SumApp {
  public static void main(String[] args) {
    Scanner input = new Scanner(System.in);
    int i = input.nextInt();
    int j = input.nextInt();
    int k = Utilities.getSum(i, j);
    System.out.println(k);
  } }
```

```
public class Utilities {
  public static int getSum(int x, int y) {
    int result = x + y;
    return result;
  } }
```

i: 3-5

conceptually:

int k = result;

what Java run time does

but you can not write this

i j