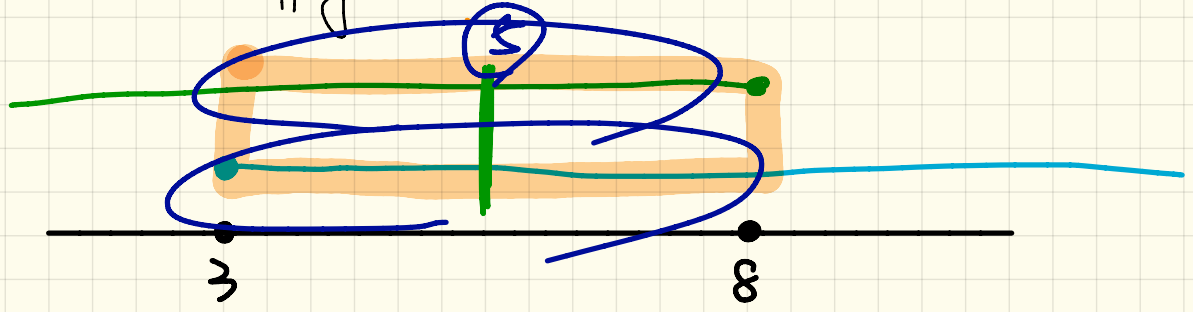


Wednesday January 23

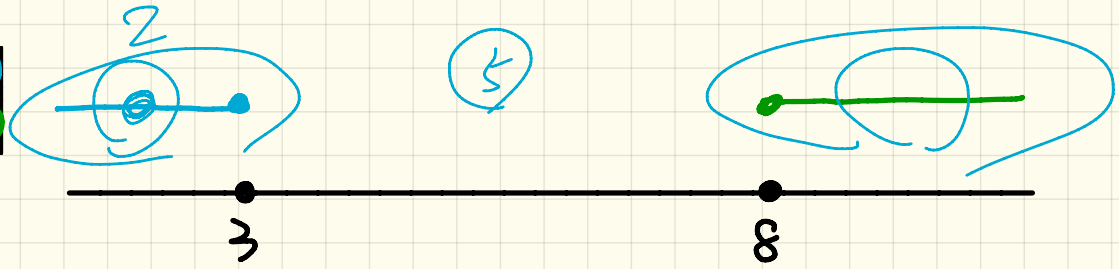
Lecture 6

Overlapping vs. Non-Overlapping Intervals

$I \Rightarrow$
 $I \Leftarrow$



$I \Leftarrow 3$
 $I \Rightarrow 8$



Single If-Statement vs. Multiple If-Statements: Overlapping Conditions

→ a single i.s.

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

i is >= 3

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

i is >= 3
i is <= 8

2 if statements.

Single If-Statement vs. Multiple If-Statements: 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");}
```

i is <= 3

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

i is <= 3

4.5

2 7.5

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);  
}
```

sub-scope

subscope

Scope of Variables: Branches

```
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) { ... }  
  }  
} int i X
```

The image shows a Java code snippet with handwritten annotations. A blue box highlights the entire code block. A pink box highlights the first branch (the `if (i > 0)` block), and a green box highlights the second branch (the `else` block). Red annotations include arrows pointing to the `int` keyword in both branches, circles around the variable `i` in both branches, and a large circle around the `else` keyword. A red 'X' is written next to the `int` keyword in the second branch. The text `} int i X` is written at the bottom left.

Scope of Variables : Illegal Use of Variable from Other Branch

```
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) { ... }  
    }  
}
```

illegal: k is declared in a diff. mb scope.

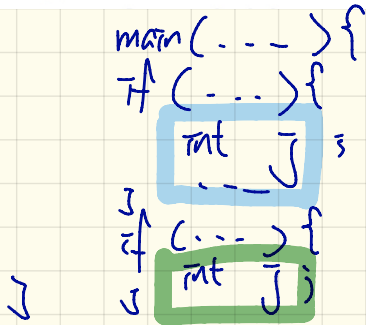
illegal: j is declared in a different mb scope.

x

Scope of Variables: Illegal Use of Variable Outside If-Statement

```
1 public static void main(String[] args) {
2     int i = input.nextInt();
3     if (i > 0) {
4         int j = i * 3; /* a new variable j */
5         if (j > 10) { ... }
6     }
7     else {
8         int j = i * -3; /* a new variable also called j */
9         if (j < 10) { ... }
10    }
11    System.out.println("i * j is " + i * j);
12 }
```

The code above illustrates variable scope. The variable `i` is declared in the `main` method and is accessible throughout. The variable `j` is declared inside two different `if` blocks. The first `if` block (lines 4-5) and the second `if` block (lines 8-9) each declare their own `j`. The use of `i * j` on line 11 is illegal because `j` is not in scope at that point. The diagram uses a blue box for the `main` method, a pink box for the first `if` block, and a green box for the second `if` block. A blue circle around `i` and a red 'X' over `j` on line 11 indicate the error.



Compound If-Statement

Test 1: $x = 5$
Test 2: $x = 10$
Test 3: $x = -2$

```
1 int x = input.nextInt();
2 int y = 0;
3 if (x >= 0) {
4     System.out.println("x is positive");
5     if (x > 10) { y = x * 2; }
6     else if (x <= 10) { y = x % 2; }
7     else { y = x * x; }
8 }
9 else { /* x < 0 */
10    System.out.println("x is negative");
11    if (x < -5) { y = -x; }
12 }
```

Handwritten annotations: Blue arrows point to lines 1-4. Green arrows point to lines 5-8. Pink arrows point to lines 9-12. A yellow highlight is on line 9. Circled values include x=5, x=10, x=-2, and x < -5. A pink 'T' is written below the x < -5 condition. A blue '10' is written above the first if statement. A blue '2' is written above the second if statement. A blue 'x' is written next to the first if statement. A blue 'x' is written next to the second if statement. A blue 'x' is written next to the else block. A blue 'x' is written next to the if (x < -5) statement. A blue 'x' is written next to the else if statement. A blue 'x' is written next to the else block. A blue 'x' is written next to the if (x < -5) statement. A blue 'x' is written next to the else if statement. A blue 'x' is written next to the else block.

Truth Tables of Logical Operators

Conjunction (and)

P	Q	$P \& Q$
false	false	false
false	true	false
true	false	false
true	true	true

Negation (not)

P	$\neg P$
true	false
false	true

Disjunction (or)

P	Q	$P \vee Q$
false	false	false
false	true	true
true	false	true
true	true	true

Example of Logical Operation: Negation

Test 1: 0
Test 2: -3
Test 3: 5

Operand	op	!op
true		false
false		true

```
double radius = input.nextDouble();  
boolean isPositive = radius > 0;  
if (!isPositive) /* not the case that isPositive is true */  
    System.out.println("Error: radius value must be positive.");  
else {  
    System.out.println("Area is " + radius * radius * PI);  
}
```

Example of Logical Operation: Conjunction

Test 1: age = 30
Test 2: age = 50
Test 3: age = 70

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

✓ 50 30

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

Example of Logical Operation: Disjunction

Test 1: age = 70
Test 2: age = 15
Test 3: 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 */ }
```

F || F