Tutorial on Object-Oriented Programming in Java

(EECS1022-W21)

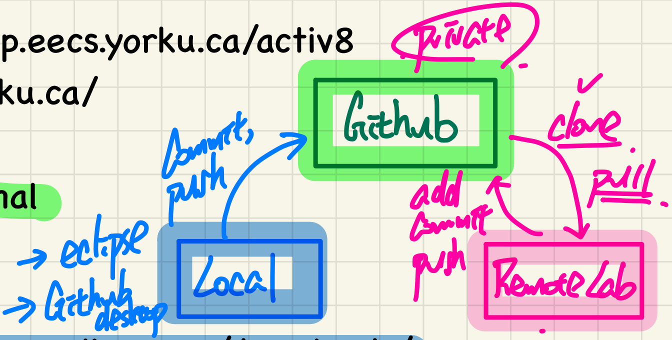
EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 1

Separation of Concerns - Model, Console Apps, Unit Testing

Work Environment

+ Software Management
  * Github tutorial:
    https://www.youtube.com/playlist?list=PL5dxAmCmjv_58KxTSd1CRbpinmSF8EPJx
  * Private repository for EECS1022 workspace: https://github.com/
+ Lab Machines
  * Create an EECS account: https://webapp.eecs.yorku.ca/activ8
  * Remote labs: https://remotelab.eecs.yorku.ca/
  * Eclipse available
  * Github commands available via a terminal
  * Clone a copy of repository here
+ Your Own Machine
  * Download the latest Eclipse: https://www.eclipse.org/downloads/
  * Github desktop: https://desktop.github.com/
  * Clone a copy of repository here



Working over Remote Labs

- Do not invoke the eclipse via the GUI icon!
  (which invokes Version 4.12)
- On a terminal, type eclipse4.16 &
- Do not type eclipse (which also invokes Version 4.12)
- Before you log out and all data get lost,
  always commit and push your work back to Github!!

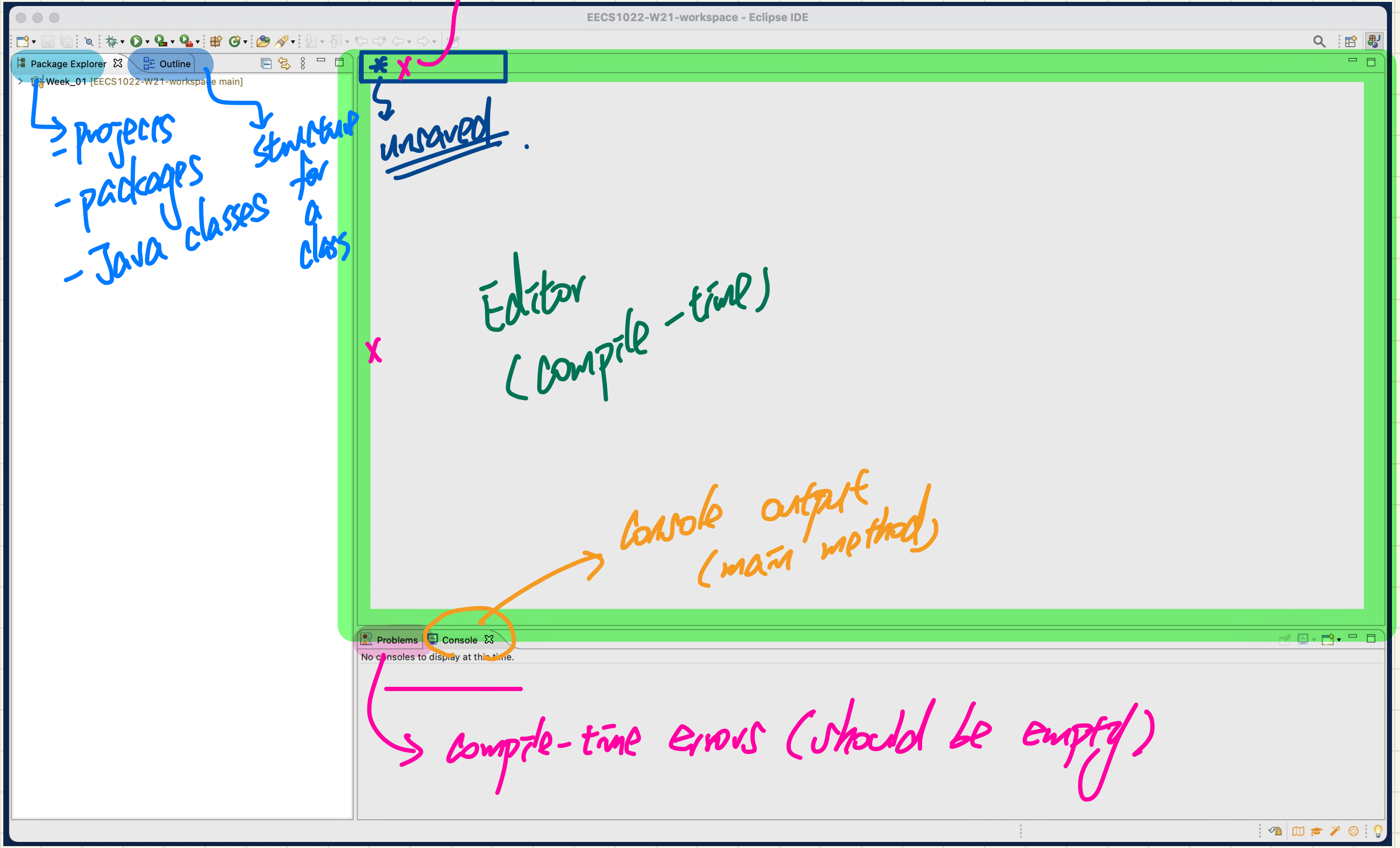
If You Want Extra Practice:

https://www.eecs.yorku.ca/~jackie/teaching/tutorials/index.html#java_from_scratch

More examples (EECS1021-W19)

Java Perspective

compile-time error → code cannot be executed.


EECS1022-W21-workspace - Eclipse IDE
Package Explorer
Outline
Week_01 [EECS1022-W21-workspace main]
Problems
Console

No consoles to display at this time.

- projects
- packages
- Java classes

structure for a class

* x

unsaved.

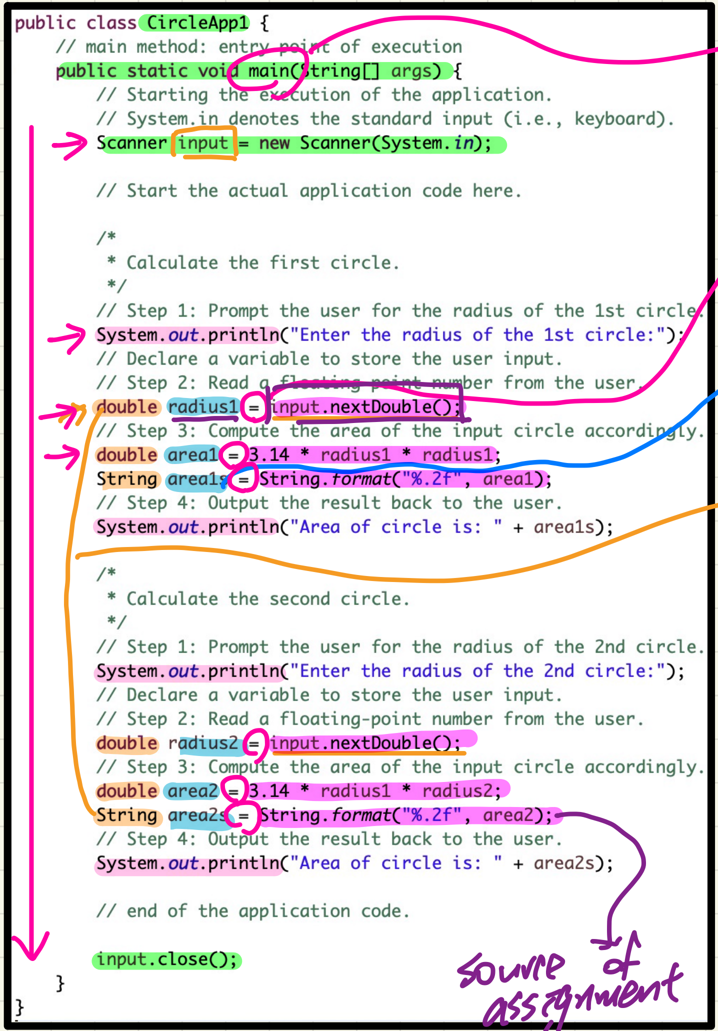
x

Editor (compile-time)

Console output (main method)

compile-time errors (should be empty)

entry point Console Application



```
public class CircleApp1 {
    // main method: entry point of execution
    public static void main(String[] args) {
        // Starting the execution of the application.
        // System.in denotes the standard input (i.e., keyboard).
        Scanner input = new Scanner(System.in);

        // Start the actual application code here.

        /*
         * Calculate the first circle.
         */
        // Step 1: Prompt the user for the radius of the 1st circle.
        System.out.println("Enter the radius of the 1st circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius1 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.
        double area1 = 3.14 * radius1 * radius1;
        String area1s = String.format("%.2f", area1);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area1s);

        /*
         * Calculate the second circle.
         */
        // Step 1: Prompt the user for the radius of the 2nd circle.
        System.out.println("Enter the radius of the 2nd circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius2 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.
        double area2 = 3.14 * radius1 * radius2;
        String area2s = String.format("%.2f", area2);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area2s);

        // end of the application code.

        input.close();
    }
}
```

source of assignment

assignment operator

names of variable

data types.

(for declaring variable).

int

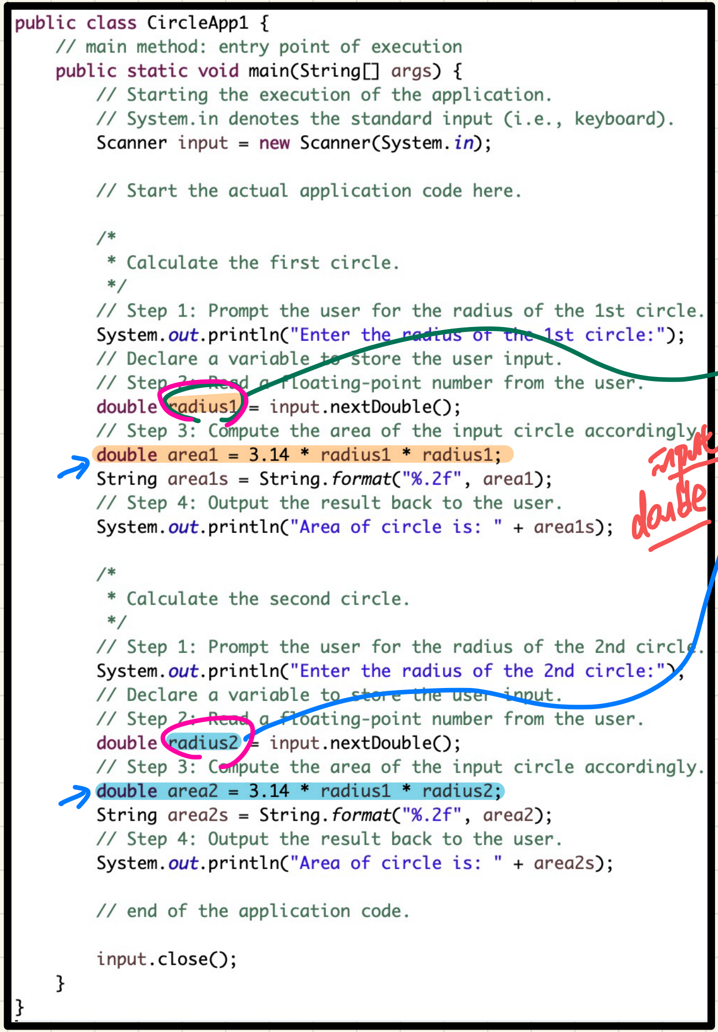
double

boolean

char

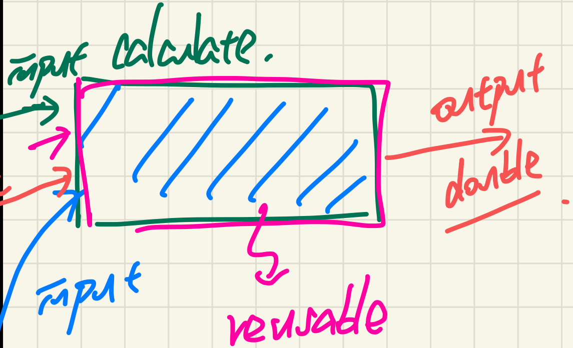
String

Console Application



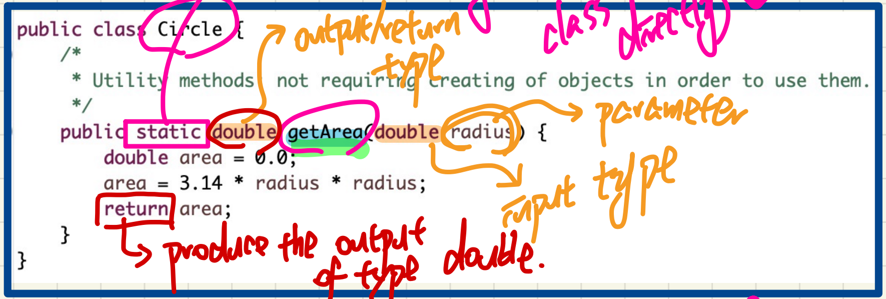
```
public class CircleApp1 {
    // main method: entry point of execution
    public static void main(String[] args) {
        // Starting the execution of the application.
        // System.in denotes the standard input (i.e., keyboard).
        Scanner input = new Scanner(System.in);

        // Start the actual application code here.

        /*
         * Calculate the first circle.
         */
        // Step 1: Prompt the user for the radius of the 1st circle.
        System.out.println("Enter the radius of the 1st circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius1 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.
        double area1 = 3.14 * radius1 * radius1;
        String area1s = String.format("%.2f", area1);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area1s);

        /*
         * Calculate the second circle.
         */
        // Step 1: Prompt the user for the radius of the 2nd circle.
        System.out.println("Enter the radius of the 2nd circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius2 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.
        double area2 = 3.14 * radius1 * radius2;
        String area2s = String.format("%.2f", area2);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area2s);

        // end of the application code.

        input.close();
    }
}
```

input double



Utility Methods

Utility Method (can call the method by using its containing class directly)



```
public class Circle {
    /*
     * Utility methods: not requiring creating of objects in order to use them.
     */
    public static double getArea(double radius) {
        double area = 0.0;
        area = 3.14 * radius * radius;
        return area;
    }
}
```

output/return type

parameter

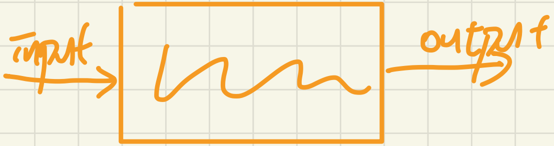
input type

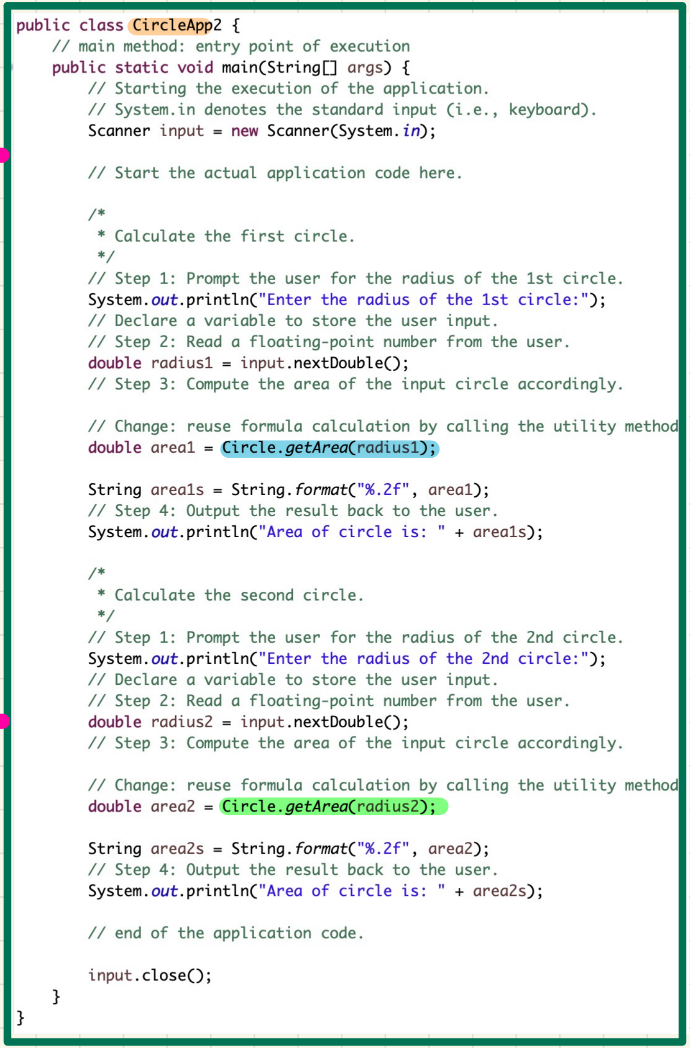
produce the output of type double.

Circle.getArea(...)

use

use

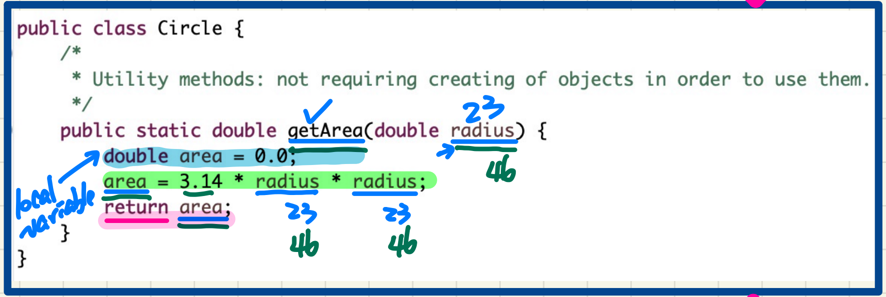




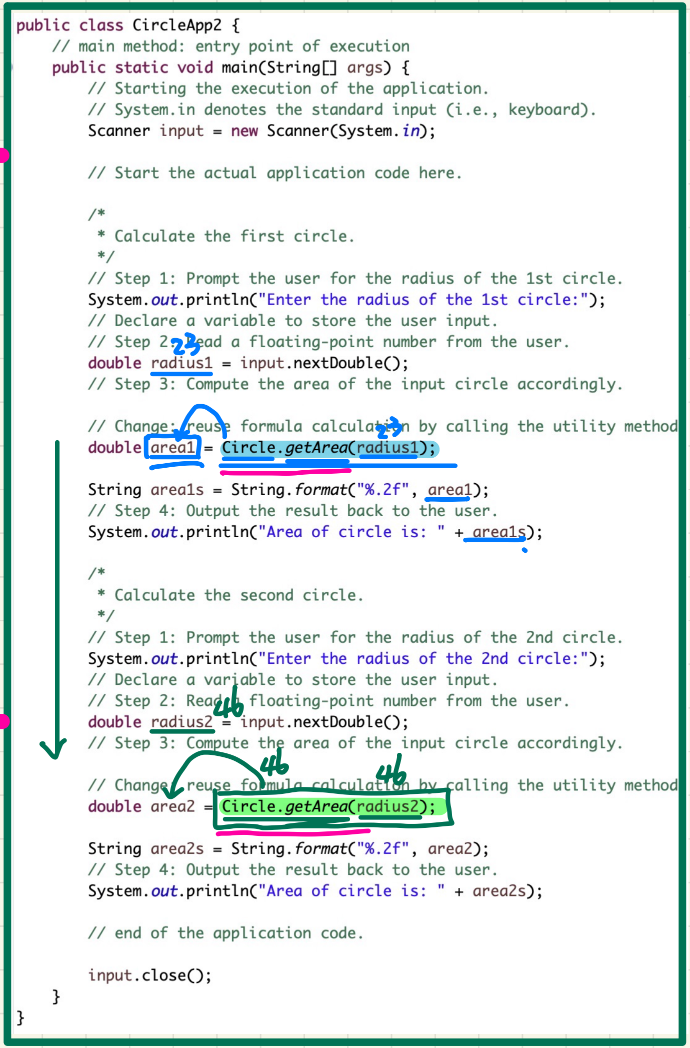
```
public class CircleApp2 {
    // main method: entry point of execution
    public static void main(String[] args) {
        // Starting the execution of the application.
        // System.in denotes the standard input (i.e., keyboard).
        Scanner input = new Scanner(System.in);

        // Start the actual application code here.

        /*
         * Calculate the first circle.
         */
        // Step 1: Prompt the user for the radius of the 1st circle.
        System.out.println("Enter the radius of the 1st circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius1 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.

        // Change: reuse formula calculation by calling the utility method
        double area1 = Circle.getArea(radius1);

        String area1s = String.format("%.2f", area1);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area1s);

        /*
         * Calculate the second circle.
         */
        // Step 1: Prompt the user for the radius of the 2nd circle.
        System.out.println("Enter the radius of the 2nd circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius2 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.

        // Change: reuse formula calculation by calling the utility method
        double area2 = Circle.getArea(radius2);

        String area2s = String.format("%.2f", area2);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area2s);

        // end of the application code.

        input.close();
    }
}
```

Calling Utility Methods

use



```
public class Circle {
    /*
     * Utility methods: not requiring creating of objects in order to use them.
     */
    public static double getArea(double radius) {
        double area = 0.0;
        area = 3.14 * radius * radius;
        return area;
    }
}
```



```
public class CircleApp2 {
    // main method: entry point of execution
    public static void main(String[] args) {
        // Starting the execution of the application.
        // System.in denotes the standard input (i.e., keyboard).
        Scanner input = new Scanner(System.in);

        // Start the actual application code here.

        /*
         * Calculate the first circle.
         */
        // Step 1: Prompt the user for the radius of the 1st circle.
        System.out.println("Enter the radius of the 1st circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius1 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.

        // Change: reuse formula calculation by calling the utility method
        double area1 = Circle.getArea(radius1);

        String area1s = String.format("%.2f", area1);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area1s);

        /*
         * Calculate the second circle.
         */
        // Step 1: Prompt the user for the radius of the 2nd circle.
        System.out.println("Enter the radius of the 2nd circle:");
        // Declare a variable to store the user input.
        // Step 2: Read a floating-point number from the user.
        double radius2 = input.nextDouble();
        // Step 3: Compute the area of the input circle accordingly.

        // Change: reuse formula calculation by calling the utility method
        double area2 = Circle.getArea(radius2);

        String area2s = String.format("%.2f", area2);
        // Step 4: Output the result back to the user.
        System.out.println("Area of circle is: " + area2s);

        // end of the application code.

        input.close();
    }
}
```

Utility Method

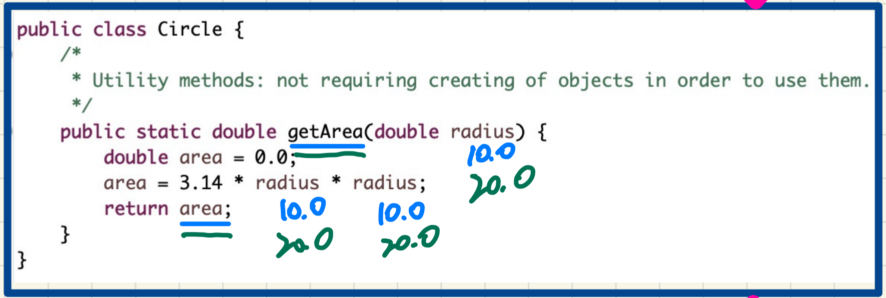
1. Declare local variables

2. Compute the output using input parameters & local var.

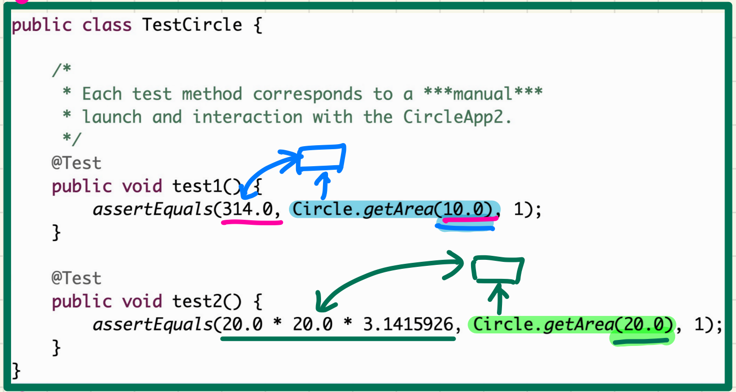
3. return result

Junit Tests (automated testing of utility methods)

use



```
public class Circle {
    /*
     * Utility methods: not requiring creating of objects in order to use them.
     */
    public static double getArea(double radius) {
        double area = 0.0;
        area = 3.14 * radius * radius;
        return area;
    }
}
```



```
public class TestCircle {

    /*
     * Each test method corresponds to a ***manual***
     * launch and interaction with the CircleApp2.
     */
    @Test
    public void test1() {
        assertEquals(314.0, Circle.getArea(10.0), 1);
    }

    @Test
    public void test2() {
        assertEquals(20.0 * 20.0 * 3.1415926, Circle.getArea(20.0), 1);
    }
}
```

ConsoleApp:

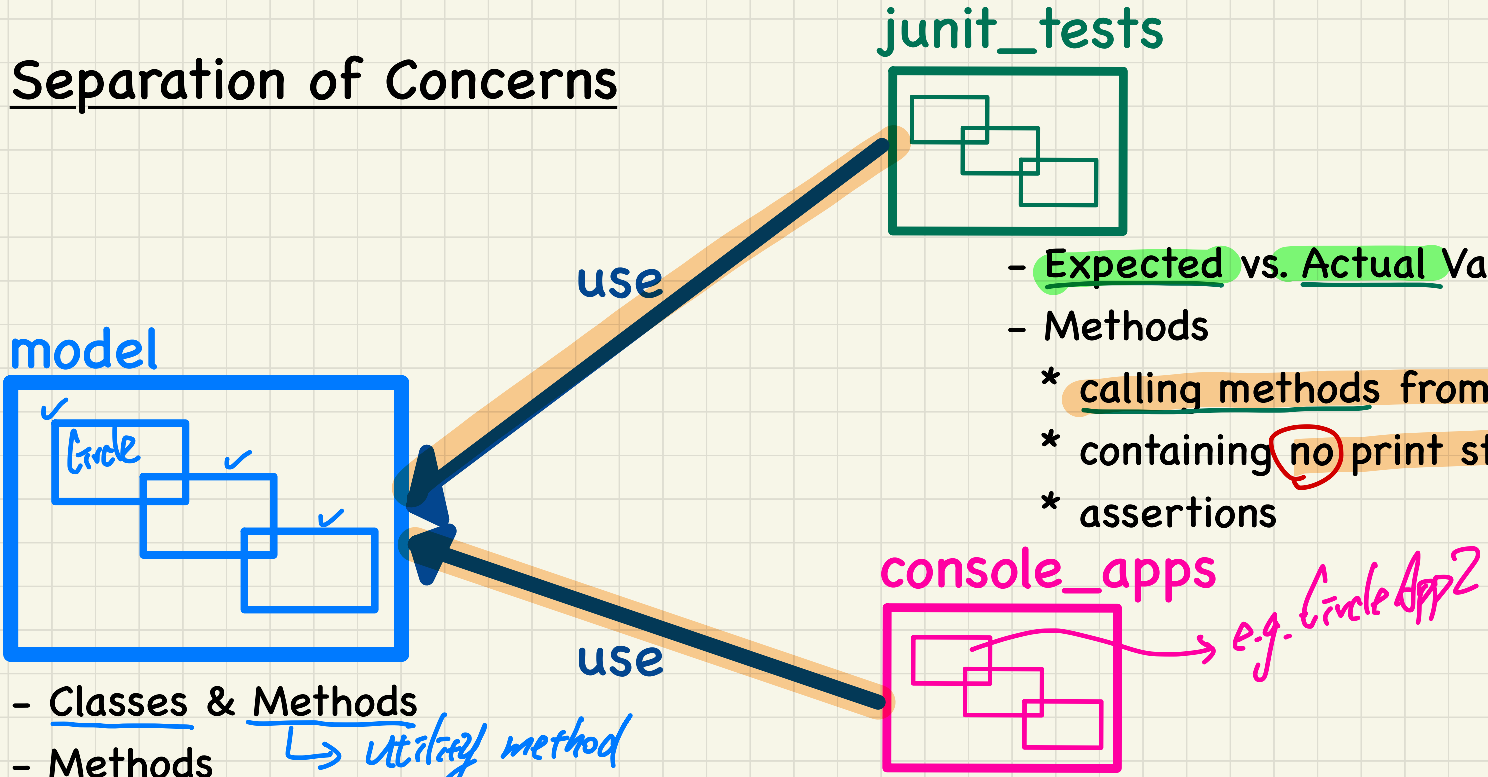
Junit Tests

call utility methods

compute and print values to console. (manual)

compare return values against the expected values (automated).

Separation of Concerns



junit_tests

- Expected vs. Actual Values
- Methods
  * calling methods from model
  * containing no print statements
  * assertions

model

- Classes & Methods
  ↳ utility method
- Methods
  * containing no print statements
  * return statements

console_apps

- main method
  * calling methods from model
  * containing print statements
  * containg no return statements

EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 2

Use of Debugger and Introduction to Conditionals

Arithmetic Sequence

$$\underbrace{3}_{FT},\ \xrightarrow{+2} 5,\ \xrightarrow{+2} 7,\ \xrightarrow{+2} 9,\ \xrightarrow{+2} 11$$

# of terms : 5

- First Term (FT)
- Common Difference (d)
- Number of Terms
- Sum

1st: FT

2nd: FT + 1d

3rd: FT + 2d

4th: FT + 3d

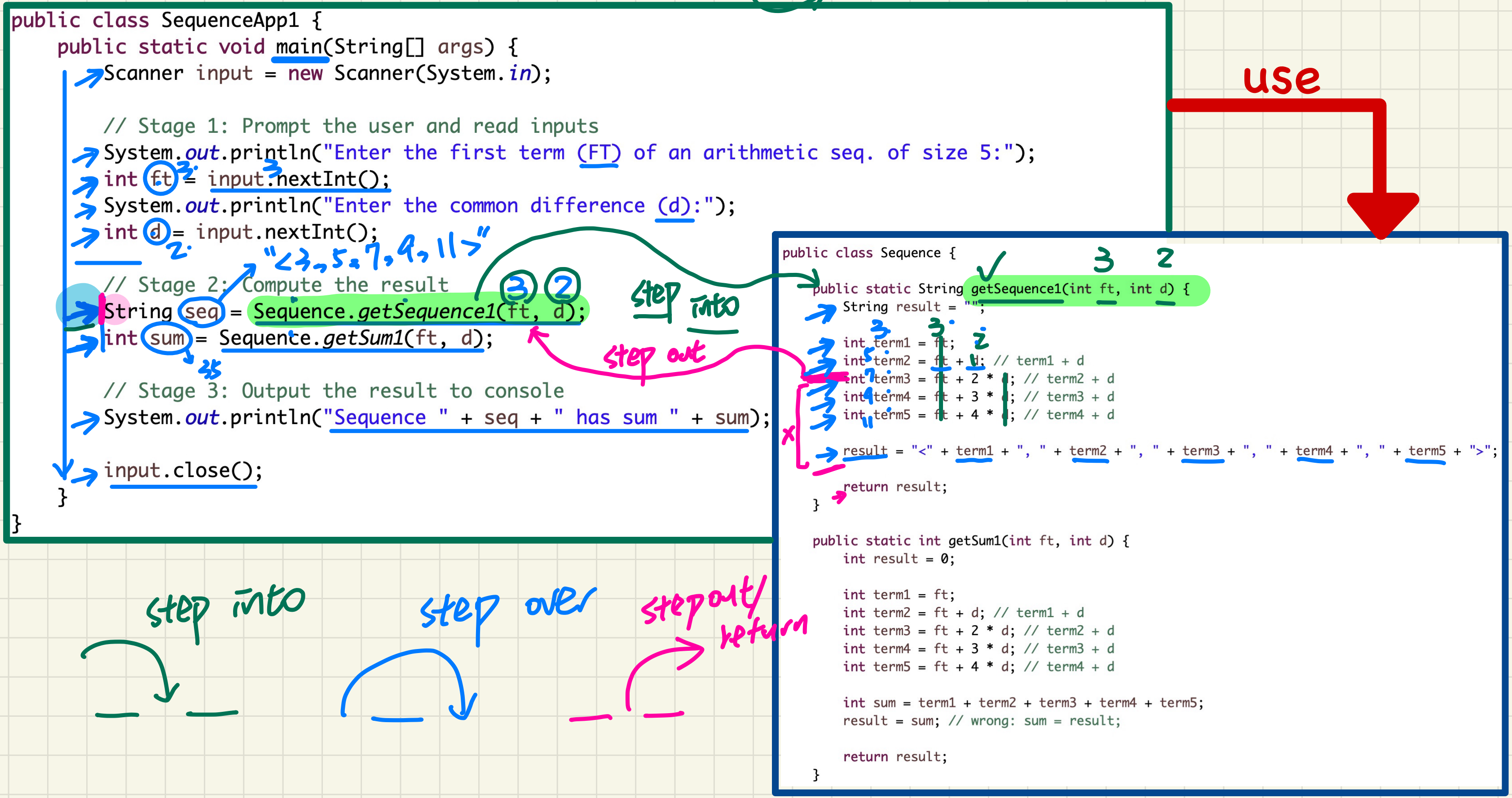
5th: FT + 4d

# of terms − 1

$$5 * \underbrace{FT}_{3} + \underbrace{(1+2+3+4)}_{10} * \underbrace{d}_{2}$$

$$15 + 20 = 35$$

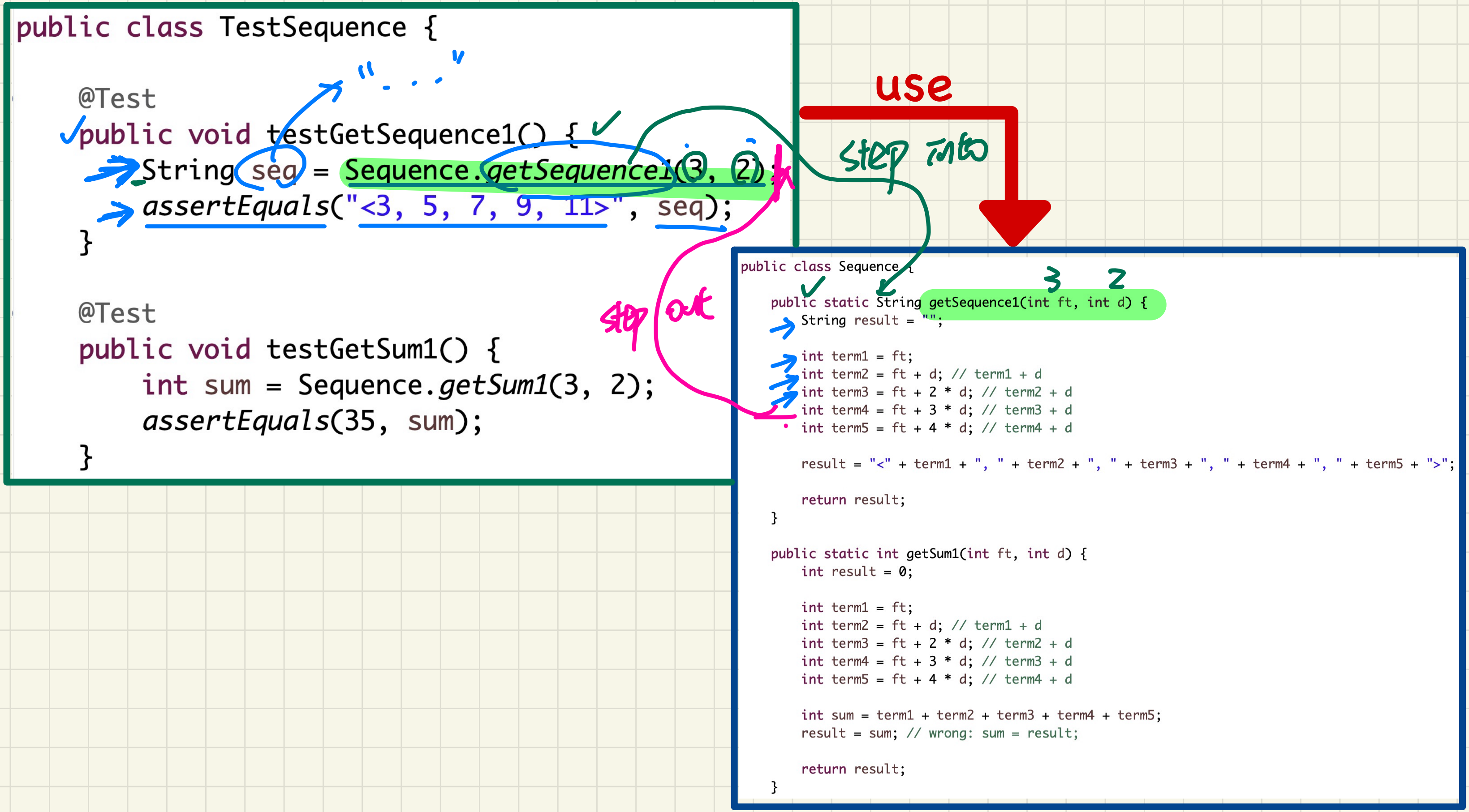
Debugger: Step Over, Step Into, Step Out



```java
public class SequenceApp1 {
    public static void main(String[] args) {
        Scanner input = new Scanner(System.in);

        // Stage 1: Prompt the user and read inputs
        System.out.println("Enter the first term (FT) of an arithmetic seq. of size 5:");
        int ft = input.nextInt();
        System.out.println("Enter the common difference (d):");
        int d = input.nextInt();

        // Stage 2: Compute the result
        String seq = Sequence.getSequence1(ft, d);
        int sum = Sequence.getSum1(ft, d);

        // Stage 3: Output the result to console
        System.out.println("Sequence " + seq + " has sum " + sum);

        input.close();
    }
}
```

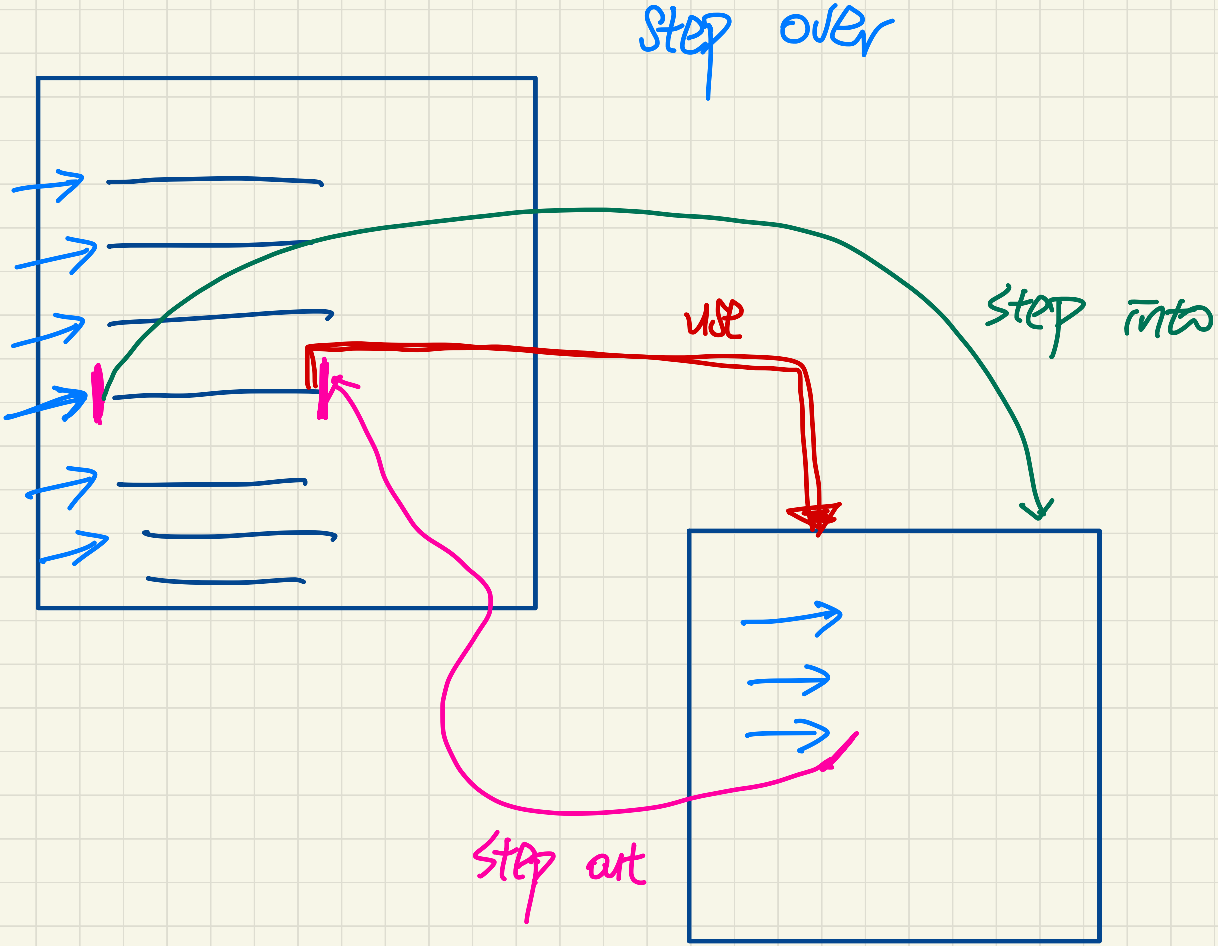
```java
public class Sequence {
    public static String getSequence1(int ft, int d) {
        String result = "";

        int term1 = ft;
        int term2 = ft + d; // term1 + d
        int term3 = ft + 2 * d; // term2 + d
        int term4 = ft + 3 * d; // term3 + d
        int term5 = ft + 4 * d; // term4 + d

        result = "<" + term1 + ", " + term2 + ", " + term3 + ", " + term4 + ", " + term5 + ">";

        return result;
    }

    public static int getSum1(int ft, int d) {
        int result = 0;

        int term1 = ft;
        int term2 = ft + d; // term1 + d
        int term3 = ft + 2 * d; // term2 + d
        int term4 = ft + 3 * d; // term3 + d
        int term5 = ft + 4 * d; // term4 + d

        int sum = term1 + term2 + term3 + term4 + term5;
        result = sum; // wrong: sum = result;

        return result;
    }
}
```

Debugger: Step Over, Step Into, Step Out



```java
public class TestSequence {

    @Test
    public void testGetSequence1() {
        String seq = Sequence.getSequence1(3, 2);
        assertEquals("<3, 5, 7, 9, 11>", seq);
    }

    @Test
    public void testGetSum1() {
        int sum = Sequence.getSum1(3, 2);
        assertEquals(35, sum);
    }
```

```java
public class Sequence {
    public static String getSequence1(int ft, int d) {
        String result = "";

        int term1 = ft;
        int term2 = ft + d; // term1 + d
        int term3 = ft + 2 * d; // term2 + d
        int term4 = ft + 3 * d; // term3 + d
        int term5 = ft + 4 * d; // term4 + d

        result = "<" + term1 + ", " + term2 + ", " + term3 + ", " + term4 + ", " + term5 + ">";

        return result;
    }

    public static int getSum1(int ft, int d) {
        int result = 0;

        int term1 = ft;
        int term2 = ft + d; // term1 + d
        int term3 = ft + 2 * d; // term2 + d
        int term4 = ft + 3 * d; // term3 + d
        int term5 = ft + 4 * d; // term4 + d

        int sum = term1 + term2 + term3 + term4 + term5;
        result = sum; // wrong: sum = result;

        return result;
    }
```

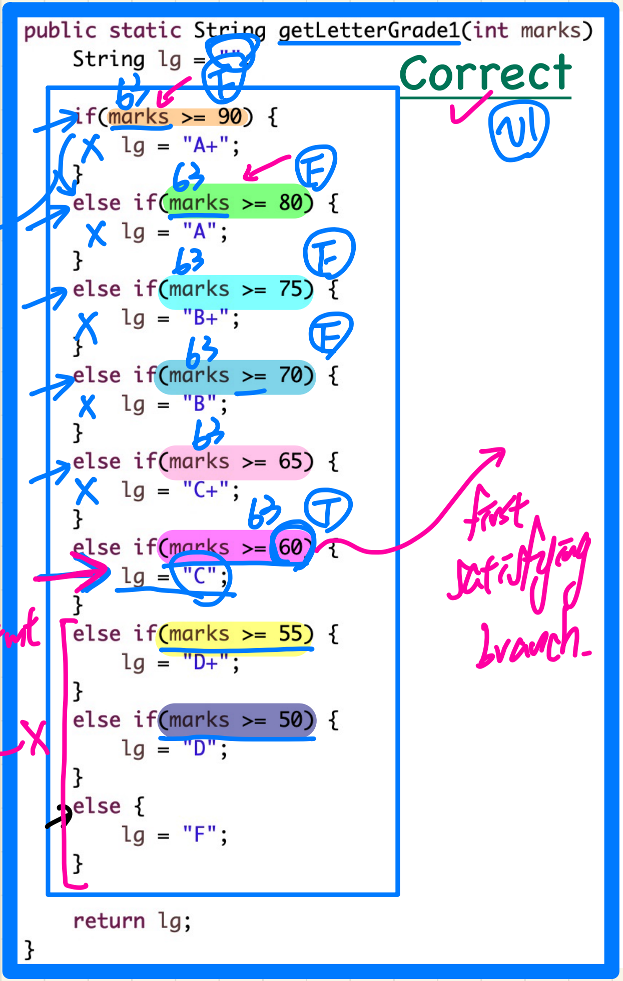

step over
use
step into
step out

EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 3

Exploration of
Logical Operations and Nested Conditionals

Single If-Stmt with Overlapping Conditions



```
public static String getLetterGrade1(int marks)
    String lg = "";
    if(marks >= 90) {
        lg = "A+";
    }
    else if(marks >= 80) {
        lg = "A";
    }
    else if(marks >= 75) {
        lg = "B+";
    }
    else if(marks >= 70) {
        lg = "B";
    }
    else if(marks >= 65) {
        lg = "C+";
    }
    else if(marks >= 60) {
        lg = "C";
    }
    else if(marks >= 55) {
        lg = "D+";
    }
    else if(marks >= 50) {
        lg = "D";
    }
    else {
        lg = "F";
    }

    return lg;
}
```

Correct ✓ vl

single if-stmt.

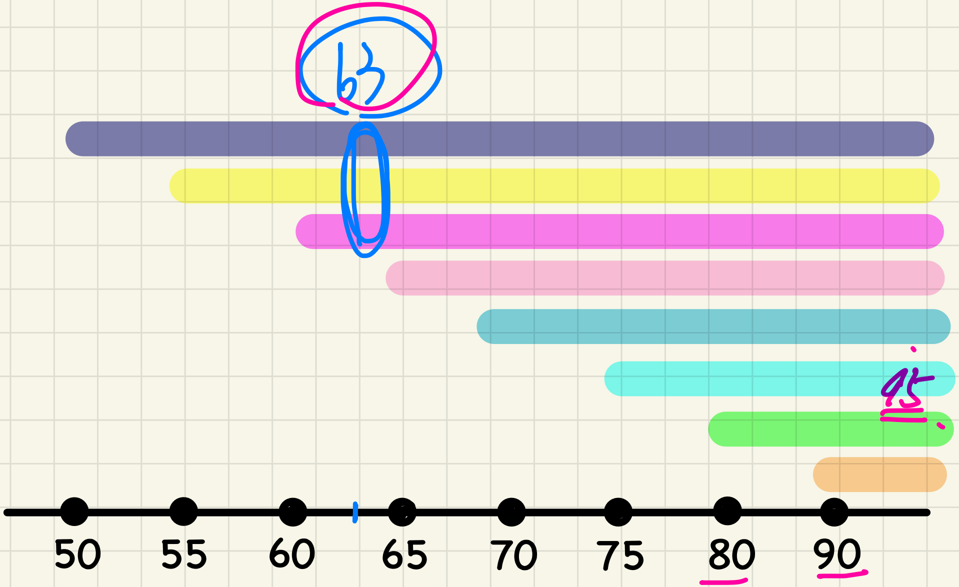
first satisfying branch.

rest of if-stmt bypassed. X

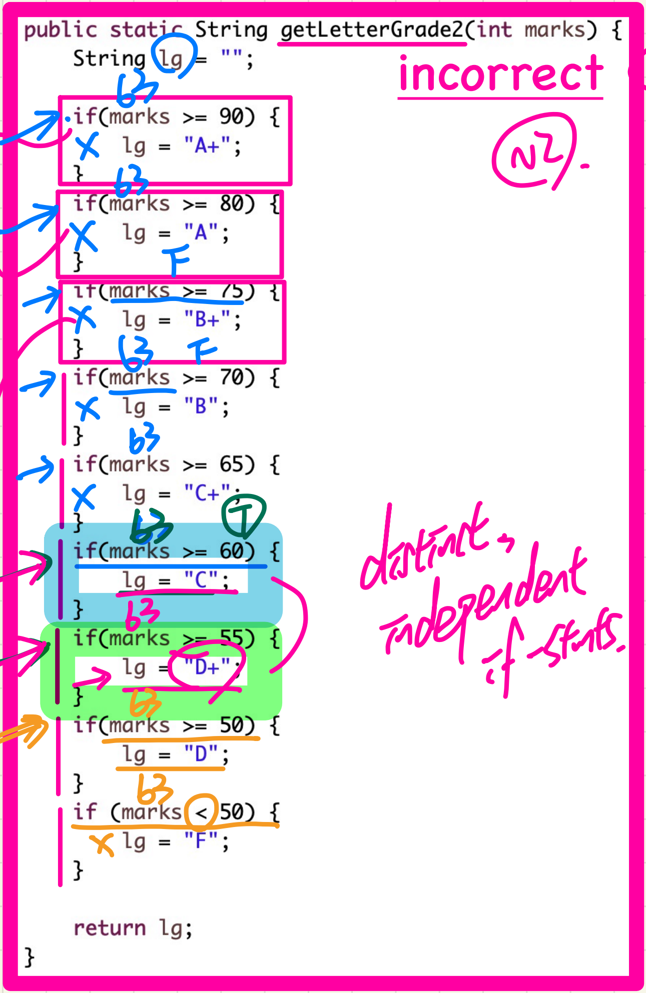
lg "C" ~~"X"~~

Test Value:

marks = 63



Multiple If-Stmts with Overlapping Conditions



```
public static String getLetterGrade2(int marks) {
    String lg = "";
    if(marks >= 90) {
        lg = "A+";
    }
    if(marks >= 80) {
        lg = "A";
    }
    if(marks >= 75) {
        lg = "B+";
    }
    if(marks >= 70) {
        lg = "B";
    }
    if(marks >= 65) {
        lg = "C+";
    }
    if(marks >= 60) {
        lg = "C";
    }
    if(marks >= 55) {
        lg = "D+";
    }
    if(marks >= 50) {
        lg = "D";
    }
    if (marks < 50) {
        lg = "F";
    }

    return lg;
}
```

incorrect

N2.

1st

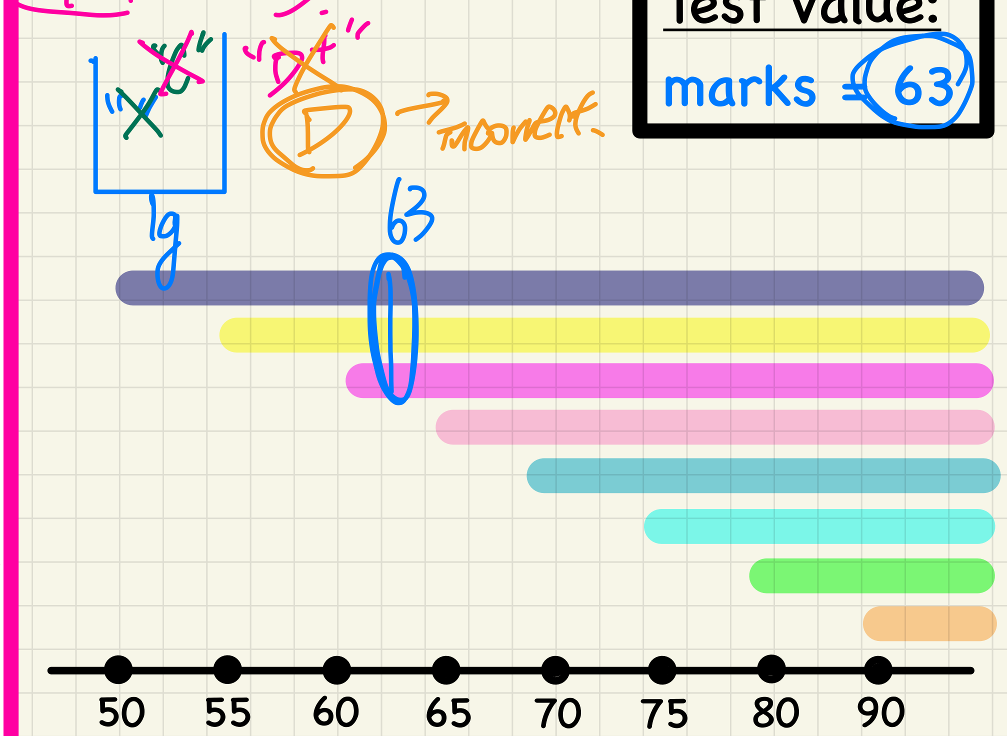
2nd

3rd

63 63 F 63 F 63 63 T 63 63 63

distinct, independent if-stmts.

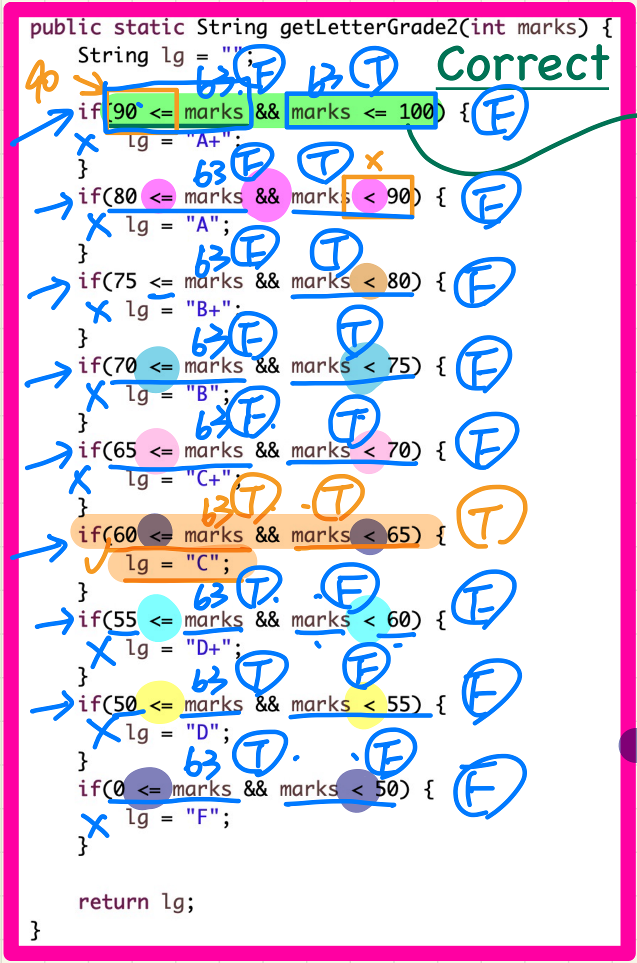
4 if-stmts



Test Value:

marks = 63

Multiple If-Stmts with Non-Overlapping Conditions



```
public static String getLetterGrade2(int marks) {
    String lg = "";
    if(90 <= marks && marks <= 100) {
        lg = "A+";
    }
    if(80 <= marks && marks < 90) {
        lg = "A";
    }
    if(75 <= marks && marks < 80) {
        lg = "B+";
    }
    if(70 <= marks && marks < 75) {
        lg = "B";
    }
    if(65 <= marks && marks < 70) {
        lg = "C+";
    }
    if(60 <= marks && marks < 65) {
        lg = "C";
    }
    if(55 <= marks && marks < 60) {
        lg = "D+";
    }
    if(50 <= marks && marks < 55) {
        lg = "D";
    }
    if(0 <= marks && marks < 50) {
        lg = "F";
    }

    return lg;
}
```

(1B)

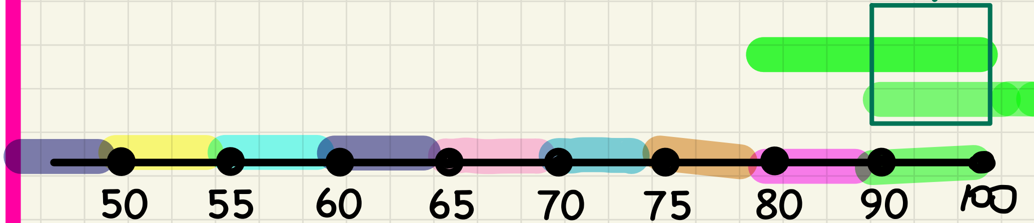
$90 \le marks \le 100$

marks $\geq$ 90

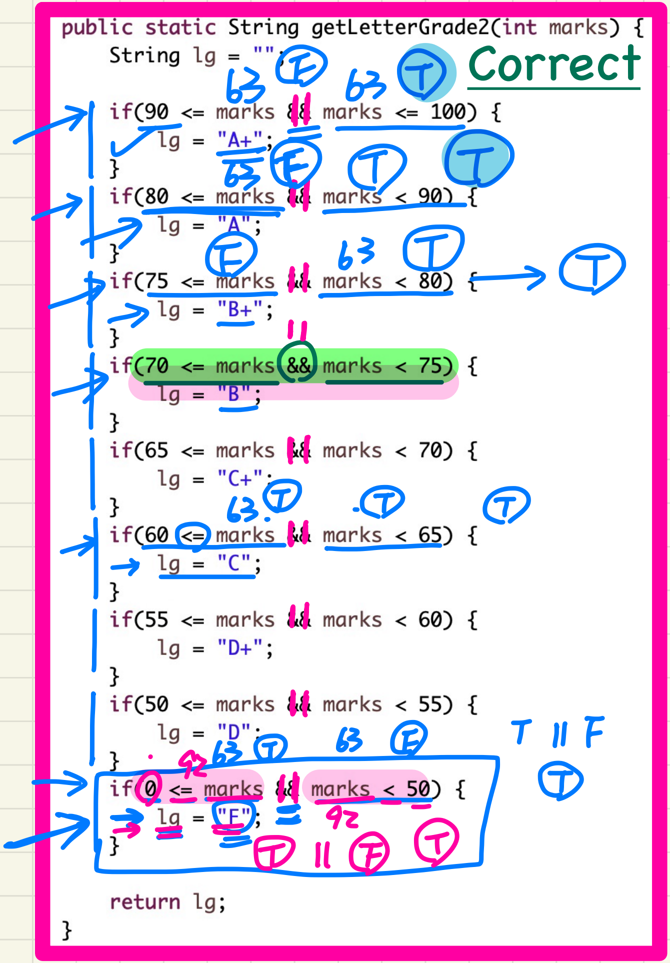
&& marks $\le$ 100

Test Value:

marks = 63



Multiple If-Stmts with Non-Overlapping Conditions (v3)



Correct

```
public static String getLetterGrade2(int marks) {
    String lg = "";
    if(90 <= marks && marks <= 100) {
        lg = "A+";
    }
    if(80 <= marks && marks < 90) {
        lg = "A";
    }
    if(75 <= marks && marks < 80) {
        lg = "B+";
    }
    if(70 <= marks && marks < 75) {
        lg = "B";
    }
    if(65 <= marks && marks < 70) {
        lg = "C+";
    }
    if(60 <= marks && marks < 65) {
        lg = "C";
    }
    if(55 <= marks && marks < 60) {
        lg = "D+";
    }
    if(50 <= marks && marks < 55) {
        lg = "D";
    }
    if(0 <= marks && marks < 50) {
        lg = "F";
    }

    return lg;
}
```

9 independent if-stmts

v4

Test Value:

marks = 63

- marks >= 70
- &&
- marks < 75

"F" &

eg

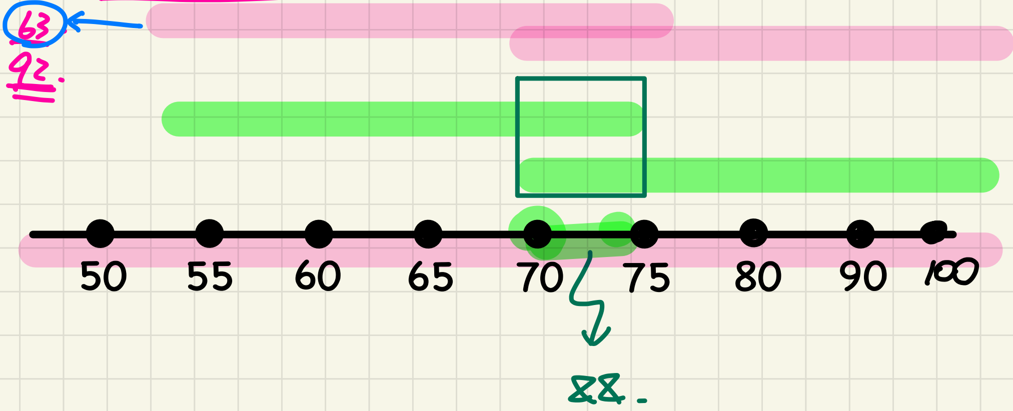
- marks >= 70
- ||
- marks < 75

63

92

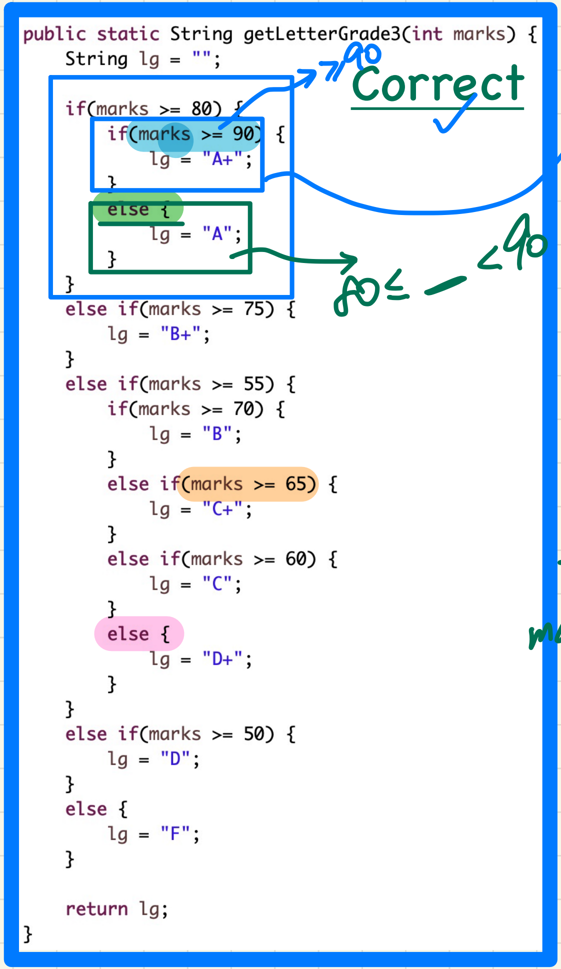
~~A~~ ~~B+~~ B

~~A+~~



Nested If-Stmts

N5



```
public static String getLetterGrade3(int marks) {
    String lg = "";

    if(marks >= 80) {
        if(marks >= 90) {
            lg = "A+";
        }
        else {
            lg = "A";
        }
    }
    else if(marks >= 75) {
        lg = "B+";
    }
    else if(marks >= 55) {
        if(marks >= 70) {
            lg = "B";
        }
        else if(marks >= 65) {
            lg = "C+";
        }
        else if(marks >= 60) {
            lg = "C";
        }
        else {
            lg = "D+";
        }
    }
    else if(marks >= 50) {
        lg = "D";
    }
    else {
        lg = "F";
    }

    return lg;
}
```

90

Correct ✓

80 ≤ — < 90

```
if (marks >= 80) {        T
    if (marks >= 90) {
        lg = "A+";
    }
    else {
        lg = "A";
    }
}
```

marks < 90 && marks >= 80

marks >= 80 && marks >= 90

marks >= 90 (F)

!(marks >= 90) (T)

marks < 90

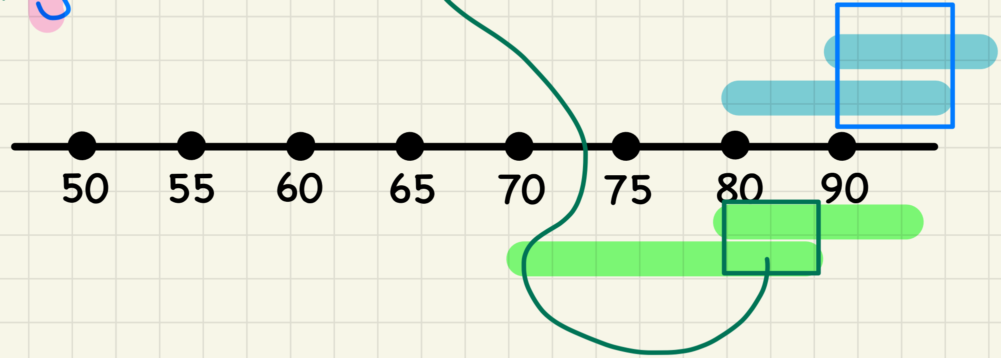
Test Value:

marks = 93

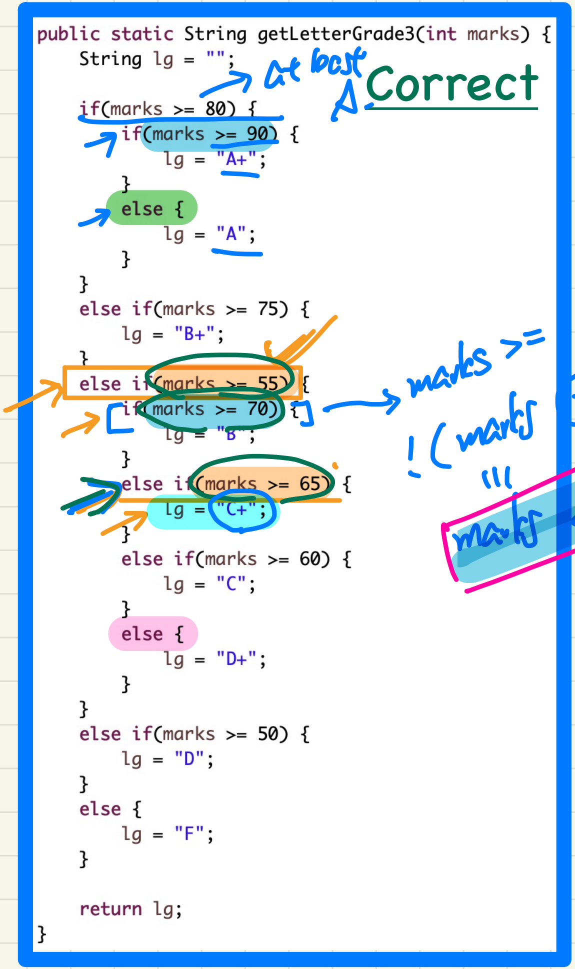
marks = 86

marks = 67

marks = 52



Nested If-Stmts



```
public static String getLetterGrade3(int marks) {
    String lg = "";

    if(marks >= 80) {
        if(marks >= 90) {
            lg = "A+";
        }
        else {
            lg = "A";
        }
    }
    else if(marks >= 75) {
        lg = "B+";
    }
    else if(marks >= 55) {
        if(marks >= 70) {
            lg = "B";
        }
        else if(marks >= 65) {
            lg = "C+";
        }
        else if(marks >= 60) {
            lg = "C";
        }
        else {
            lg = "D+";
        }
    }
    else if(marks >= 50) {
        lg = "D";
    }
    else {
        lg = "F";
    }

    return lg;
}
```

at least Correct

→ marks >= 70 F

!(marks >= 70) T

||

marks < 70

marks < 70

&&

marks >= 55

&&

marks >= 65

65 <= _ < 70

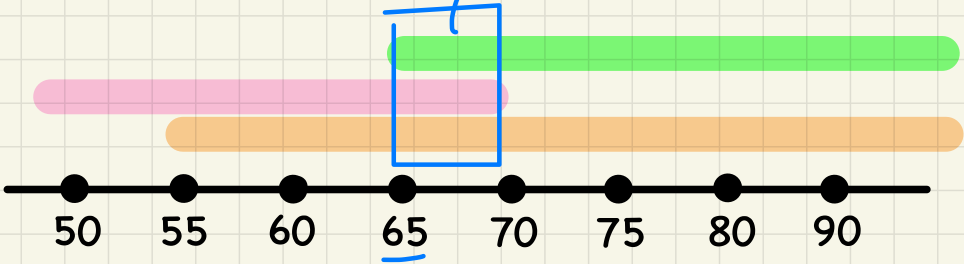
Test Value:

marks = 93

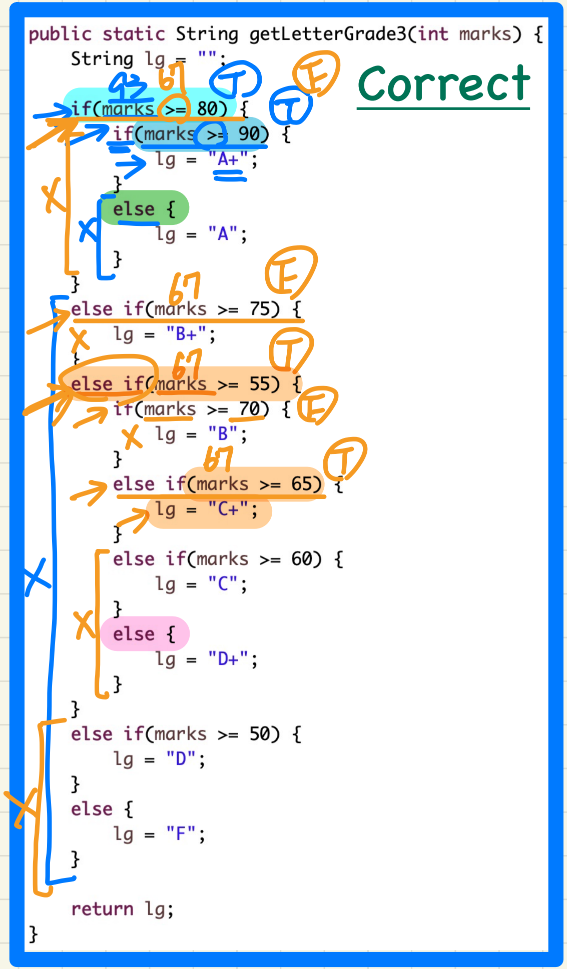
marks = 86

marks = 67

marks = 52



Nested If-Stmts



```
public static String getLetterGrade3(int marks) {
    String lg = "";
    if(marks >= 80) {
        if(marks >= 90) {
            lg = "A+";
        }
        else {
            lg = "A";
        }
    }
    else if(marks >= 75) {
        lg = "B+";
    }
    else if(marks >= 55) {
        if(marks >= 70) {
            lg = "B";
        }
        else if(marks >= 65) {
            lg = "C+";
        }
        else if(marks >= 60) {
            lg = "C";
        }
        else {
            lg = "D+";
        }
    }
    else if(marks >= 50) {
        lg = "D";
    }
    else {
        lg = "F";
    }

    return lg;
}
```

Correct

Test Value:

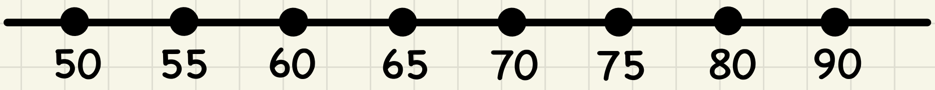
A+ marks = 93

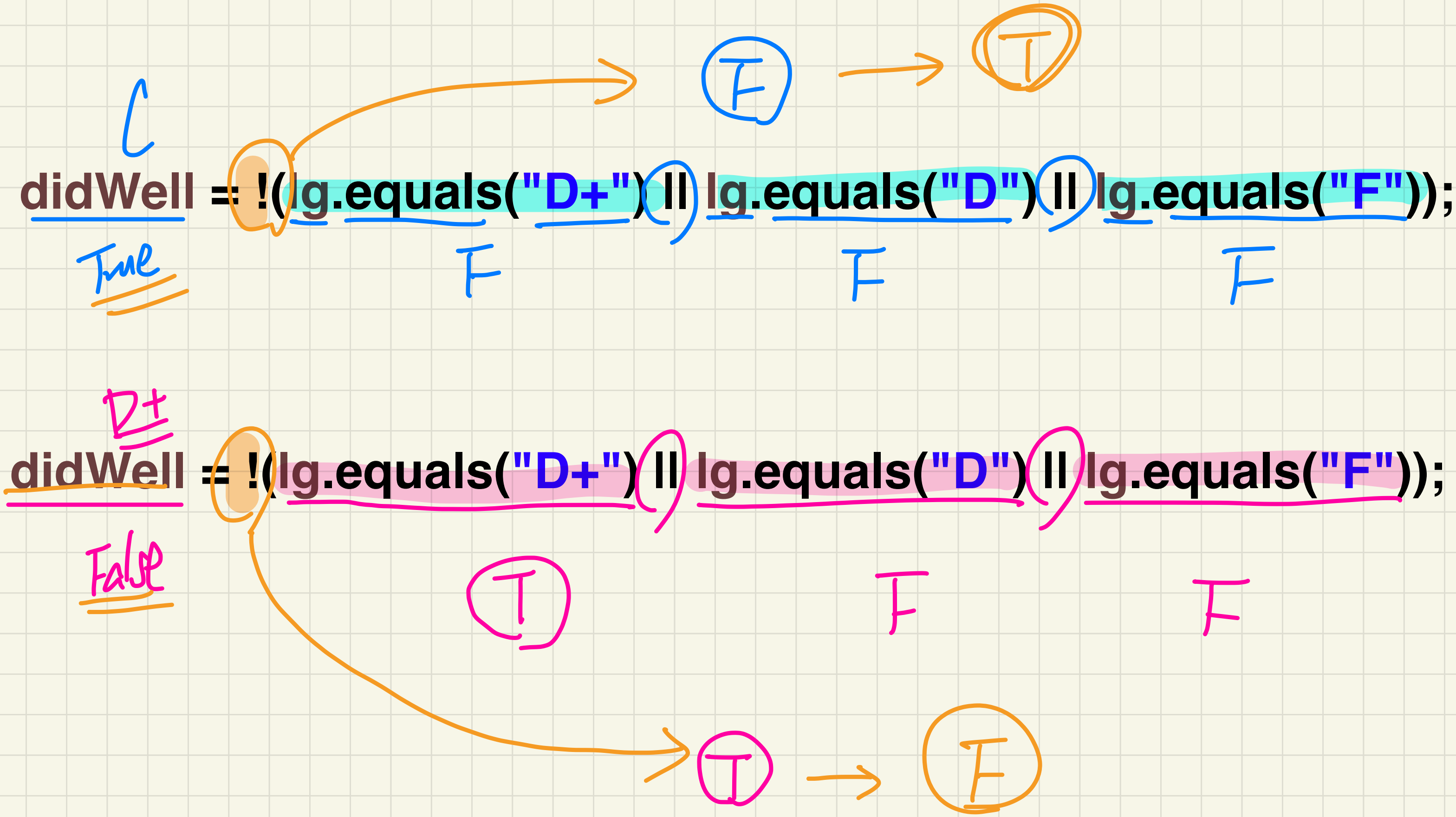
marks = 86

C+ marks = 67

marks = 52

Trace : Exercise





```
didWell = !(lg.equals("D+") || lg.equals("D") || lg.equals("F"));
```

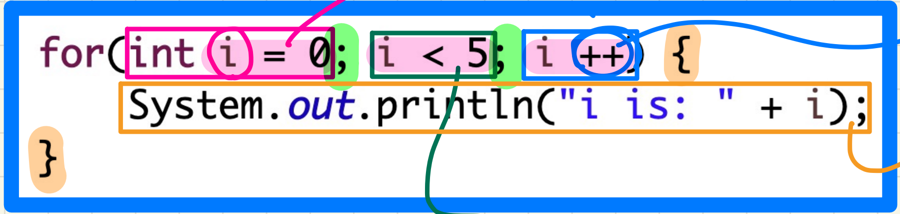
```
didWell = !(lg.equals("D+") || lg.equals("D") || lg.equals("F"));
```

EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 4

Introducing Loops to Patternize Repetitive Actions

for-Loop Syntax



```
for(int i = 0; i < 5; i++) {
    System.out.println("i is: " + i);
}
```

initialization
↳ declaring a loop counter.

update the loop counter. (ensure termination)

body of loop.

stay condition Boolean expression.

↳ as long as "i< 5" evaluates to true, keep executing the body implementation of loop.

termination

for-Loop: Tracing

Iteration.

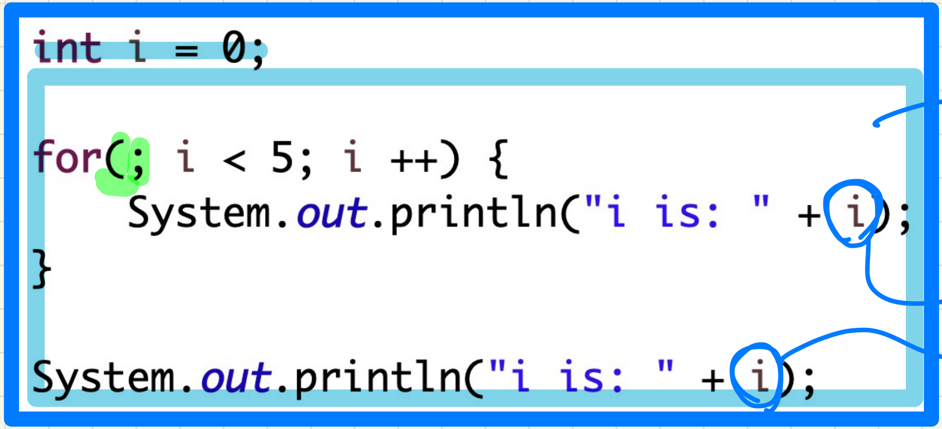
```
for(int i = 0; i < 5; i ++) {
  System.out.println("i is:  " + i);
}
```

i value	Stay Condition	Stage	Action
0	0 < 5 T	It. # 1	print i i++
1	1 < 5 T	It. #2	print i i++
2.	2 < 5 T	It. #3	print i i++
3	3 < 5 T	It. #4	print i i++
4.	4 < 5 T	It#5	print i I++
5	5 < 5 F		

Console

```
i is 0
i is 1
i is 2
i is 3
i is 4
```

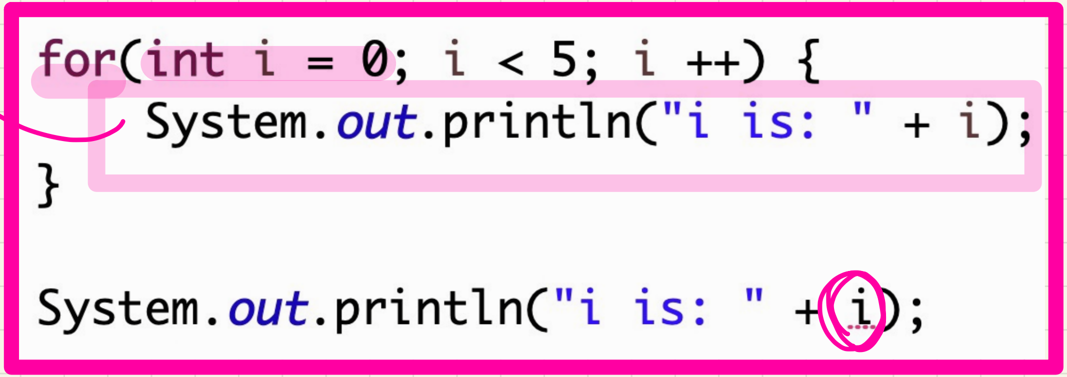
for-Loop: Scope of Loop Counter



```
int i = 0;

for(; i < 5; i ++) {
    System.out.println("i is: " + i);
}

System.out.println("i is: " + i);
```

scope of loop counter i

within the scope of i

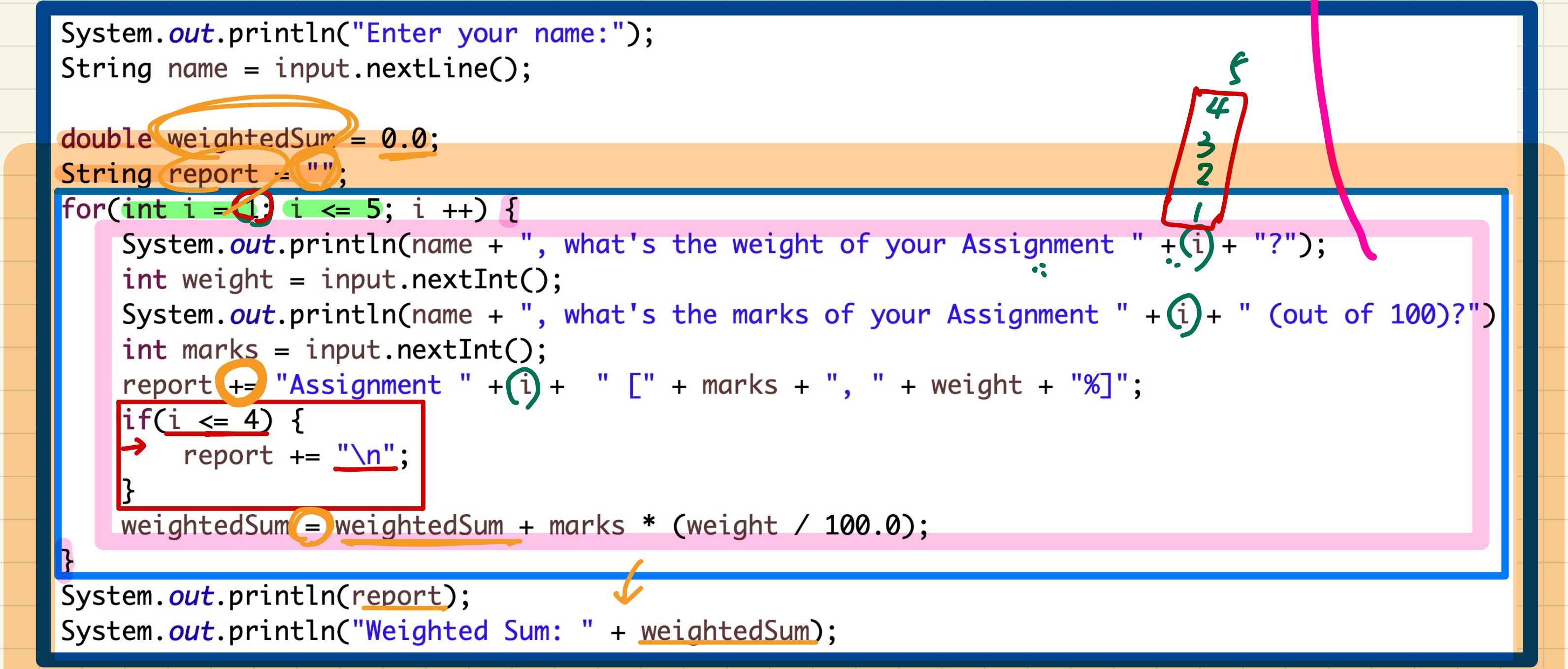


```
for(int i = 0; i < 5; i ++) {
    System.out.println("i is: " + i);
}

System.out.println("i is: " + i);
```

scope of loop counter i

Grade Calculator using a for-Loop

body of loop.

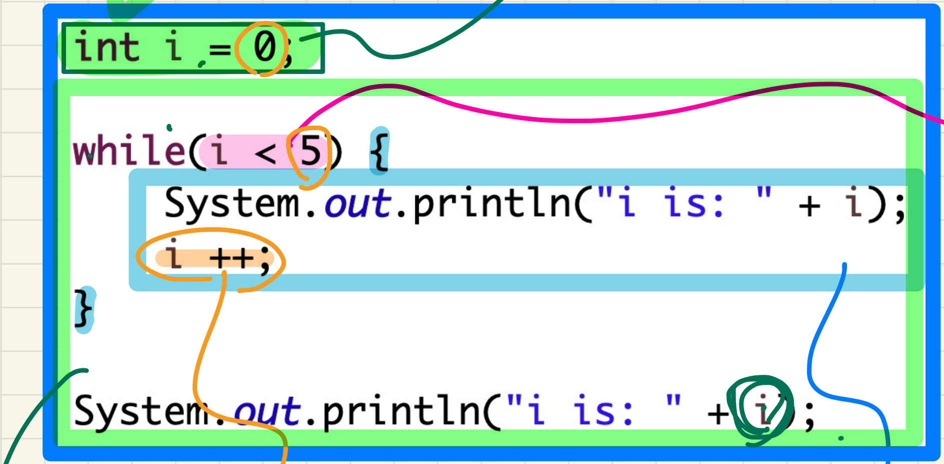


```
System.out.println("Enter your name:");
String name = input.nextLine();

double weightedSum = 0.0;
String report = "";
for(int i = 1; i <= 5; i ++) {
    System.out.println(name + ", what's the weight of your Assignment " + i + "?");
    int weight = input.nextInt();
    System.out.println(name + ", what's the marks of your Assignment " + i + " (out of 100)?")
    int marks = input.nextInt();
    report += "Assignment " + i +  " [" + marks + ", " + weight + "%]";
    if(i <= 4) {
        report += "\n";
    }
    weightedSum = weightedSum + marks * (weight / 100.0);
}
System.out.println(report);
System.out.println("Weighted Sum: " + weightedSum);
```

5
4
3
2
1

scope of weightedSum, report

while-Loop Syntax



```
int i = 0;

while(i < 5) {
    System.out.println("i is: " + i);
    i ++;
}

System.out.println("i is: " + i);
```

declare and initialize loop counter

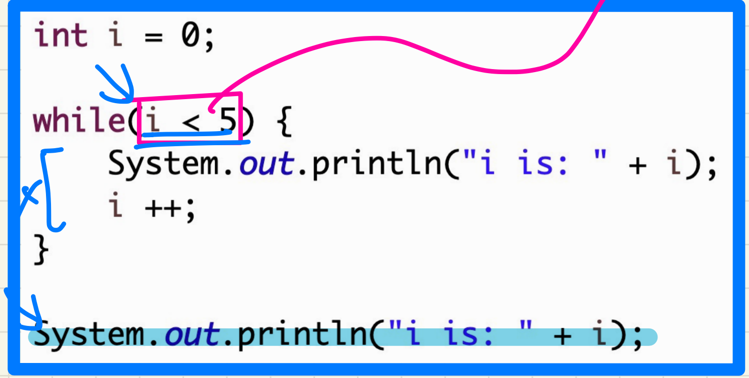
stay condition
↳ as long as it evaluates to true, execute another iteration.

update loop counter (ensure termination)

body of loop.

Scope of loop counter i.

while-Loop: Tracing



```
int i = 0;

while(i < 5) {
    System.out.println("i is: " + i);
    i ++;
}

System.out.println("i is: " + i);
```

when evaluating to false, exit.

i value	Stay Condition	Stage	Action
0	0 < 5 T	It. # 1	print i i++
1	1 < 5 T	It. # 2	print i i++
2.	2 < 5 T	It. # 3	print i i++
3	3 < 5 T	It. # 4	print i i++
4.	4 < 5 T	It # 5	print i i++
5	5 < 5 F		

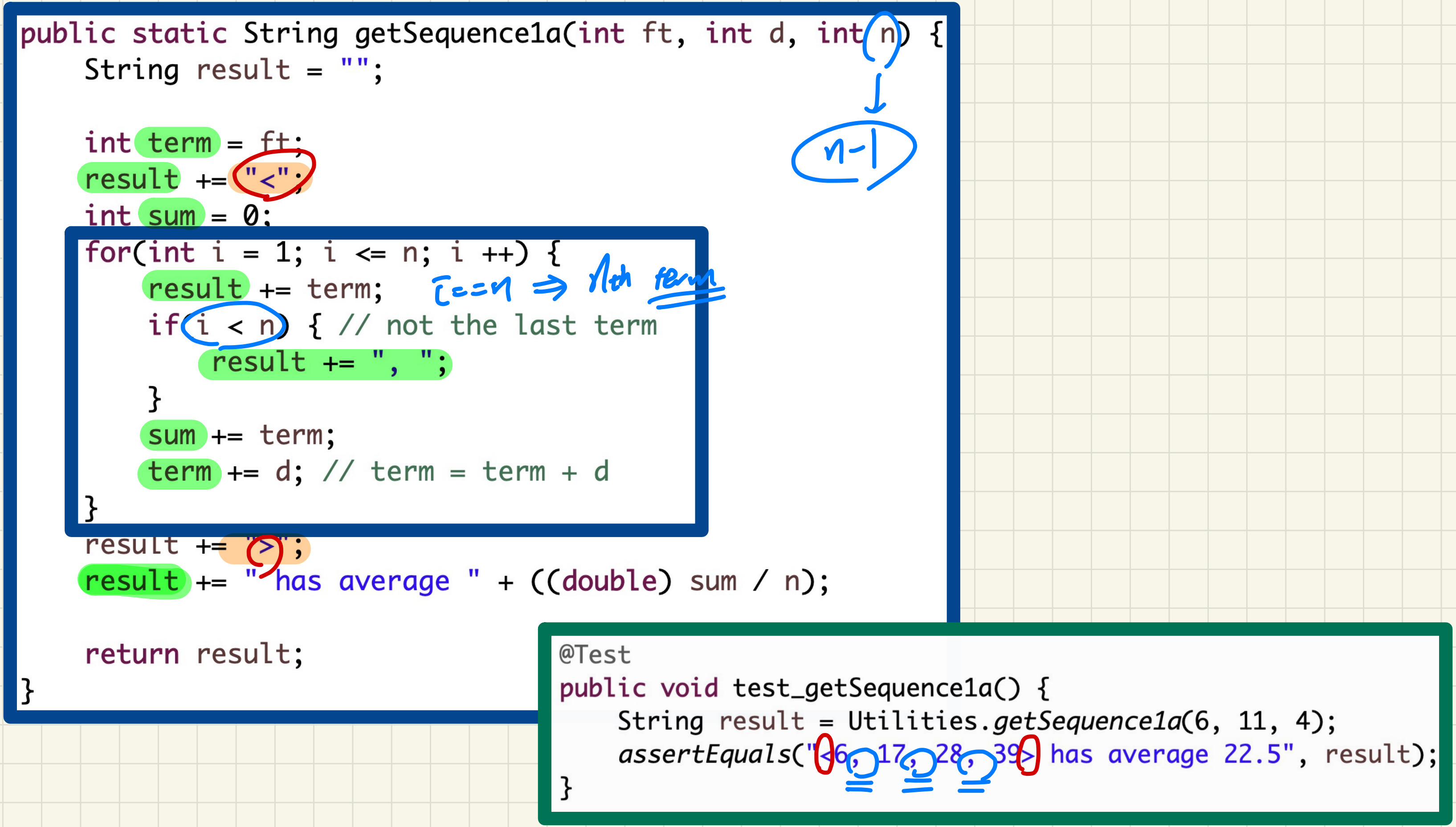
Console

```
i is 0
i is 1
i is 2
i is 3
i is 4
i is 5
```

printed from 5 iterations.

(after exit) executed outside while loop

Get Arithmetic Sequence using a for-Loop



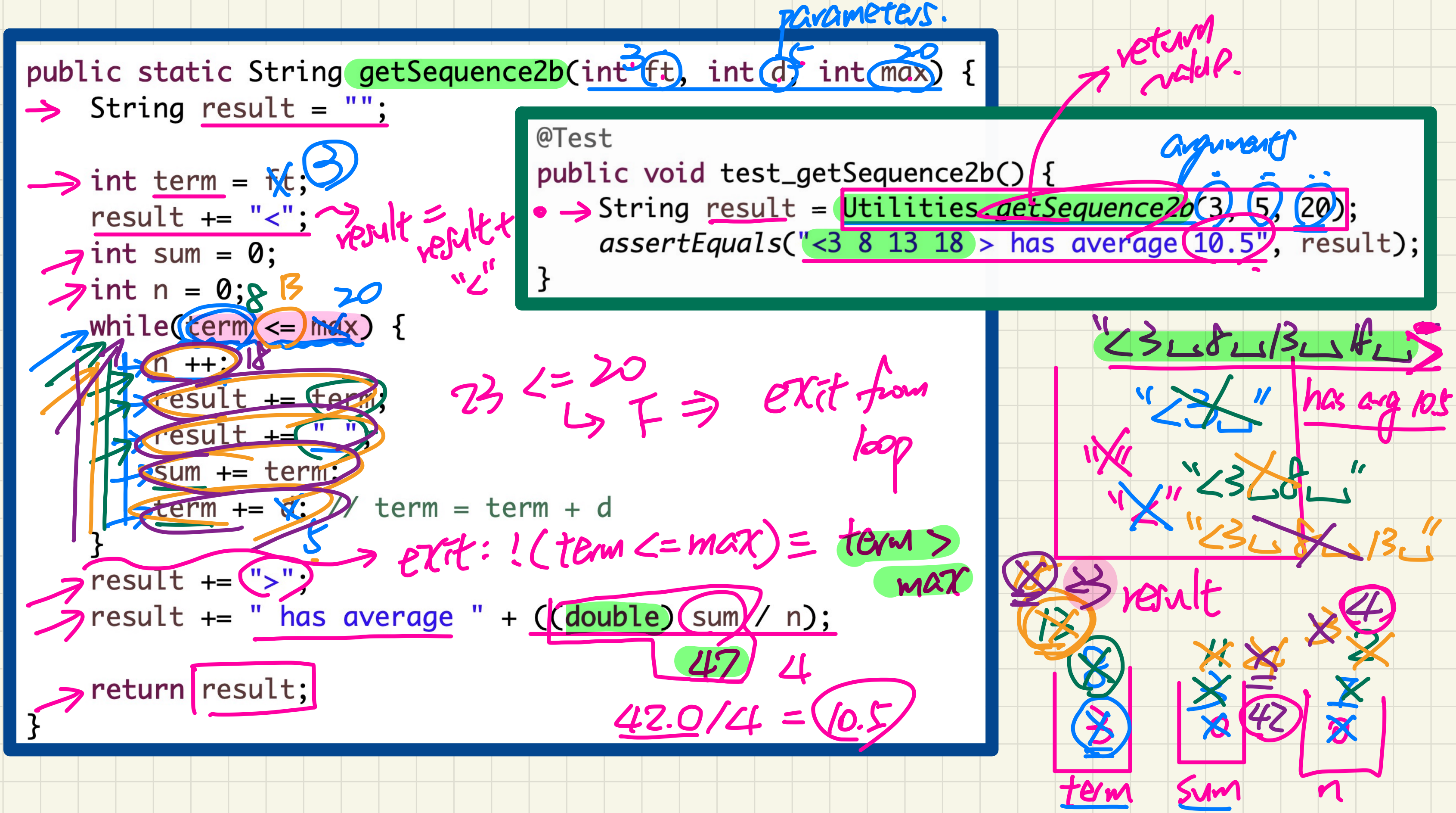
```java
public static String getSequence1a(int ft, int d, int n) {
    String result = "";

    int term = ft;
    result += "<";
    int sum = 0;
    for(int i = 1; i <= n; i ++) {
        result += term;
        if(i < n) { // not the last term
            result += ", ";
        }
        sum += term;
        term += d; // term = term + d
    }
    result += ">";
    result += " has average " + ((double) sum / n);

    return result;
}
```

n-1

i==n ⇒ nth term

```java
@Test
public void test_getSequence1a() {
    String result = Utilities.getSequence1a(6, 11, 4);
    assertEquals("<6, 17, 28, 39> has average 22.5", result);
}
```

Tracing: Arithmetic Sequence with Indefinite Length



```
public static String getSequence2b(int ft, int d, int max) {
  String result = "";

  int term = ft;
  result += "<";
  int sum = 0;
  int n = 0;
  while(term <= max) {
    n ++;
    result += term;
    result += " ";
    sum += term;
    term += d; // term = term + d
  }
  result += ">";
  result += " has average " + ((double) sum / n);

  return result;
}
```

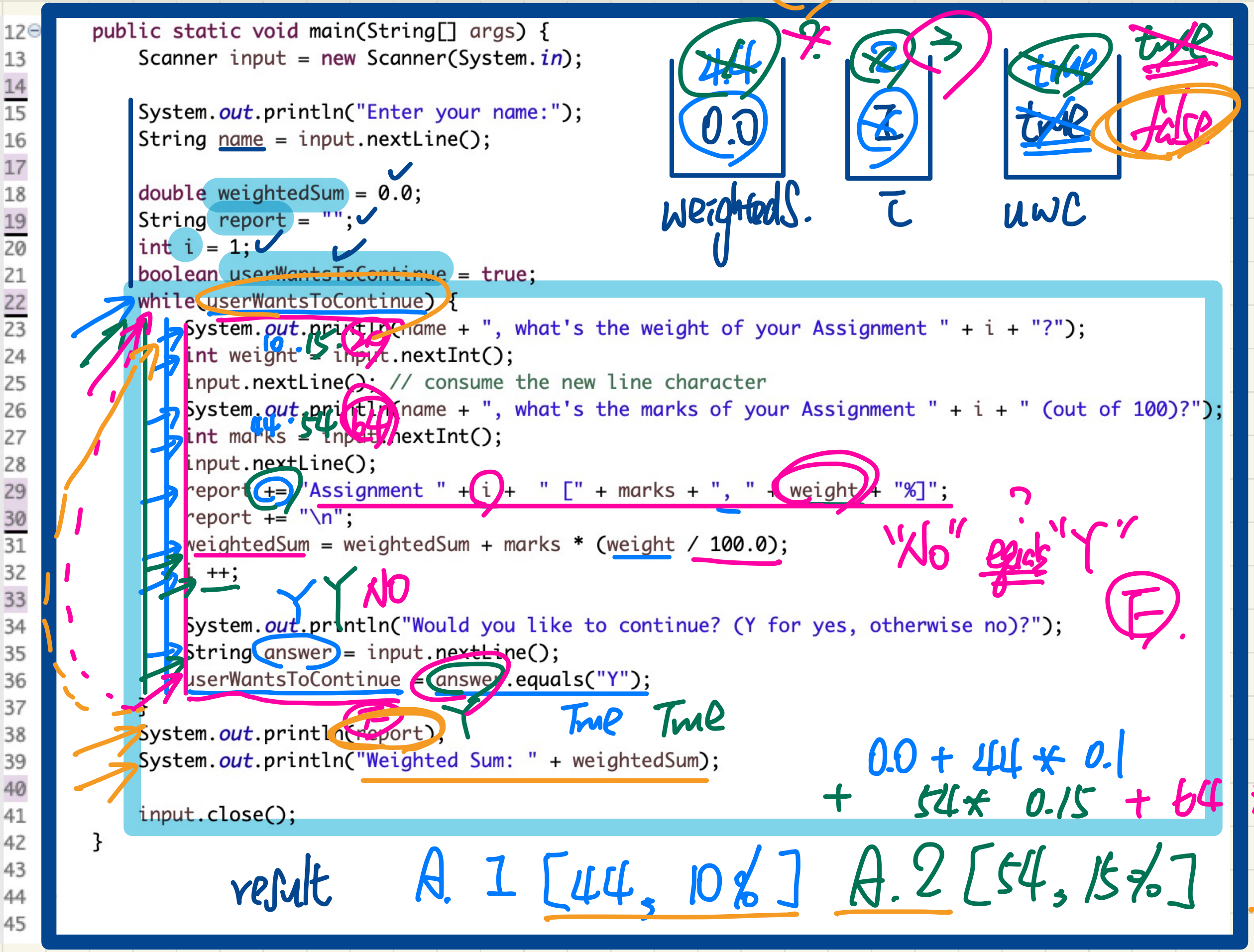
```
@Test
public void test_getSequence2b() {
  String result = Utilities.getSequence2b(3, 5, 20);
  assertEquals("<3 8 13 18 > has average 10.5", result);
}
```

EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 5

Introducing Arrays:
Syntax, Applications, and Tracing

Tracing: Calculation of Indefinite Number of Assignments



```java
public static void main(String[] args) {
    Scanner input = new Scanner(System.in);

    System.out.println("Enter your name:");
    String name = input.nextLine();

    double weightedSum = 0.0;
    String report = "";
    int i = 1;
    boolean userWantsToContinue = true;
    while(userWantsToContinue) {
        System.out.println(name + ", what's the weight of your Assignment " + i + "?");
        int weight = input.nextInt();
        input.nextLine(); // consume the new line character
        System.out.println(name + ", what's the marks of your Assignment " + i + " (out of 100)?");
        int marks = input.nextInt();
        input.nextLine();
        report += "Assignment " + i + " [" + marks + ", " + weight + "%]";
        report += "\n";
        weightedSum = weightedSum + marks * (weight / 100.0);
        i ++;

        System.out.println("Would you like to continue? (Y for yes, otherwise no)?");
        String answer = input.nextLine();
        userWantsToContinue = answer.equals("Y");
    }
    System.out.println(report);
    System.out.println("Weighted Sum: " + weightedSum);

    input.close();
}
```

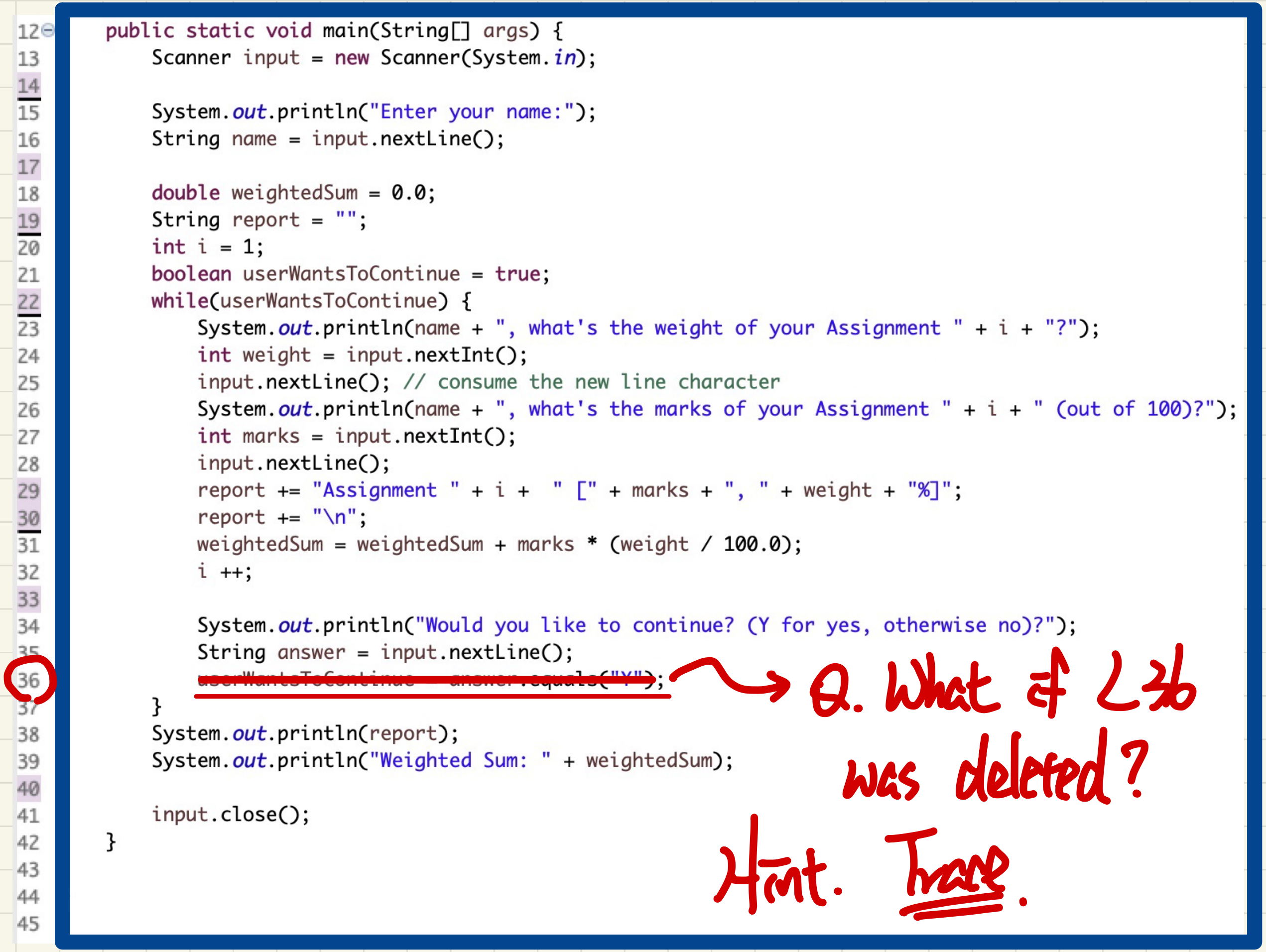
0.0 + 44 * 0.1
+ 54 * 0.15 + 64 * 0.2

result A. 1 [44, 10%] A. 2 [54, 15%] A. 3 [64, 20%]

Test Inputs:

weight = 10
marks = 44
Y
weight = 15
marks = 54
Y
weight = 20
marks = 64
No

Tracing: Calculation of Indefinite Number of Assignments



```
12     public static void main(String[] args) {
13         Scanner input = new Scanner(System.in);
14 
15         System.out.println("Enter your name:");
16         String name = input.nextLine();
17 
18         double weightedSum = 0.0;
19         String report = "";
20         int i = 1;
21         boolean userWantsToContinue = true;
22         while(userWantsToContinue) {
23             System.out.println(name + ", what's the weight of your Assignment " + i + "?");
24             int weight = input.nextInt();
25             input.nextLine(); // consume the new line character
26             System.out.println(name + ", what's the marks of your Assignment " + i + " (out of 100)?");
27             int marks = input.nextInt();
28             input.nextLine();
29             report += "Assignment " + i +  " [" + marks + ", " + weight + "%]";
30             report += "\n";
31             weightedSum = weightedSum + marks * (weight / 100.0);
32             i ++;
33 
34             System.out.println("Would you like to continue? (Y for yes, otherwise no)?");
35             String answer = input.nextLine();
36             ~~userWantsToContinue = answer.equals("Y")~~;
37         }
38         System.out.println(report);
39         System.out.println("Weighted Sum: " + weightedSum);
40 
41         input.close();
42     }
43 
44 
45 
```

Q. What if L36 was deleted?

Hint. Trace.

Test Inputs:

weight = 10

marks = 44

Y

weight = 15

marks = 54

Y

weight = 20

marks = 64

No

Indexing of an Array: Initializer

```
int[] ns = {10, 20, 30};
ns[0] = 23;
ns[1] = 46;
ns[2] = 69;
```

32 bits
|||
4 bytes

array

int[] ns =

data type.

type of each element.

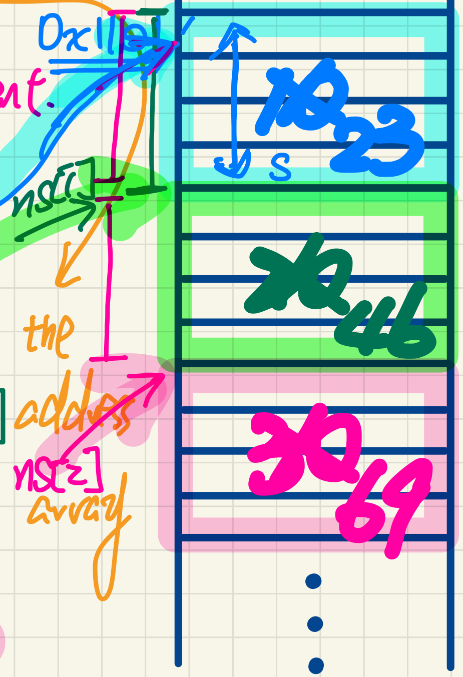
ns is an array of integers

0 1 2

ns → |10|20|30|

|0x1101|
ns

Computer Memory



ns[0] → base + s * 0

stores the base. starting address of the array

ns[1] → base + s * 1

ns[2] → base + s * 2

Address Calculation

- base address
- offset unit

Indexing of an Array: New Object

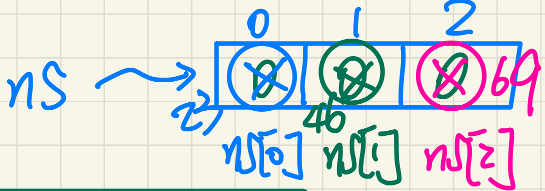
ns == ns[0] X

```
int[] ns = new int[3];
ns[0] = 23;
ns[1] = 46;
ns[2] = 69;
```

ns.length ③

allocate space in memory

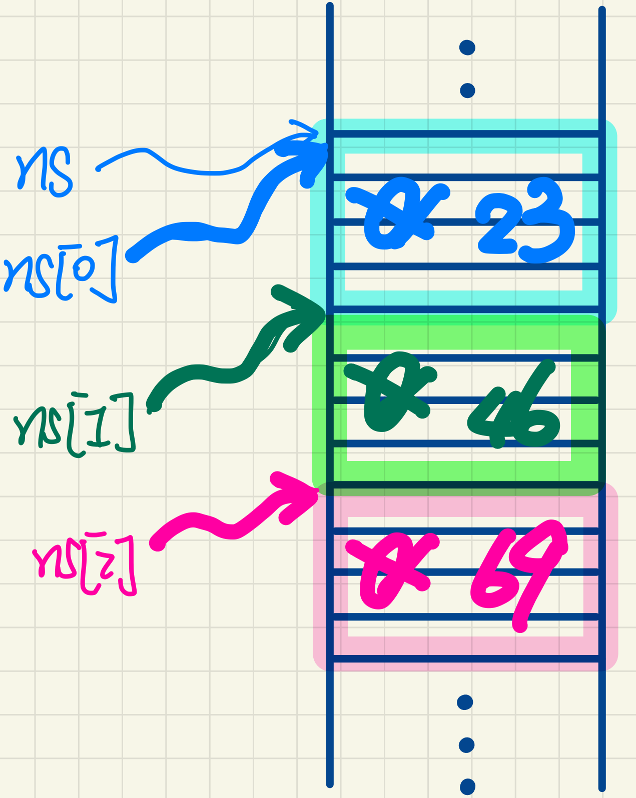
step.



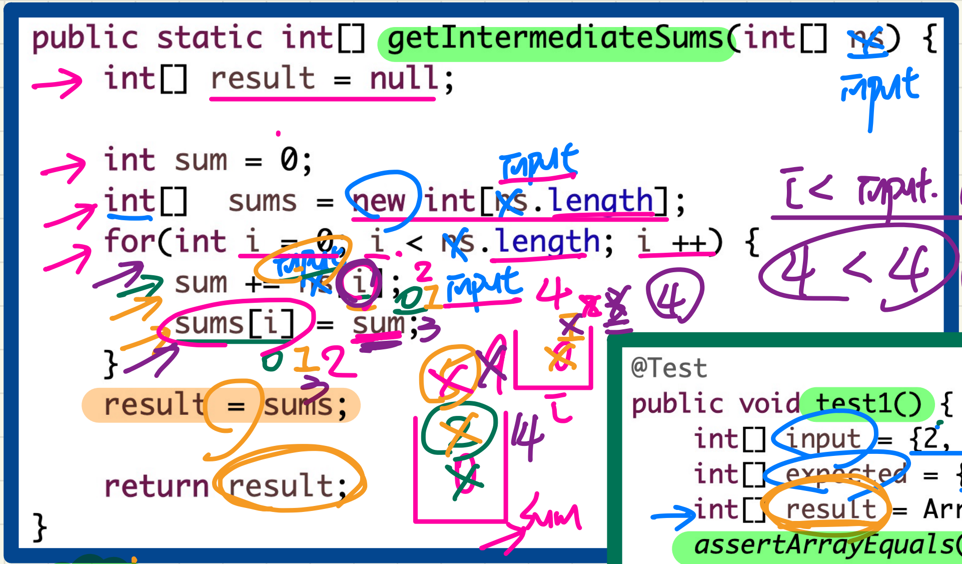
Address Calculation

- base address
- offset unit

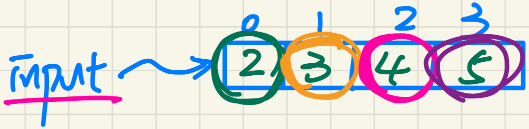
Computer Memory

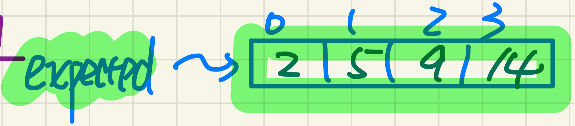


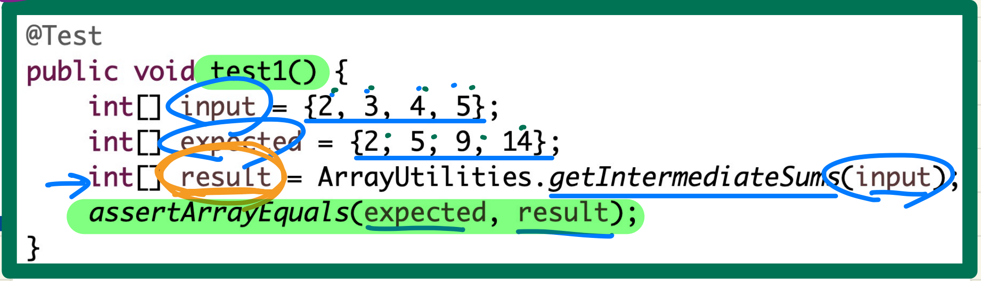
Implementing Utility Method using Arrays



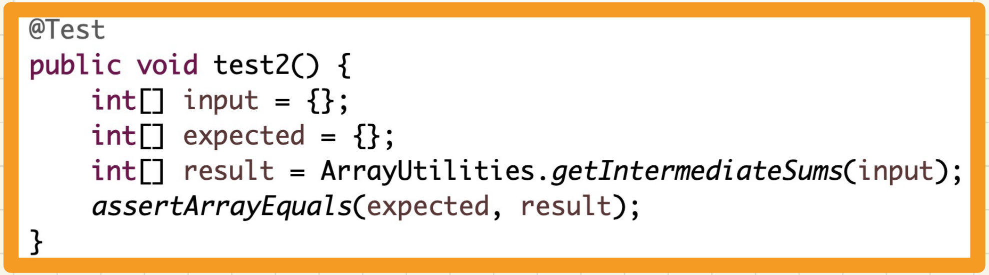
```
public static int[] getIntermediateSums(int[] ns) {
    int[] result = null;

    int sum = 0;
    int[] sums = new int[ns.length];
    for(int i = 0; i < ns.length; i ++) {
        sum += ns[i];
        sums[i] = sum;
    }
    result = sums;

    return result;
}
```



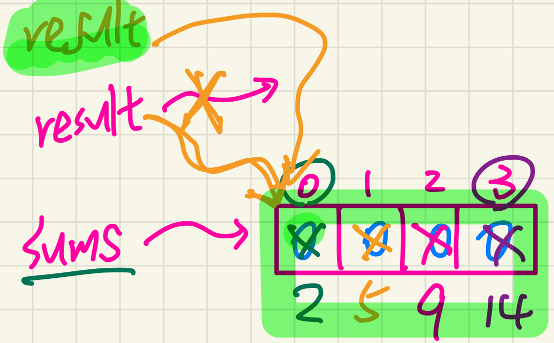




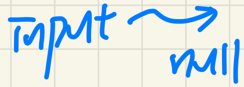
```
@Test
public void test1() {
    int[] input = {2, 3, 4, 5};
    int[] expected = {2, 5, 9, 14};
    int[] result = ArrayUtilities.getIntermediateSums(input);
    assertArrayEquals(expected, result);
}
```

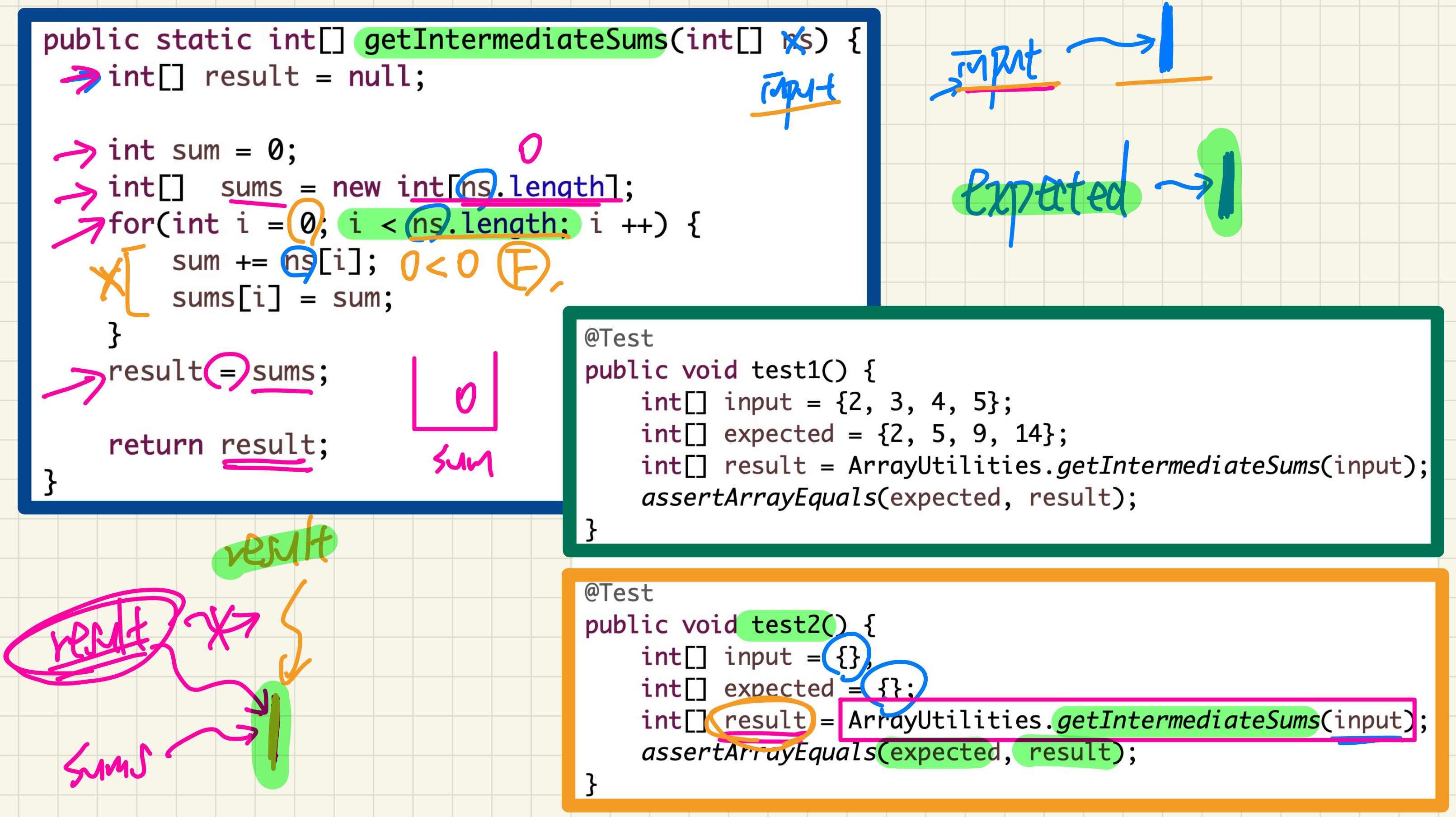


```
@Test
public void test2() {
    int[] input = {};
    int[] expected = {};
    int[] result = ArrayUtilities.getIntermediateSums(input);
    assertArrayEquals(expected, result);
}
```



Implementing Utility Method using Arrays





```
public static int[] getIntermediateSums(int[] ns) {
    int[] result = null;

    int sum = 0;
    int[]  sums = new int[ns.length];
    for(int i = 0; i < ns.length; i ++) {
        sum += ns[i];
        sums[i] = sum;
    }
    result = sums;

    return result;
}
```

```
@Test
public void test1() {
    int[] input = {2, 3, 4, 5};
    int[] expected = {2, 5, 9, 14};
    int[] result = ArrayUtilities.getIntermediateSums(input);
    assertArrayEquals(expected, result);
}
```

```
@Test
public void test2() {
    int[] input = {};
    int[] expected = {};
    int[] result = ArrayUtilities.getIntermediateSums(input);
    assertArrayEquals(expected, result);
}
```

EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 6

Classes, Attributes, Constructors, Accessors, Mutators, Method Invocations

Member suyeon ;

Visualization: Creating Objects

this . id "of" context object

suyeon == yuna false

```
public class Member {
    private int id;
    private char type;
    private double balance;
    private String name;
    private double weight;
    private double height;

    public Member() {
        // do nothing
    }

    public Member(int id, char type, double balance) {
        this.id = id;
        this.type = type;
        this.balance = balance;
    }
}
```

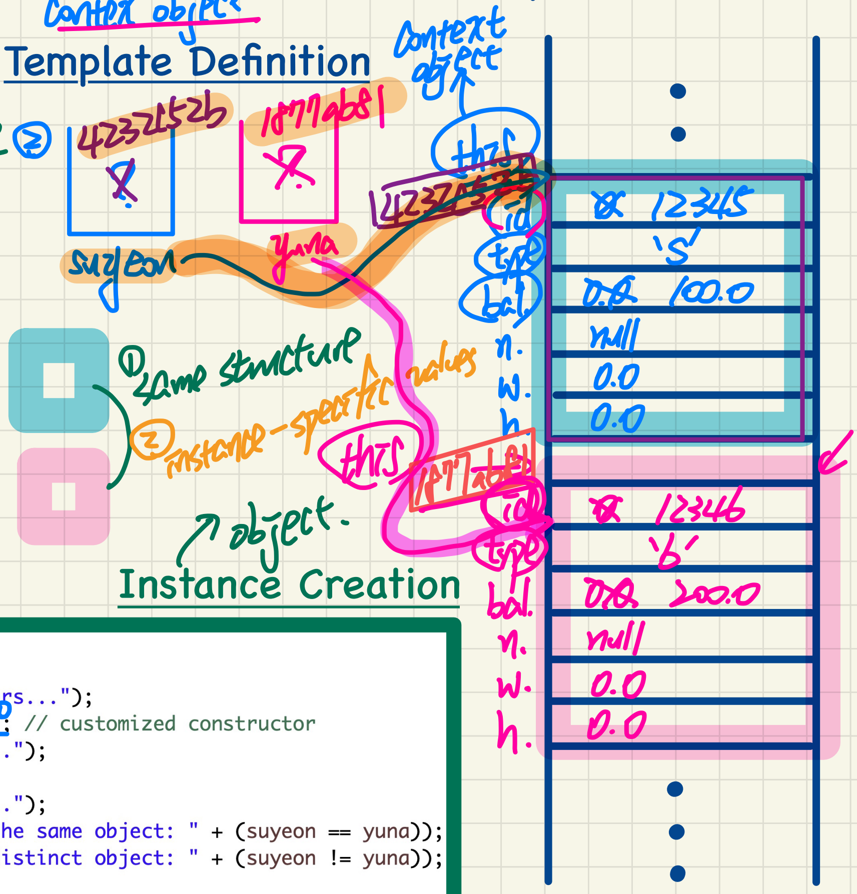
attributes defining the structure shared by all Member objects. ②

12346 12345 'b' 's' 200.0 100.0

12345 12346 's' 'b' 100.0 200.0

```
public class MemberApp2 {
    public static void main(String[] args) {
        System.out.println("Before creating two members...");
        Member suyeon = new Member(12345, 's', 100.0); // customized constructor
        System.out.println("After creating member 1...");
        Member yuna = new Member(12346, 'b', 200.0);
        System.out.println("After creating member 2...");
        System.out.println("Member1 and Member2 are the same object: " + (suyeon == yuna));
        System.out.println("Member1 and Member2 are distinct object: " + (suyeon != yuna));
    }
}
```

② ③ address of object type Member

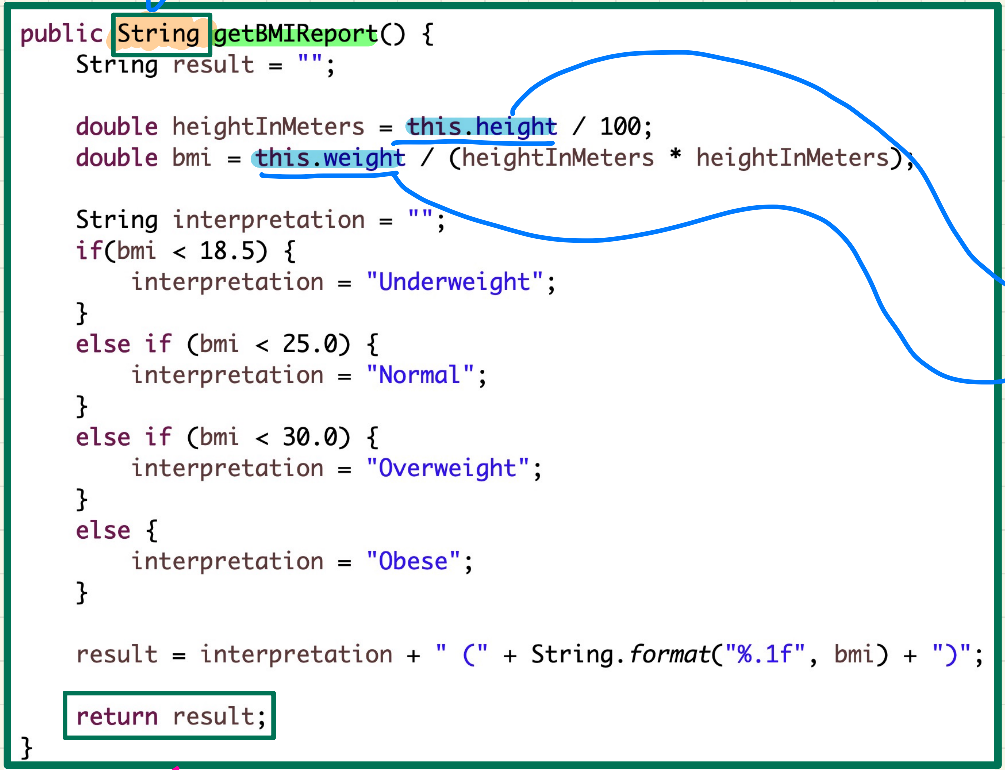


Template Definition

Instance Creation

Computer Memory

Methods: Accessor vs. Mutators

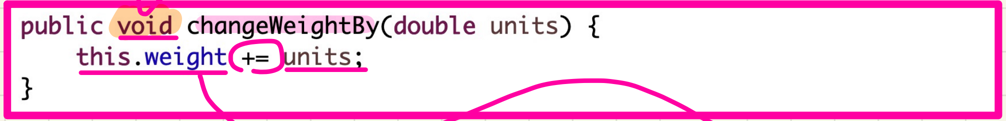


```java
public String getBMIReport() {
    String result = "";

    double heightInMeters = this.height / 100;
    double bmi = this.weight / (heightInMeters * heightInMeters);

    String interpretation = "";
    if(bmi < 18.5) {
        interpretation = "Underweight";
    }
    else if (bmi < 25.0) {
        interpretation = "Normal";
    }
    else if (bmi < 30.0) {
        interpretation = "Overweight";
    }
    else {
        interpretation = "Obese";
    }

    result = interpretation + " (" + String.format("%.1f", bmi) + ")";

    return result;
}
```

Accessor

↳ Computation on attributes without modifications.

↳ typically, no modification should be made to attributes.

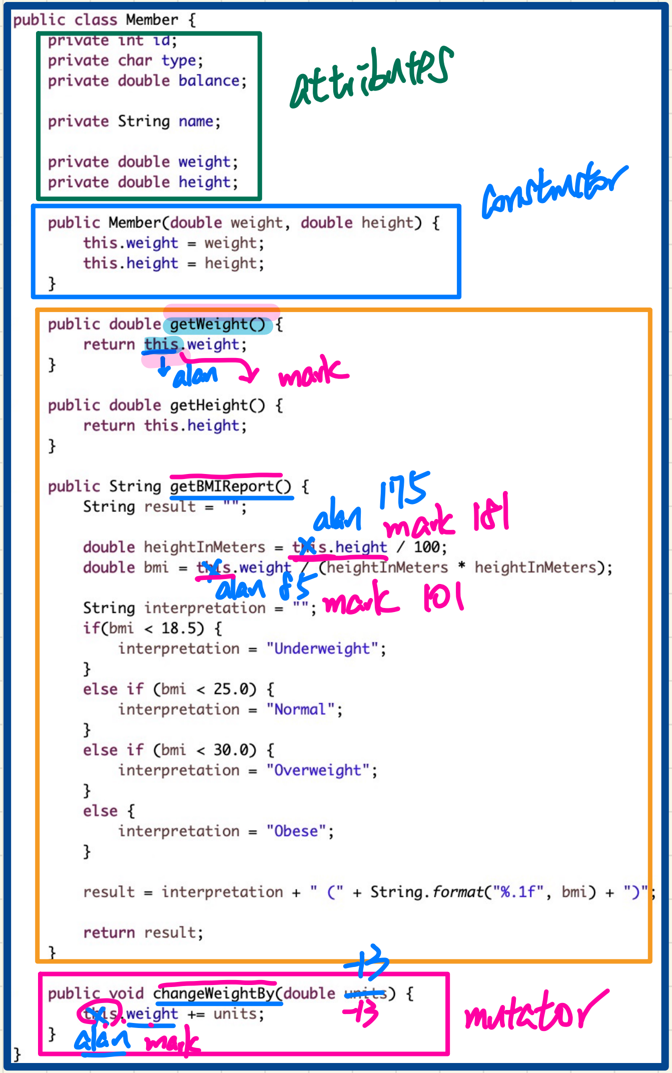


```java
public void changeWeightBy(double units) {
    this.weight += units;
}
```

Mutator

↳ modify values of attributes.

Visualization: Calling Methods



```java
public class Member {
    private int id;
    private char type;
    private double balance;

    private String name;

    private double weight;
    private double height;

    public Member(double weight, double height) {
        this.weight = weight;
        this.height = height;
    }

    public double getWeight() {
        return this.weight;
    }

    public double getHeight() {
        return this.height;
    }

    public String getBMIReport() {
        String result = "";

        double heightInMeters = this.height / 100;
        double bmi = this.weight / (heightInMeters * heightInMeters);

        String interpretation = "";
        if(bmi < 18.5) {
            interpretation = "Underweight";
        }
        else if (bmi < 25.0) {
            interpretation = "Normal";
        }
        else if (bmi < 30.0) {
            interpretation = "Overweight";
        }
        else {
            interpretation = "Obese";
        }

        result = interpretation + " (" + String.format("%.1f", bmi) + ")";

        return result;
    }

    public void changeWeightBy(double units) {
        this.weight += units;
    }
}
```

attributes

constructor

accessors

alan → mark

alan 175 mark 181

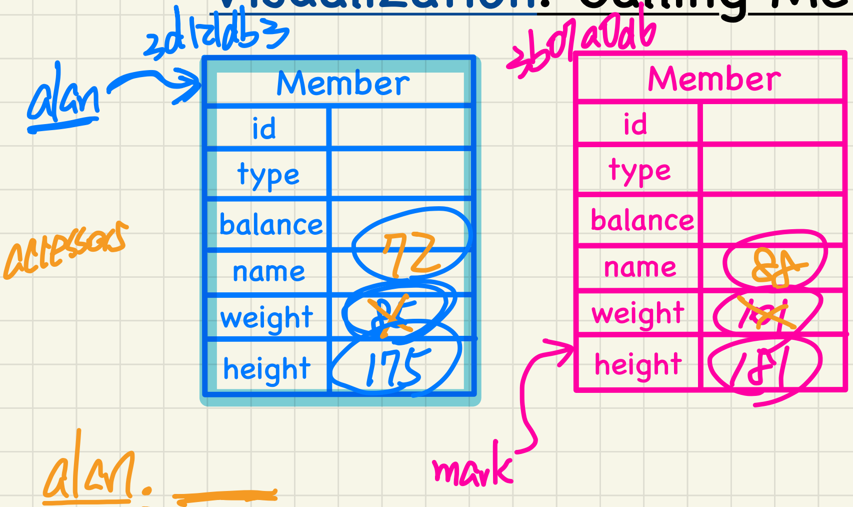
alan 85 mark 101

mutator

-13

-13

alan mark



3d12ddb3

alan →

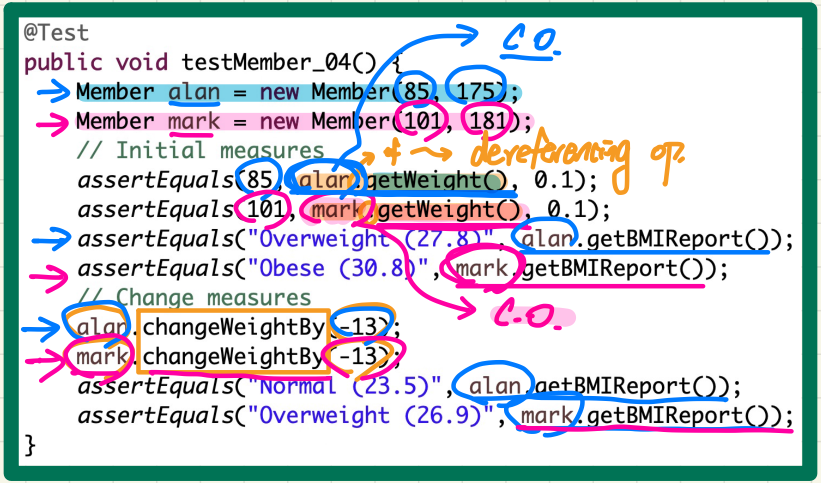
Member	
id	
type	
balance	
name	72
weight	~~85~~
height	175

3b07a0db

mark →

Member	
id	
type	
balance	
name	88
weight	~~101~~
height	181

alan.



```java
@Test
public void testMember_04() {
  Member alan = new Member(85, 175);
  Member mark = new Member(101, 181);
  // Initial measures
  assertEquals(85, alan.getWeight(), 0.1);
  assertEquals(101, mark.getWeight(), 0.1);
  assertEquals("Overweight (27.8)", alan.getBMIReport());
  assertEquals("Obese (30.8)", mark.getBMIReport());
  // Change measures
  alan.changeWeightBy(-13);
  mark.changeWeightBy(-13);
  assertEquals("Normal (23.5)", alan.getBMIReport());
  assertEquals("Overweight (26.9)", mark.getBMIReport());
}
```

c.o.

dereferencing op.

c.o.

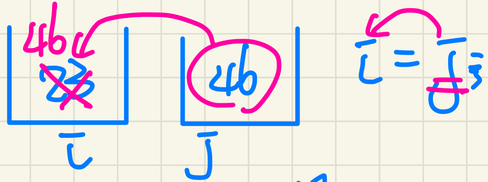
template

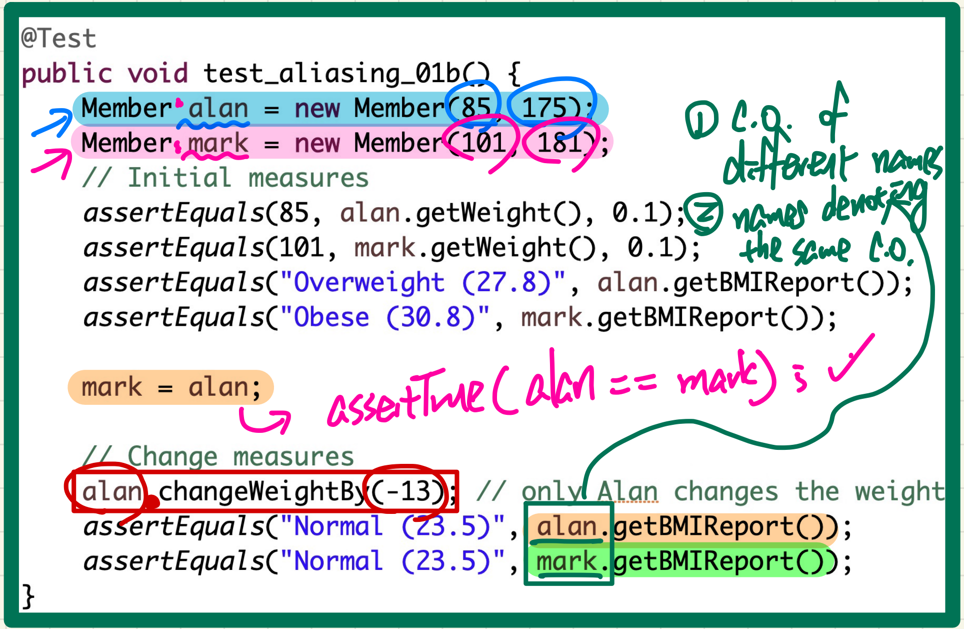
EEECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 7

Aliasing,
Reference-Typed, Single-Valued Attributes

Reference Aliasing: Copying Address



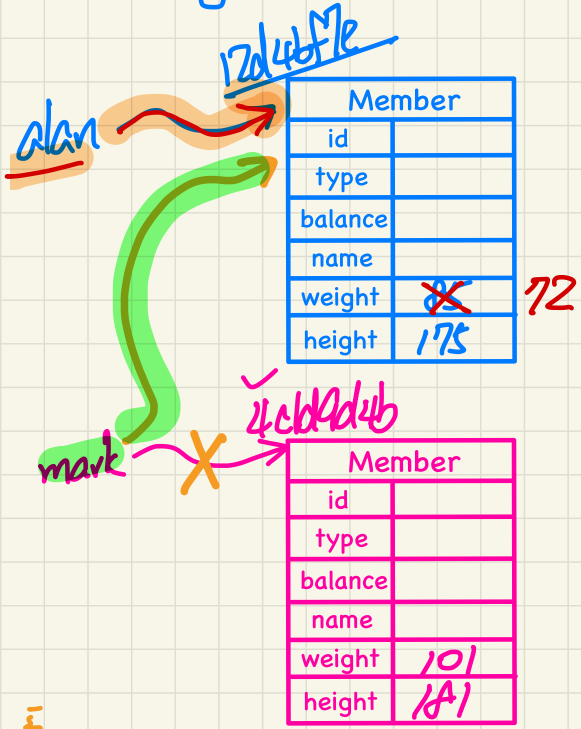


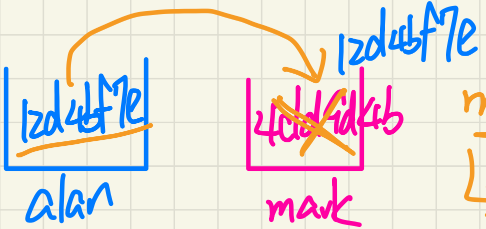
```
@Test
public void test_aliasing_01b() {
    Member alan = new Member(85, 175);
    Member mark = new Member(101, 181);
    // Initial measures
    assertEquals(85, alan.getWeight(), 0.1);
    assertEquals(101, mark.getWeight(), 0.1);
    assertEquals("Overweight (27.8)", alan.getBMIReport());
    assertEquals("Obese (30.8)", mark.getBMIReport());

    mark = alan;

    // Change measures
    alan.changeWeightBy(-13); // only Alan changes the weight
    assertEquals("Normal (23.5)", alan.getBMIReport());
    assertEquals("Normal (23.5)", mark.getBMIReport());
}
```

① C.O. of different names

② names denoting the same C.O.

assertTrue(alan == mark); ✓

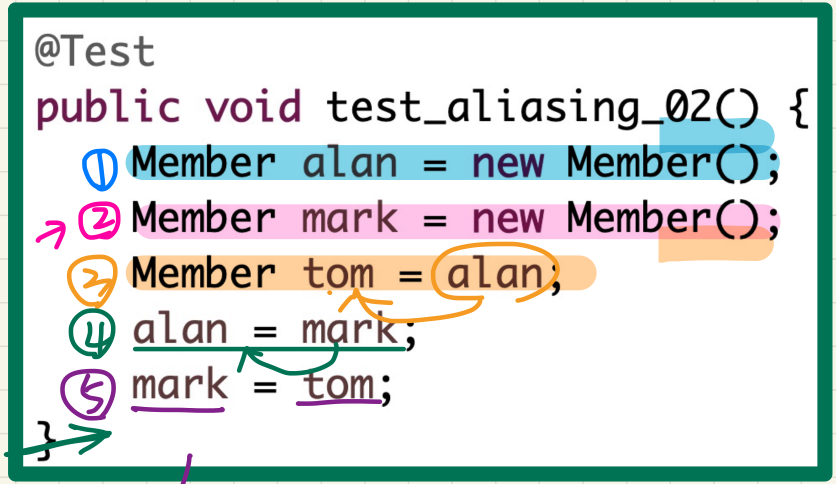




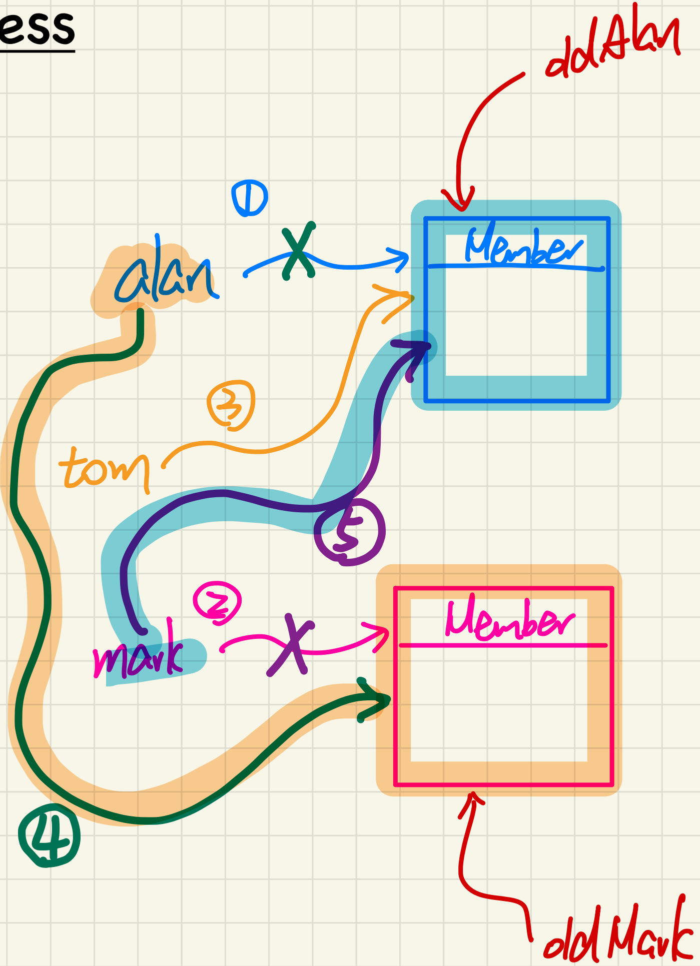
mark = alan;

↳ copy addressed stored in alan over to mark.

Reference Aliasing: ~~Copying~~ Swapping Address



```
@Test
public void test_aliasing_02() {
 ① Member alan = new Member();
 ② Member mark = new Member();
 ③ Member tom = alan;
 ④ alan = mark;
 ⑤ mark = tom;
}
```



↳ alternatively:

alan = mark;
mark = alan;
] swap?

** tom.trainer = alan.getTrainer();

Reference-Typed, Single-Valued Attributes

* tom.trainer = mark.getTrainer()

```
public class Member {
    private int id;
    private char type;
    private String name;
    private double weight;
    private double height;

    Trainer trainer;

    public Member(String name) {
        this.name = name;
    }
    public Trainer getTrainer() {
        return this.trainer;
    }
    public void registerTrainer(Trainer trainer) {
        this.trainer = trainer;
    }
    public void referTrainer(Member m) {
      * this.trainer = m.getTrainer();
    }**
}
```

alan.trainer = jared;

mark.trainer = jon;

alan mark tom

jared

alan mark

jared jon

jon

mark alan

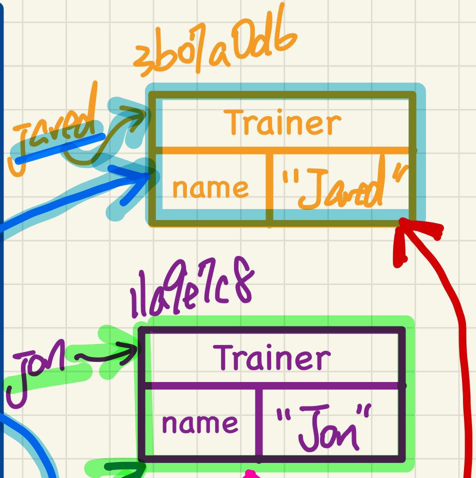
tom

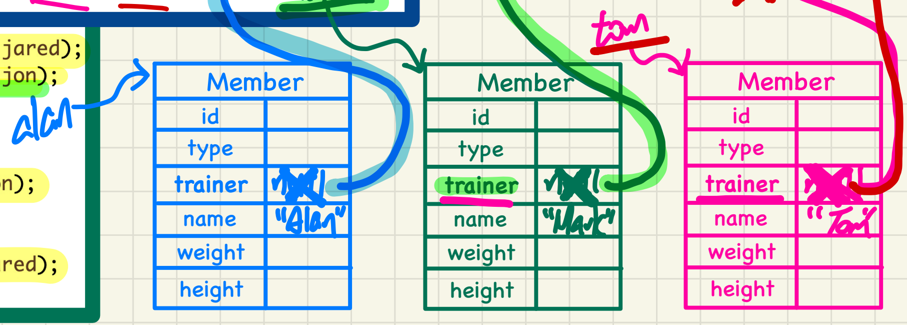
tom

mark alan

mark

```
@Test
public void test_MemberTrainer_01() {
    Member alan = new Member("Alan");
    Member mark = new Member("Mark");
    Member tom = new Member("Tom");
    assertTrue(alan.getTrainer() == null);
    assertNull(mark.getTrainer());
    assertFalse(tom.getTrainer() != null);

    Trainer jared = new Trainer("Jared");
    Trainer jon = new Trainer("Jon");

    alan.registerTrainer(jared);
    mark.registerTrainer(jon);
  ✓ assertTrue(alan.getTrainer() != null && alan.getTrainer() == jared);
  ✓ assertTrue(mark.getTrainer() != null && mark.getTrainer() == jon);
    assertNull(tom.getTrainer());

    tom.referTrainer(mark);
    assertTrue(tom.getTrainer() != null && tom.getTrainer() == jon);

    tom.referTrainer(alan);
    assertTrue(tom.getTrainer() != null && tom.getTrainer() == jared);
}
```



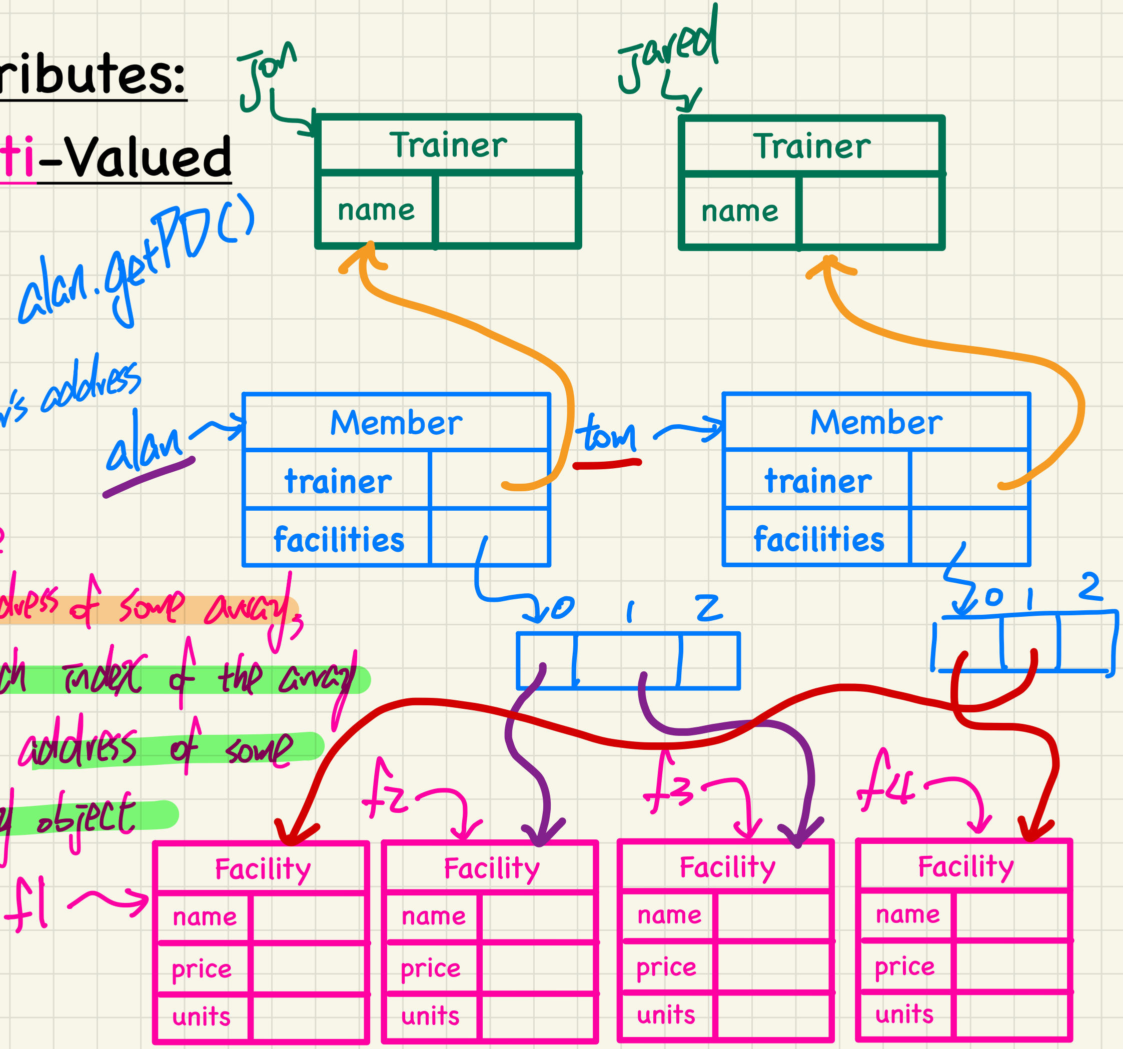


Reference-Typed Attributes: Single-Valued vs. Multi-Valued

```
public class Member {
  /* single-valued */
  private Trainer trainer;

  /* multi-valued */
  private Facility[] facilities;
}
```

↳ stores a single Trainer's address

↳ stores the starting address of some array, where each index of the array stores the address of some Facility object



EEC S1022 Programming for Mobile Computing (Winter 2021)

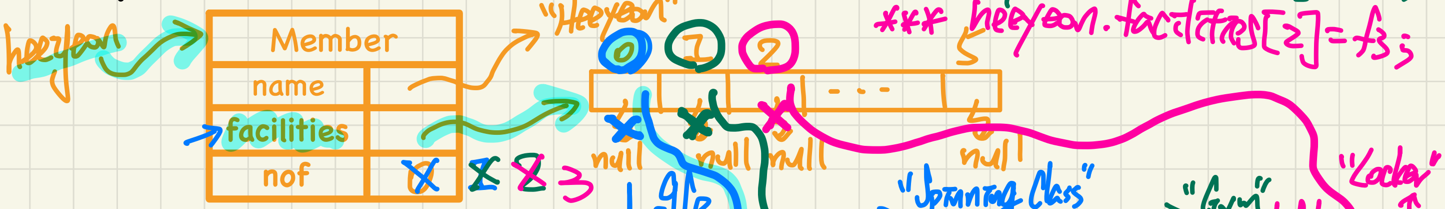
Java Tutorials - Week 8

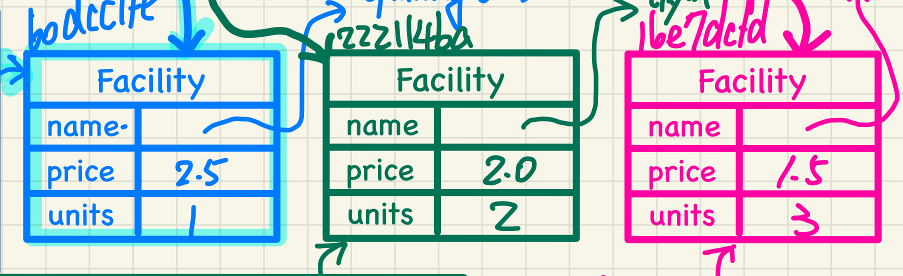
Aliasing,
Reference-Typed, Multi-Valued Attributes

String → ref. type

Reference-Typed, Multi-Valued Attributes

* heeyeon.facilities[0] = f1;
** heeyeon.facilities[1] = f2;
*** heeyeon.facilities[2] = f3;





```
public class Member {
    private String name;

    private final int MAX_NUMBER_OF_FACILITIES = 6;
    private Facility[] facilities;
    private int nof;

    public Member() {
        this.facilities = new Facility[MAX_NUMBER_OF_FACILITIES];
        this.nof = 0;
    }
      heeyeon            "Heeyeon"
    public Member(String name) {
        this();
        this.name = name;
    }
      heeyeon
    public int getNumberOfFacilites() {
        return this.nof;
    }
          heeyeon                      f1
    public void addFacility(Facility f) {
*       this.facilities[this.nof] = f;
**      this.nof ++;
***  }
}    heeyeon              heeyeon     f1
     heeyeon                          f2
     heeyeon                          f3
```

f2

```
@Test
public void test() {
    Facility f1 = new Facility("Spinning Class", 2.5, 1);
    Facility f2 = new Facility("Gym", 2.0, 2);
    Facility f3 = new Facility("Locker", 1.5, 3);

    Member heeyeon = new Member("Heeyeon");
    assertEquals(0, heeyeon.getNumberOfFacilites());
    heeyeon.addFacility(f1);
    assertEquals(1, heeyeon.getNumberOfFacilites());
    heeyeon.addFacility(f2);
    assertEquals(2, heeyeon.getNumberOfFacilites());
    heeyeon.addFacility(f3);
    assertEquals(3, heeyeon.getNumberOfFacilites());
}
```

f3

aliasing

heeyeon.facilities[0] == f1

Reference-Typed, Multi-Valued Attributes

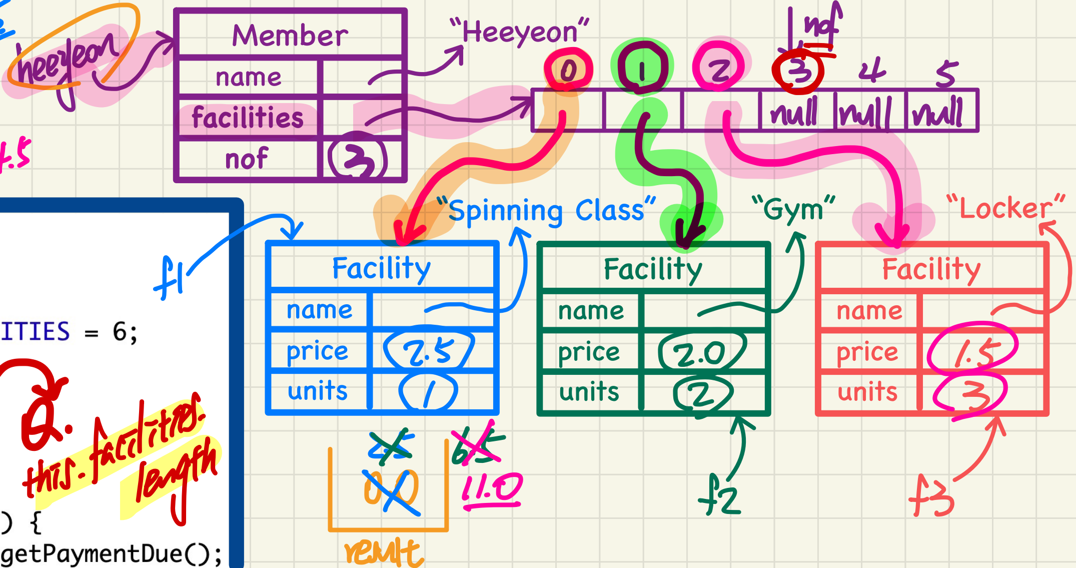
getPaymentDue

i==0 heeyeon.facilities[0].getPD() 2.5

c.o.

i==1 heeyeon.facilities[i].getPD() 4.0

i==2 heeyeon.facilities[2].getPD() 4.5



```
public class Member {
    private String name;

    private final int MAX_NUMBER_OF_FACILITIES = 6;
    private Facility[] facilities;
    private int nof;

    public double getPaymentDue() {
        double result = 0.0;
        for(int i = 0; i < this.nof; i ++) {
            result += this.facilities[i].getPaymentDue();
        }
        return result;
    }

    public void addFacility(Facility f) {
        this.facilities[this.nof] = f;
        this.nof ++;
    }
}
```

heeyeon

Q. this.facilities.length

0.
1
2

null

NullPointerException.

A. i==3 heeyeon.facilities[3].getPD

```
@Test
public void test_2() {
    Facility f1 = new Facility("Spinning Class", 2.5, 1);
    Facility f2 = new Facility("Gym", 2.0, 2);
    Facility f3 = new Facility("Locker", 1.5, 3);

    Member heeyeon = new Member("Heeyeon");
    heeyeon.addFacility(f1);
    heeyeon.addFacility(f2);
    heeyeon.addFacility(f3);

    assertEquals(2.5 * 1 + 2.0 * 2 + 1.5 * 3, heeyeon.getPaymentDue(), 0.01);
}
```

C.O.

* heeyeon.facilities[~~0~~ 1].getName().equals("Gym") F T

Reference-Typed, Multi-Valued Attributes

getFacilityUnits

extendFacilityUnits

```
public class Member {
    private String name;

    private final int MAX_NUMBER_OF_FACILITIES = 6;
    private Facility[] facilities;
    private int nof;

    public int getFacilityUnits(String name) {
        Facility f = this.getFacility(name);
        int units = -1;
        if(f != null) {
            units = f.getUnits();
        }
        return units;
    }
    public void extendFacilityUnits(String name, int howMany) {
        Facility f = this.getFacility(name);
        if(f != null) {
            f.setUnits(f.getUnits() + howMany);
        }
    }
    private Facility getFacility(String name) {
        Facility f = null;
        boolean hasFound = false;
        for(int i = 0; !hasFound && i < this.nof; i ++) {
            if(this.facilities[i].getName().equals(name)) {
                f = this.facilities[i];
                hasFound = true;
            }
        }
        return f;
    }
}
```

heeyeon

"Gym" heeyeon 2

"Gym" 2

"Gym" heeyeon

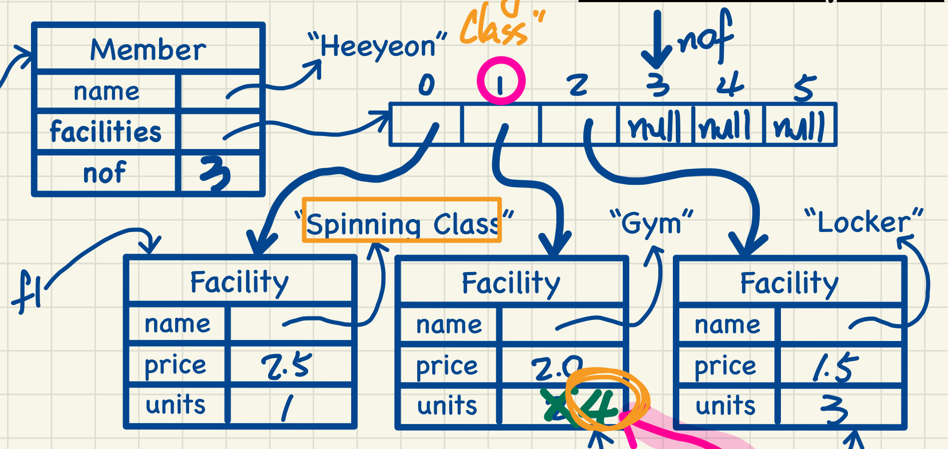
"Gym" F T F

* heeyeon 1 X

"Gym"

1

f = heeyeon.facilities[1]



```
@Test
public void test_3() {
    Facility f1 = new Facility("Spinning Class", 2.5, 1);
    Facility f2 = new Facility("Gym", 2.0, 2);
    Facility f3 = new Facility("Locker", 1.5, 3);

    Member heeyeon = new Member("Heeyeon");
    heeyeon.addFacility(f1);
    heeyeon.addFacility(f2);
    heeyeon.addFacility(f3);

    int units = heeyeon.getFacilityUnits("Gym");
    assertEquals(2, units);

    heeyeon.extendFacilityUnits("Gym", 2);
    assertEquals(4, heeyeon.getFacilityUnits("Gym"));
    assertEquals(2.5 * 1 + 2.0 * 4 + 1.5 * 3, heeyeon.getPaymentDue(), 0.01);
}
```

f2 f3

c.o. 2

4

f X → null

EEC S1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 9

Mobile App Development in Android Studio

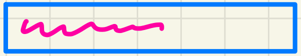
Model, View, Controller

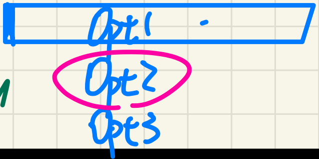
GUI Components

- Text -> TextView
- Text -> Plain Text
- Containers -> Spinner
- Buttons -> Button

displaying output to user] System.out.println.

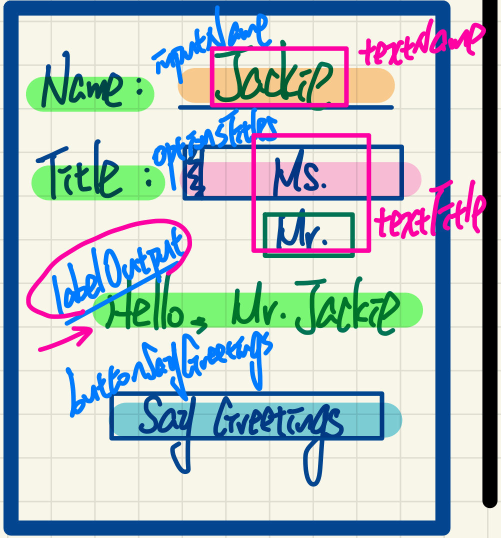
user inputs] → Scanner → nextLine.



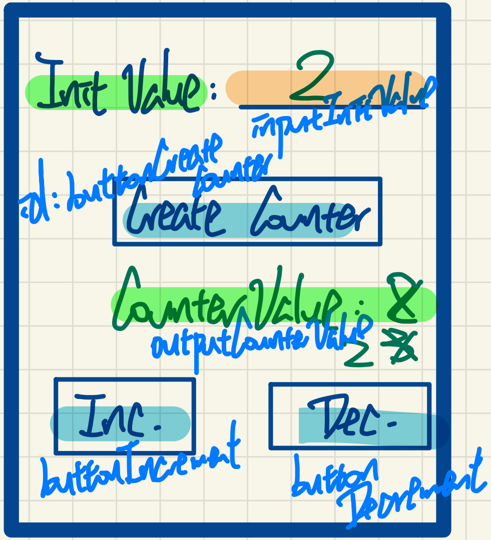


trigger for computation

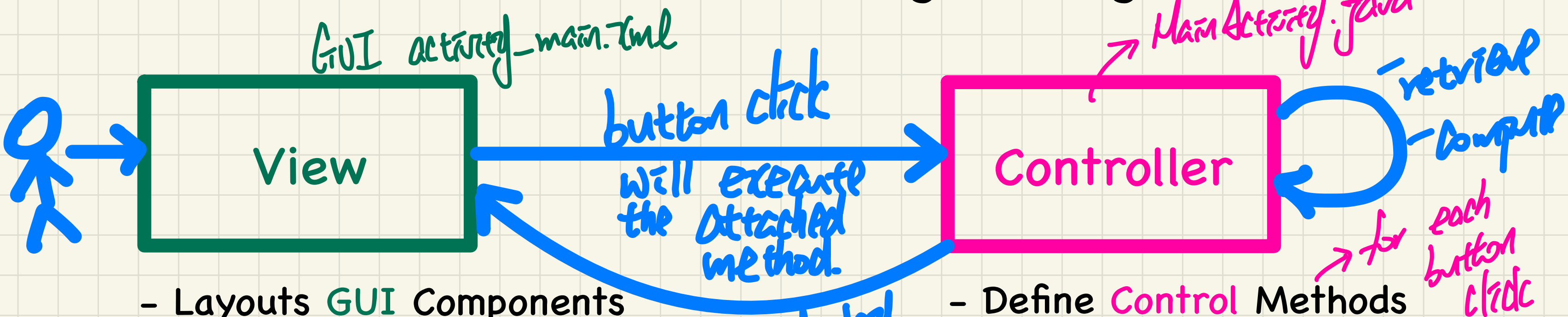
App: Greeting



App: Counter



View vs. Controller: Event-Driven Programming



- Layouts GUI Components
- Declare id's of GUI Components
- Attach/Register Control Methods

↳ for each button click (onClick).

- Define Control Methods
- Retrieve Inputs from GUI
- Display Outputs on GUI

helper methods

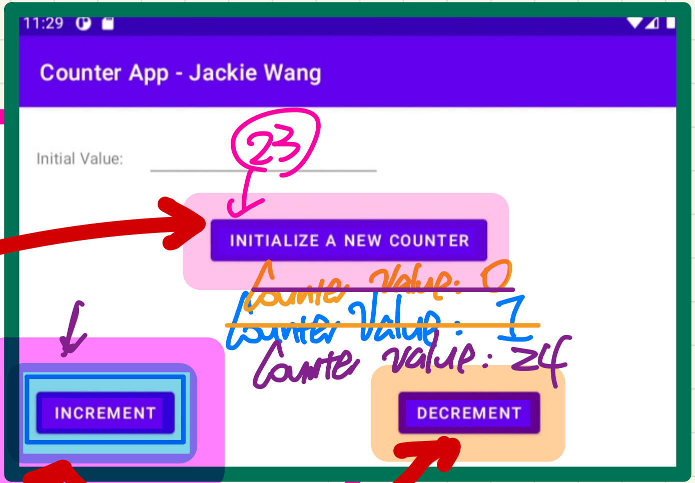
Each user interaction with a GUI component generates an event whose occurrence executes the attached/registered control method

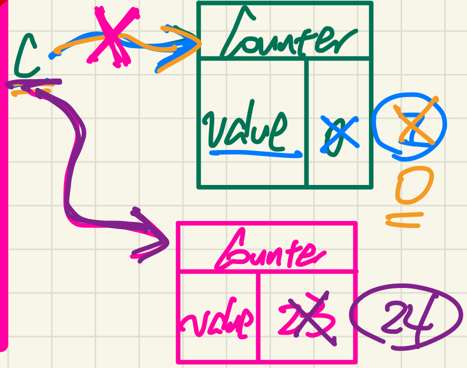
Model-View-Controller: Tracing

Controller

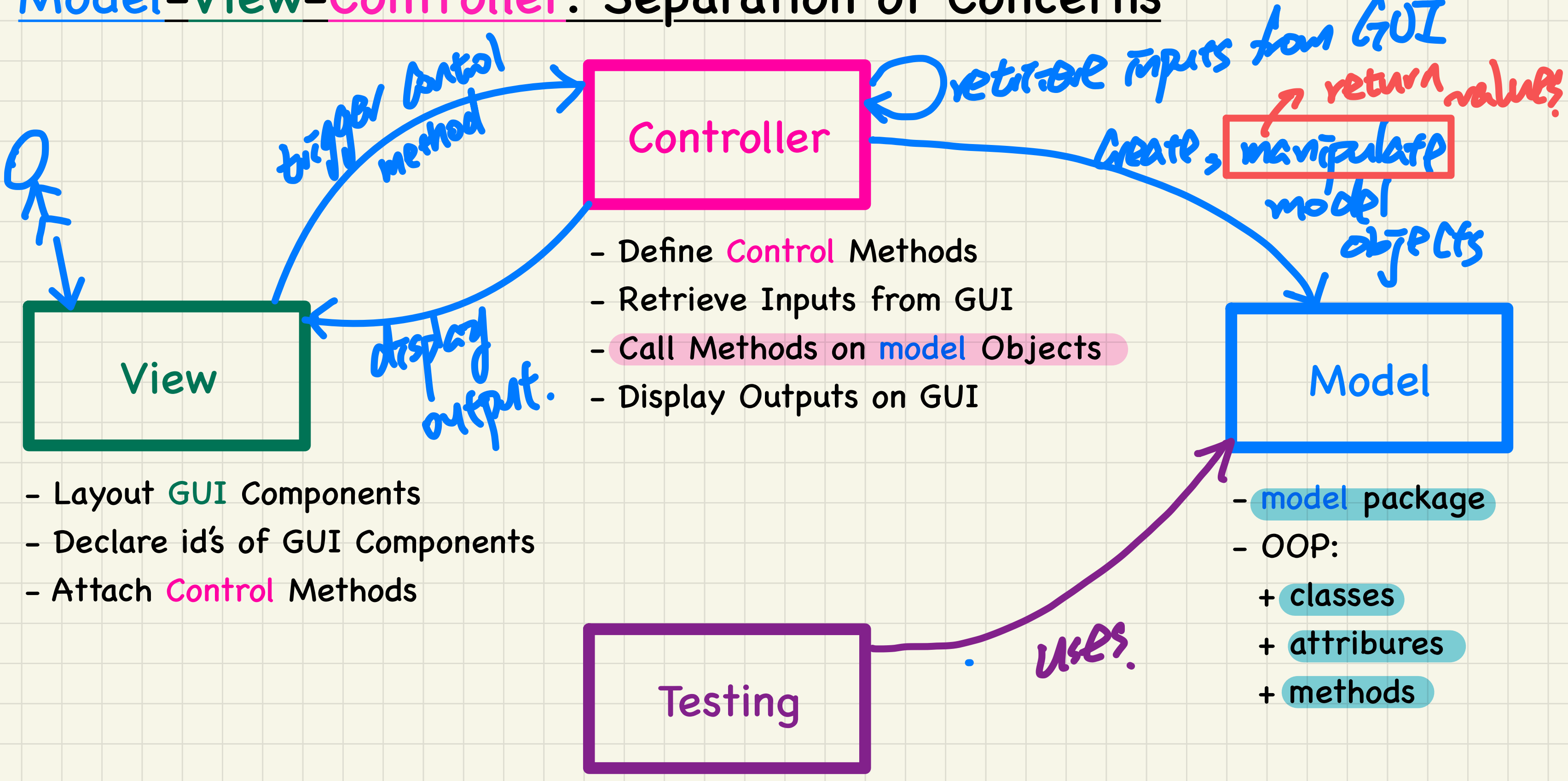
```
Counter c;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    c = new Counter( value: 0)
}

public void computeButtonCreateCounterClicked(View view) {
    String textInitValue = getInputOfTextField(R.id.inputInitValue);
    int initValue = Integer.parseInt(textInitValue);
    c = new Counter(initValue);
    setContentsOfTextView(R.id.outputCounterValue, newContents: "Counter Value: " + c.getValue());
}

public void computeButtonIncrementClicked(View view) {
    c.increment();
    setContentsOfTextView(R.id.outputCounterValue, newContents: "Counter Value: " + c.getValue());
}

public void computeButtonDecrementClicked(View view) {
    c.decrement();
    setContentsOfTextView(R.id.outputCounterValue, newContents: "Counter Value: " + c.getValue());
}
```

View





Model-View-Controller: Separation of Concerns



Controller

- Define Control Methods
- Retrieve Inputs from GUI
- Call Methods on model Objects
- Display Outputs on GUI

View

- Layout GUI Components
- Declare id's of GUI Components
- Attach Control Methods

Model

- model package
- OOP:
  - + classes
  - + attribures
  - + methods

Testing

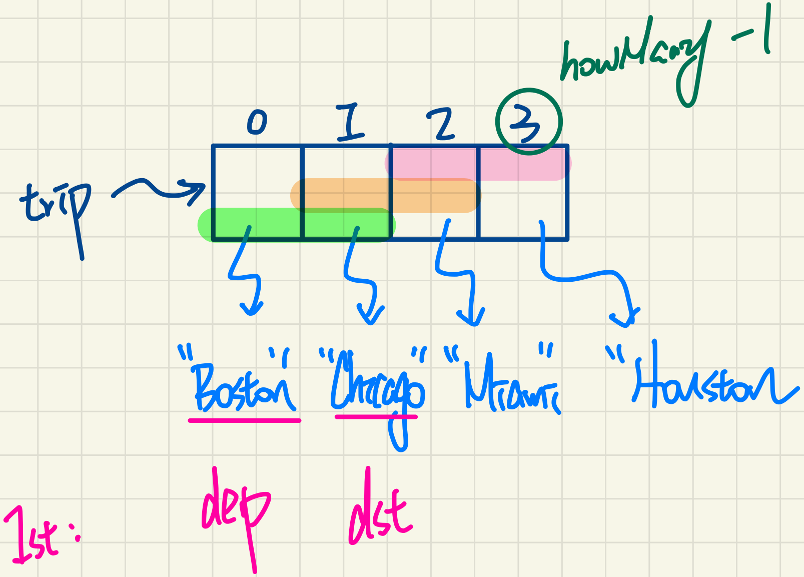
- Assertions on model Objects

EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 10

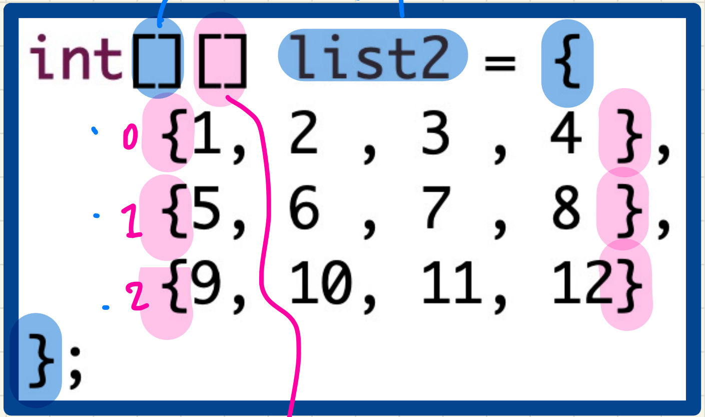
Two-Dimensional Arrays
Part I: Console App

Calculating Distances between Cities



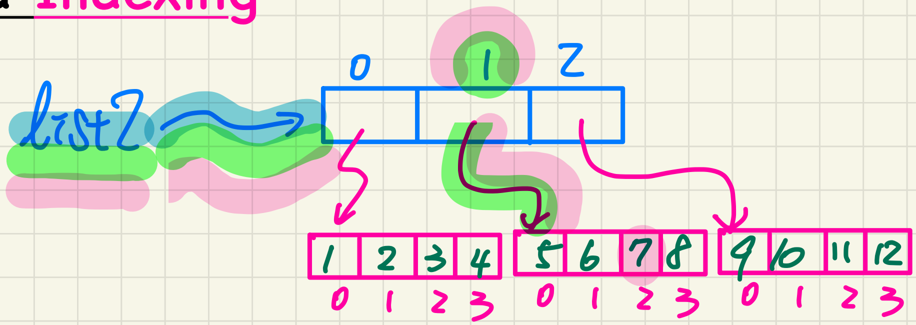
2D Array: Initialization and Indexing

1st dimension (top).



```
int[][] list2 = {
    {1, 2 , 3 , 4 },
    {5, 6 , 7 , 8 },
    {9, 10, 11, 12}
};
```

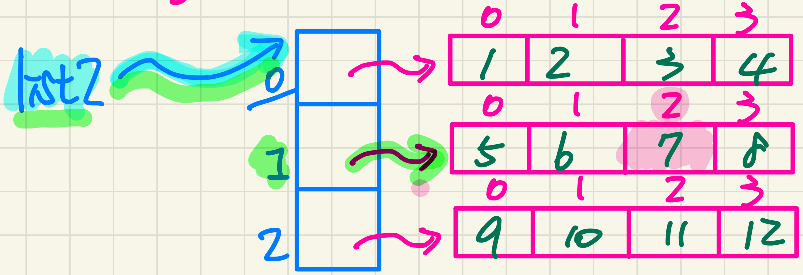
2nd dimension



list2

list2[1] (2nd row)

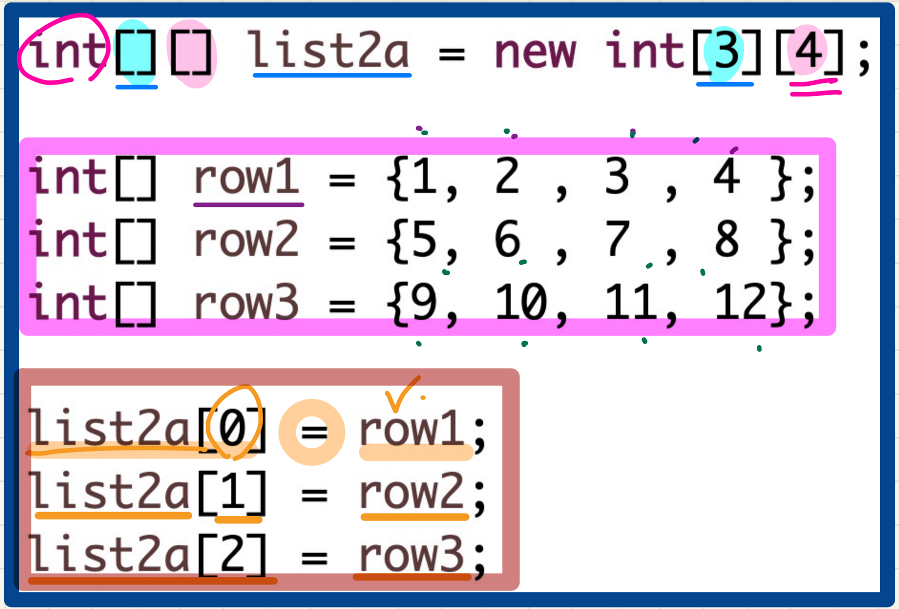
list2[1][2] (intersection of 2nd row & 3rd column)



row (index into 1st dimen.)

column (index into 2nd dimen.).

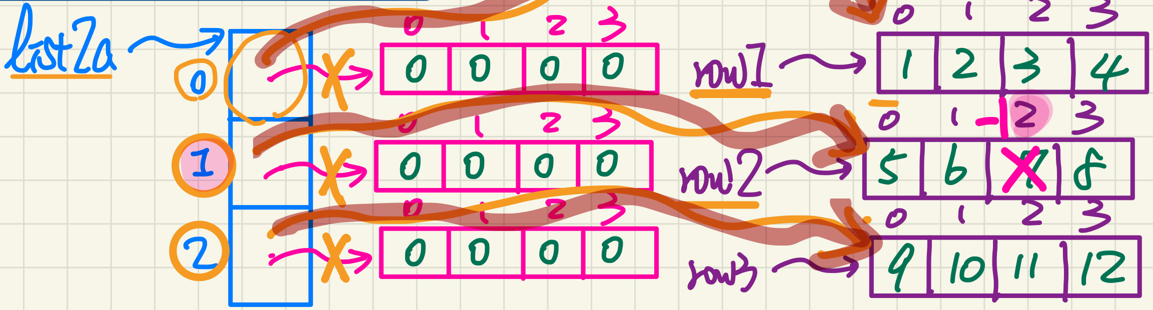
Creating a new 2D Array Object



```
int[][] list2a = new int[3][4];

int[] row1 = {1, 2 , 3 , 4 };
int[] row2 = {5, 6 , 7 , 8 };
int[] row3 = {9, 10, 11, 12};

list2a[0] = row1;
list2a[1] = row2;
list2a[2] = row3;
```

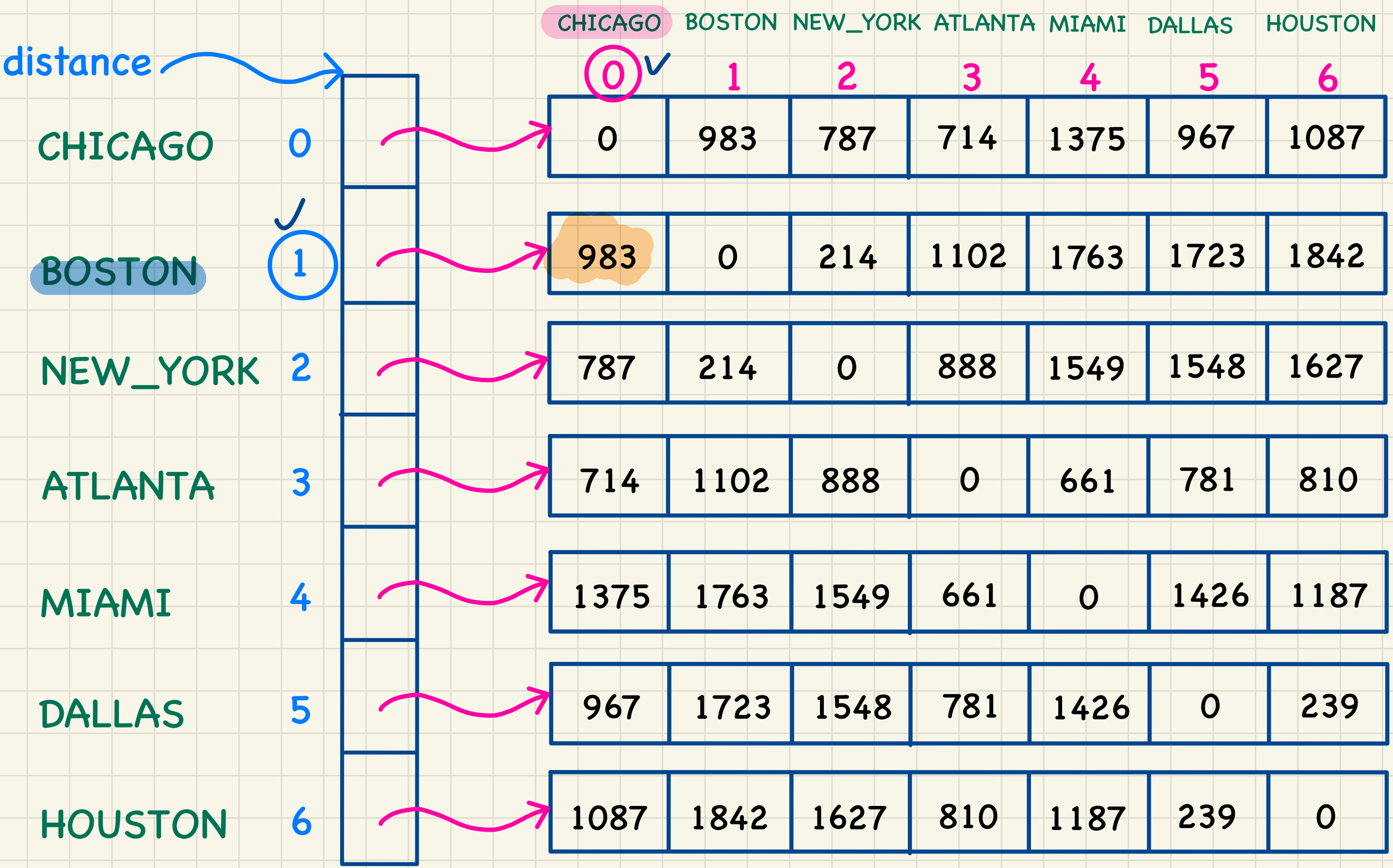
list2a[0] == row1
↳ TRUE.

list2a[1][2] = -1;



Encoing Distances Table via a 2D Array

distance[1][0]



		CHICAGO 0	BOSTON 1	NEW_YORK 2	ATLANTA 3	MIAMI 4	DALLAS 5	HOUSTON 6
CHICAGO	0	0	983	787	714	1375	967	1087
BOSTON	1	983	0	214	1102	1763	1723	1842
NEW_YORK	2	787	214	0	888	1549	1548	1627
ATLANTA	3	714	1102	888	0	661	781	810
MIAMI	4	1375	1763	1549	661	0	1426	1187
DALLAS	5	967	1723	1548	781	1426	0	239
HOUSTON	6	1087	1842	1627	810	1187	239	0

EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 11

Two-Dimensional Arrays
Part I: Utility Methods

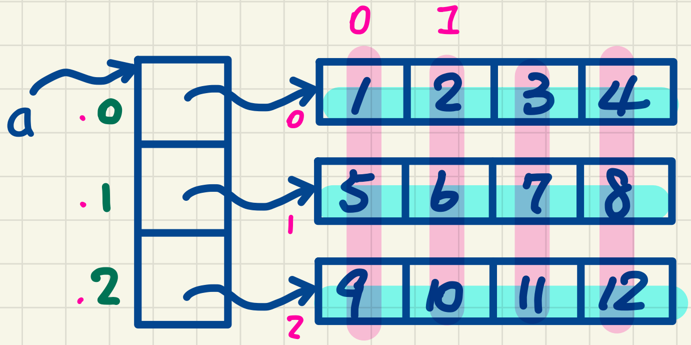
Problem: Given an input 2D array of integers, return a string displaying its values: ✓ row by row vs. ✓ col by col (assuming a rectangle).

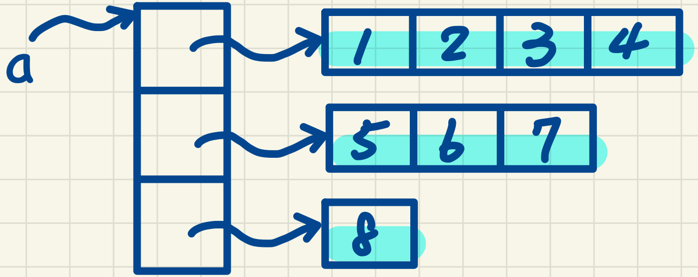
4*3

r	c
0	0
1	
2	
0	1
1	
2	
0	2
1	
2	
0	3
1	
2	

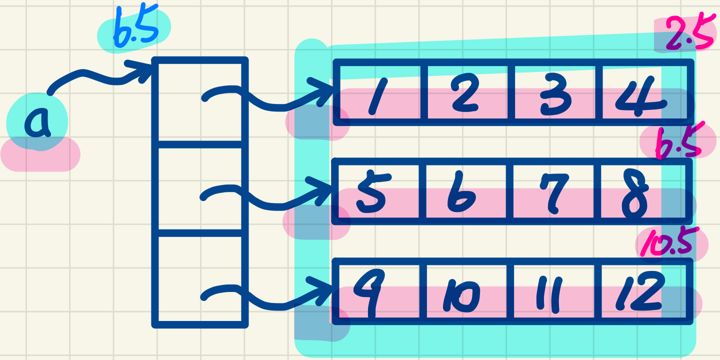
3*4

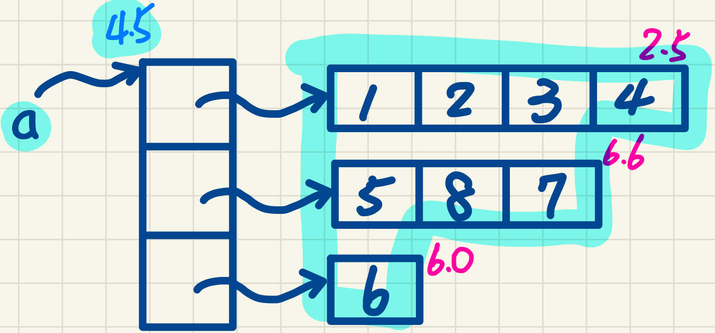
r	c
0	0
	1
	2
	3
1	0
	1
	2
	3
2	0
	1
	2
	3





Problem: Given an input 2D array of integers, return its overall average vs. row averages.





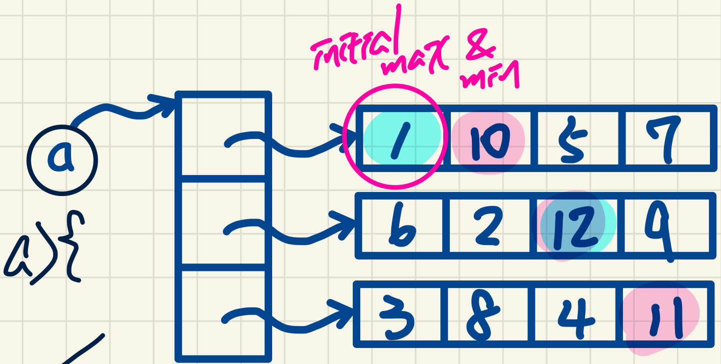
Problem: Given an input 2D array of integers, return an array of size 2 storing its maximum and minimum values.

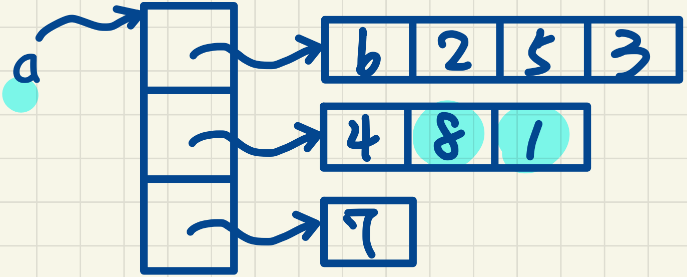
Exercise

① implement
② test

```
public int[] getRowMax(int[][] a){

    ??

}
```

getRowMax(a)
↳ {10, 12, 11}



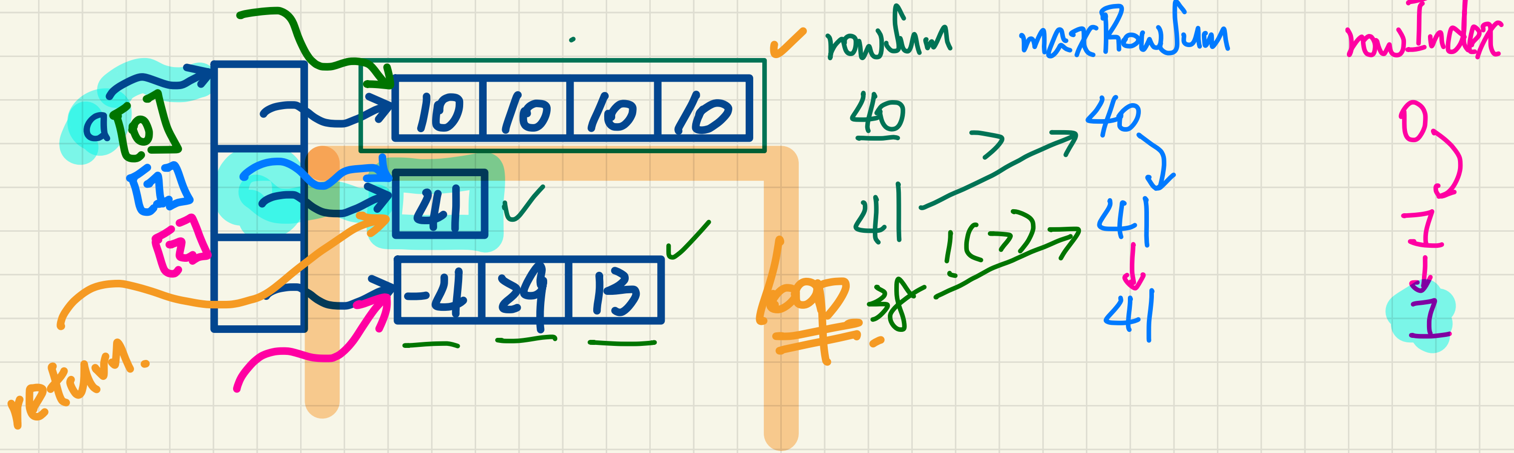


Problem: Given an input 2D array of integers, return the row with the maximum sum.

assumption: 'a' is non-empty.

Exercise

Row with min. sum.



return a[rowIndex]

Problem: Given an input 2D array of integers, determine if elements are all positive.

↳ universal property

↳ as soon as a violation witness is found, exit & conclude false

break

↳ does not scale

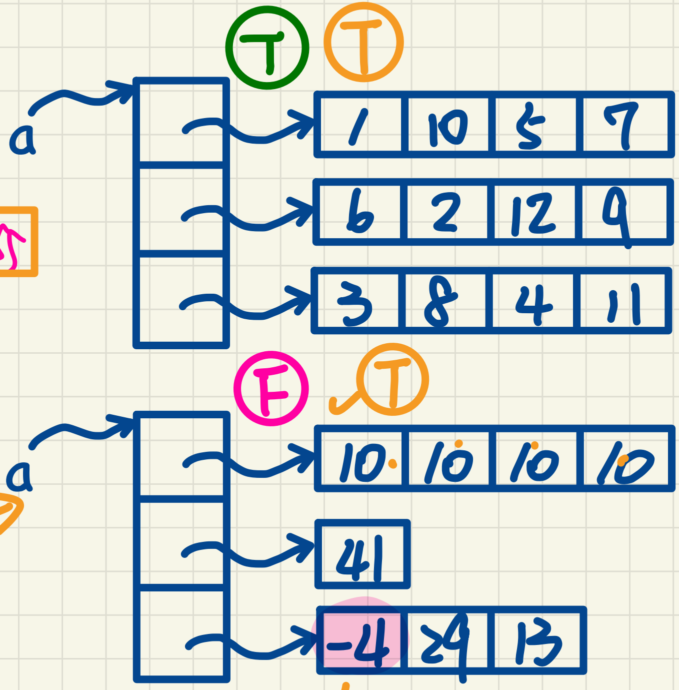
↳ ill-structured.

Exercise:

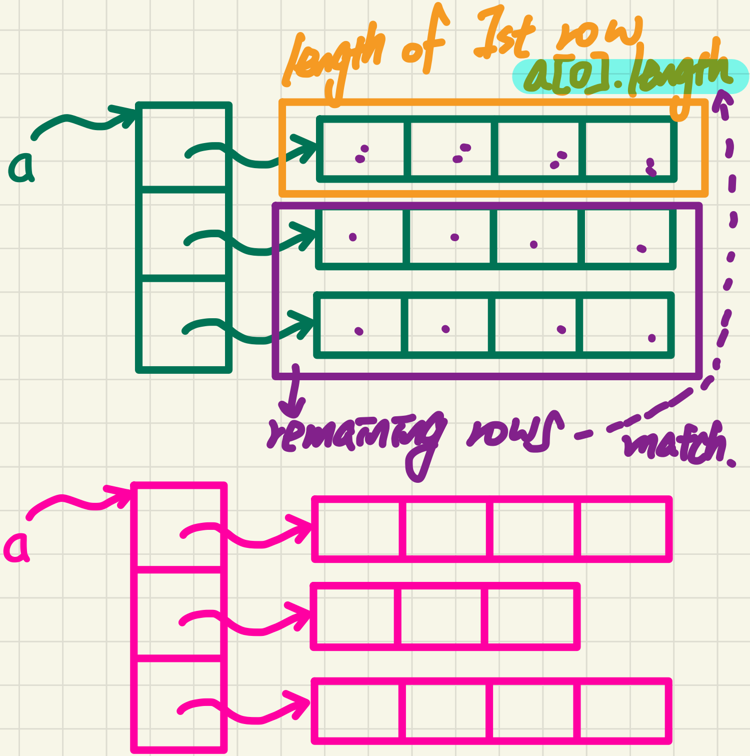
Determine if there is at least one row whose elements are all positive.

existential ≡ 1st dimension

universal ≡ 2nd dimension.



Problem: Given an input 2D array of integers, determine if it is a rectangle (i.e., each row has the same number of columns).

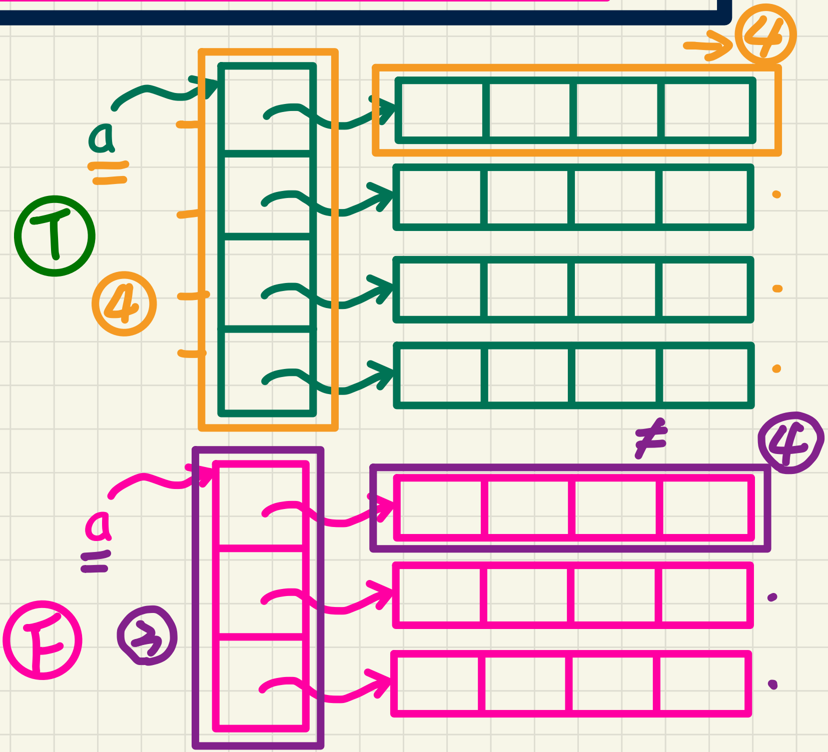


Problem: Given an input 2D array of integers, determine if it is a square (i.e., each row has the same number of columns, and that number is equal to the number of rows of the 2-D array).

RECTANGLE

square

Exercise



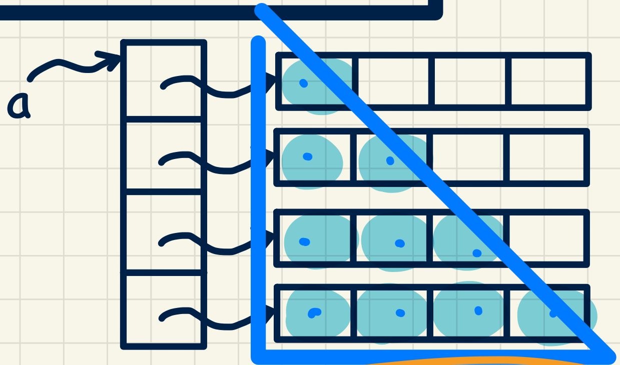
Problem: Given an input 2D array of integers, return a string array of size 2 displaying the lower-left and upper-left triangular areas of elements.

Assumption: The input 2D array is of a square shape.

Lower-Left

a[0][0]
a[1][0] a[1][1]
a[2][0] a[2][1] a[2][2]
a[3][0] a[3][1] a[3][2] a[3][3]

upper bound of 'c' : r

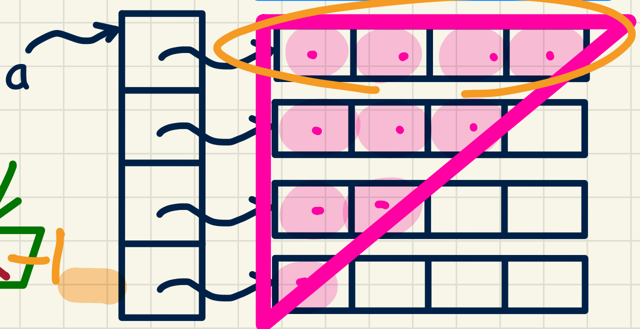


Upper-Left

a[0][0] a[0][1] a[0][2] a[0][3] ✓ 0
a[1][0] a[1][1] a[1][2]
a[2][0] a[2][1]
a[3][0]

upper bound of 'c'

$= \boxed{a[\underset{r}{\cancel{x}}].\text{length} - \underset{r}{\cancel{x}}} - 1$

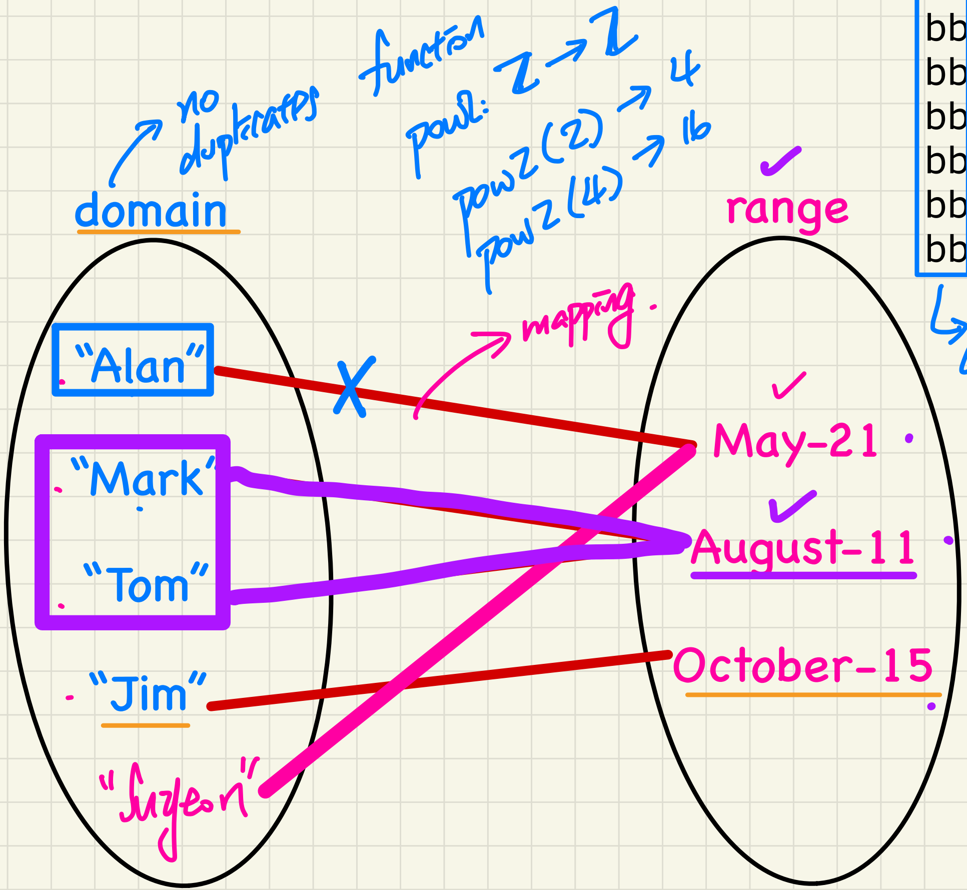


EECS1022 Programming for Mobile Computing (Winter 2021)

Java Tutorials - Week 12

Java API - Developing a Birthday Book using ArrayList vs. Hashtable

An Example Birthday Book 1019

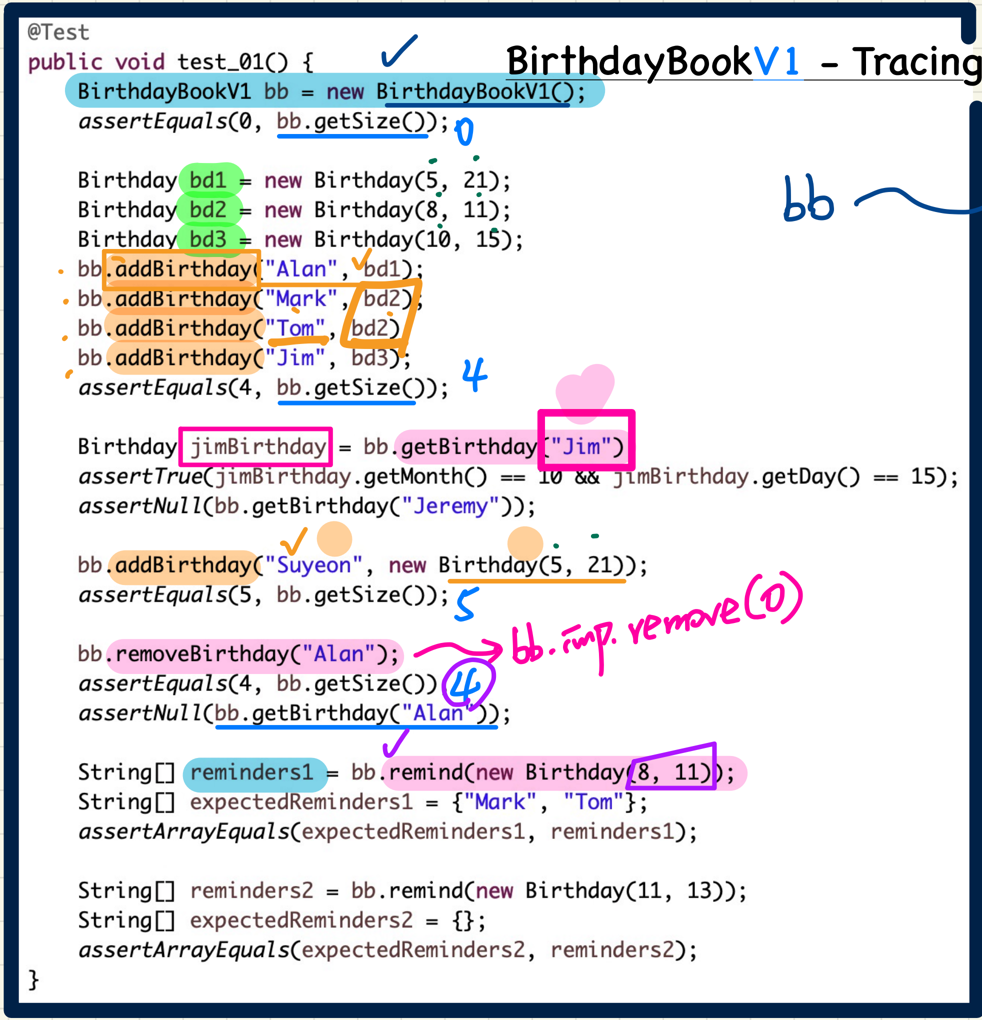


```
bb.getSize()                         4
bb.getBirthday("Jim")             → Oct-15
bb.getBirthday("Jeremy")          → null
bb.addBirthday("Suyeon", May-21)
bb.removeBirthday("Alan")
bb.remind(August-11)              → {"Mark",
                                      "Tom"}
bb.remind(November-13)
                                  → { }
```

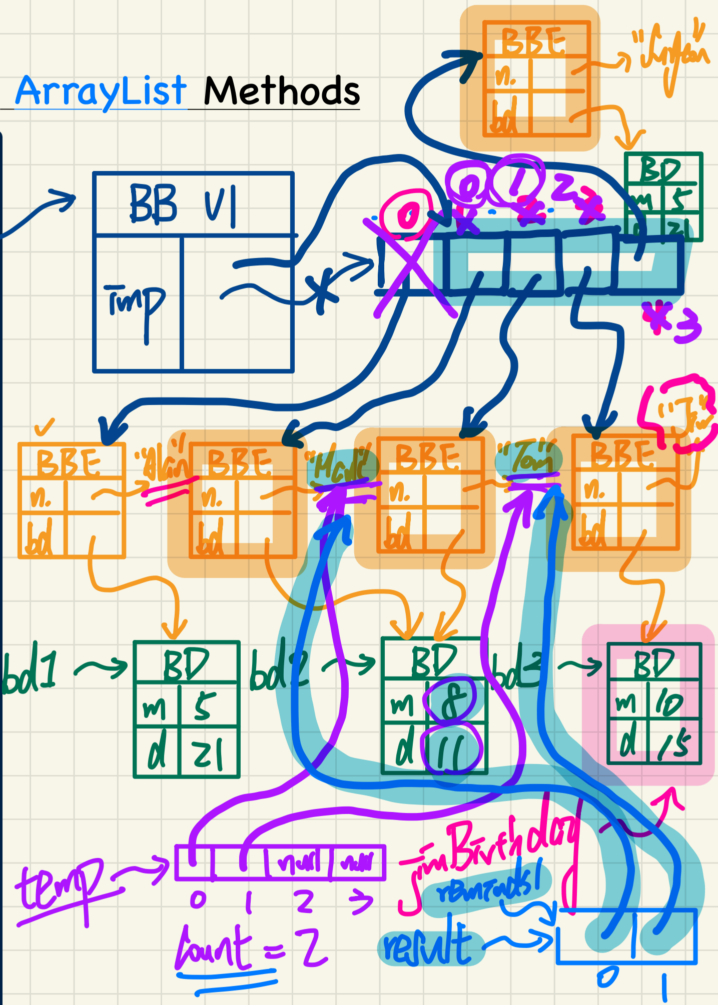
↳ context object

BirthdayBook bb =

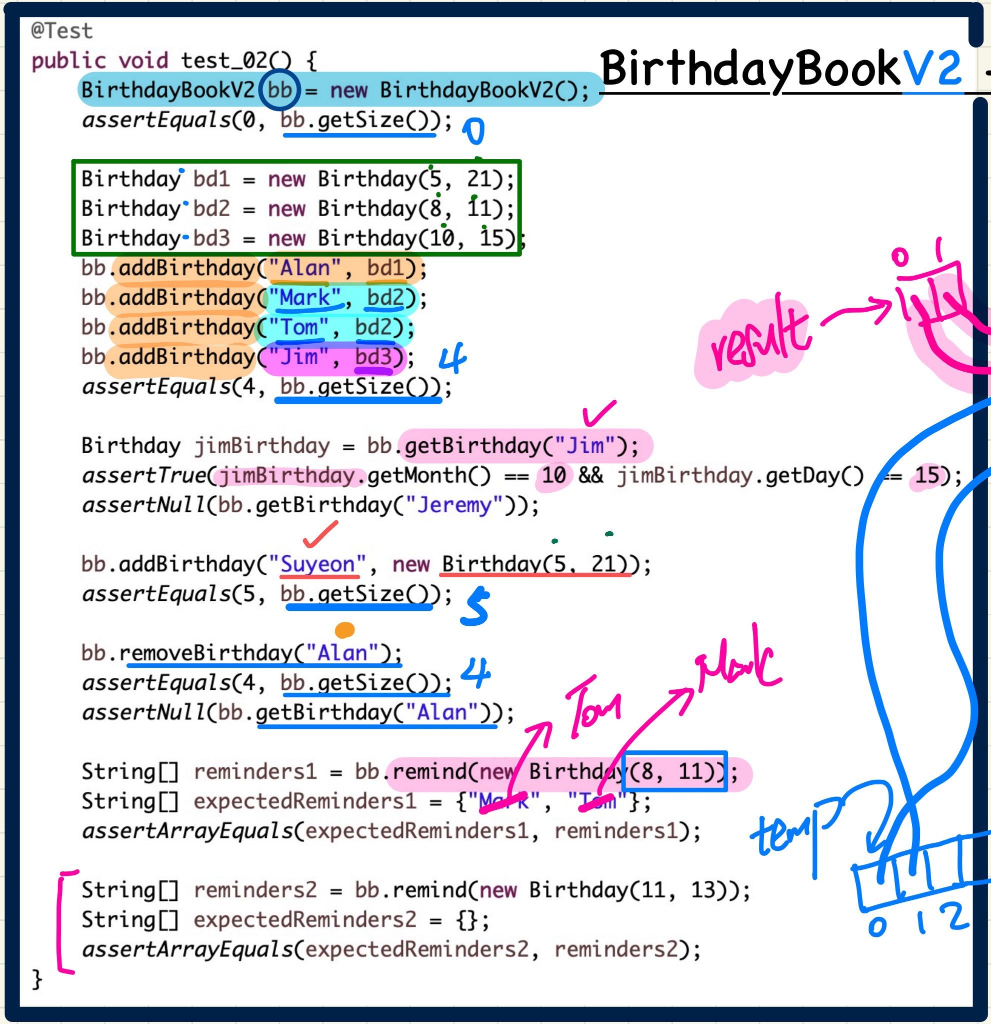
BirthdayBookV1 - Tracing ArrayList Methods



```
@Test
public void test_01() {
    BirthdayBookV1 bb = new BirthdayBookV1();
    assertEquals(0, bb.getSize());

    Birthday bd1 = new Birthday(5, 21);
    Birthday bd2 = new Birthday(8, 11);
    Birthday bd3 = new Birthday(10, 15);
    bb.addBirthday("Alan", bd1);
    bb.addBirthday("Mark", bd2);
    bb.addBirthday("Tom", bd2);
    bb.addBirthday("Jim", bd3);
    assertEquals(4, bb.getSize());

    Birthday jimBirthday = bb.getBirthday("Jim");
    assertTrue(jimBirthday.getMonth() == 10 && jimBirthday.getDay() == 15);
    assertNull(bb.getBirthday("Jeremy"));

    bb.addBirthday("Suyeon", new Birthday(5, 21));
    assertEquals(5, bb.getSize());

    bb.removeBirthday("Alan");
    assertEquals(4, bb.getSize());
    assertNull(bb.getBirthday("Alan"));

    String[] reminders1 = bb.remind(new Birthday(8, 11));
    String[] expectedReminders1 = {"Mark", "Tom"};
    assertArrayEquals(expectedReminders1, reminders1);

    String[] reminders2 = bb.remind(new Birthday(11, 13));
    String[] expectedReminders2 = {};
    assertArrayEquals(expectedReminders2, reminders2);
}
```

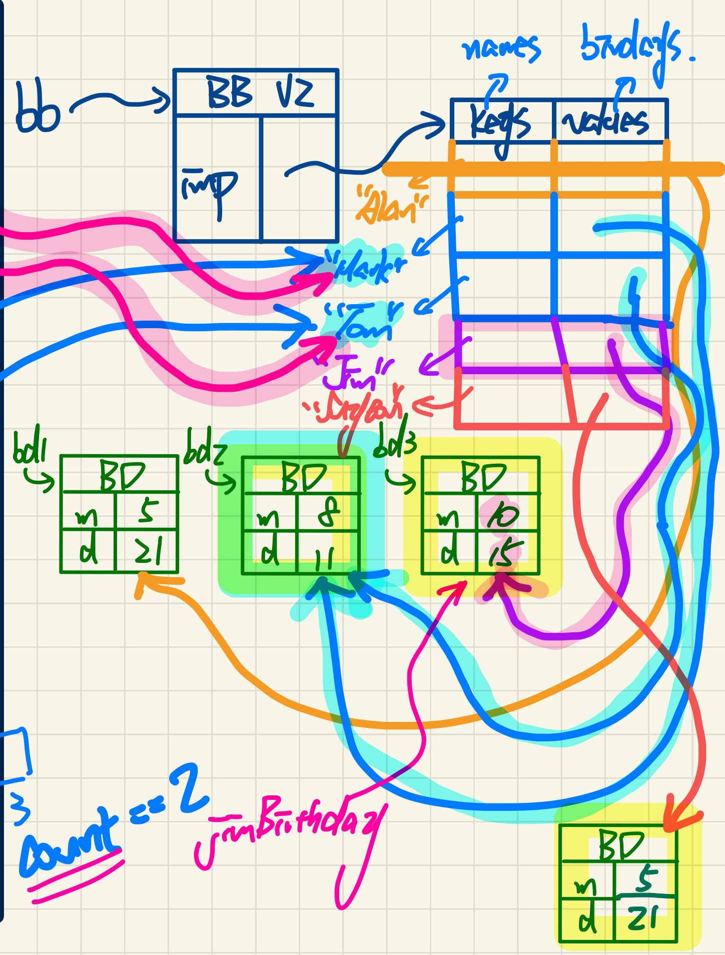


BirthdayBookV2 - Tracing Hashtable Methods



```
@Test
public void test_02() {
    BirthdayBookV2 bb = new BirthdayBookV2();
    assertEquals(0, bb.getSize());

    Birthday bd1 = new Birthday(5, 21);
    Birthday bd2 = new Birthday(8, 11);
    Birthday bd3 = new Birthday(10, 15);
    bb.addBirthday("Alan", bd1);
    bb.addBirthday("Mark", bd2);
    bb.addBirthday("Tom", bd2);
    bb.addBirthday("Jim", bd3);
    assertEquals(4, bb.getSize());

    Birthday jimBirthday = bb.getBirthday("Jim");
    assertTrue(jimBirthday.getMonth() == 10 && jimBirthday.getDay() == 15);
    assertNull(bb.getBirthday("Jeremy"));

    bb.addBirthday("Suyeon", new Birthday(5, 21));
    assertEquals(5, bb.getSize());

    bb.removeBirthday("Alan");
    assertEquals(4, bb.getSize());
    assertNull(bb.getBirthday("Alan"));

    String[] reminders1 = bb.remind(new Birthday(8, 11));
    String[] expectedReminders1 = {"Mark", "Tom"};
    assertArrayEquals(expectedReminders1, reminders1);

    String[] reminders2 = bb.remind(new Birthday(11, 13));
    String[] expectedReminders2 = {};
    assertArrayEquals(expectedReminders2, reminders2);
}
```

0
4
5
4
Tom
Mark



result
0
1
bb
BB V2
imp
names
birthdays.
keys
values
"Alan"
"Mark"
"Tom"
"Jim"
"Suyeon"
bd1
bd2
bd3
BD
m 5
d 21
BD
m 8
d 11
BD
m 10
d 15
temp
0 1 2 3
count == 2
jimBirthday
BD
m 5
d 21