

# COMP 110/L Lecture 15

Mahdi Ebrahimi

Slides adapted from Dr. Kyle Dewey

# Outline

- Loops with arrays

# Loops with Arrays

# Example

`LoopsWithArray.java`

# Loops with Arrays

Can *iterate* through arrays using loops

# Loops with Arrays

Can *iterate* through arrays using loops

```
for (int x = 0; x < arr.length; x++) {  
    System.out.println(x);  
}
```

# Loops with Arrays

Can *iterate* through arrays using loops

**Not** `<=`, since arrays start from 0

```
for (int x = 0; x < arr.length; x++) {  
    System.out.println(x);  
}
```

**Example:**

`PrintArgs.java`



# Computing a Single Result

Common pattern: build a single result via iteration.  
Update this result for each iteration.

# Computing a Single Result

Common pattern: build a single result via iteration.  
Update this result for each iteration.

Example: arithmetic product

# Computing a Single Result

Common pattern: build a single result via iteration.  
Update this result for each iteration.

Example: arithmetic product

{ }

# Computing a Single Result

Common pattern: build a single result via iteration.  
Update this result for each iteration.

Example: arithmetic product

{ }

1

# Computing a Single Result

Common pattern: build a single result via iteration.  
Update this result for each iteration.

Example: arithmetic product

{ }

1

{ 5 }

# Computing a Single Result

Common pattern: build a single result via iteration.  
Update this result for each iteration.

Example: arithmetic product

{ }

1

{ 5 }

1 \* 5

# Computing a Single Result

Common pattern: build a single result via iteration.  
Update this result for each iteration.

Example: arithmetic product

{ }

1

{ 5 }

1 \* 5

5

# Example: arithmetic product



# Example: arithmetic product

$\{5, 8\}$

## Example: arithmetic product

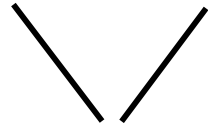
$\{5, 8\}$

$1 * 5 * 8$

# Example: arithmetic product

$\{5, 8\}$

1 \* 5 \* 8

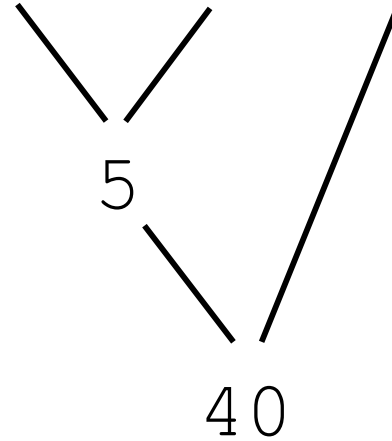


5

# Example: arithmetic product

{ 5, 8 }

1 \* 5 \* 8



# Example: arithmetic product

## Example: arithmetic product

$\{5, 8, 3\}$

## Example: arithmetic product

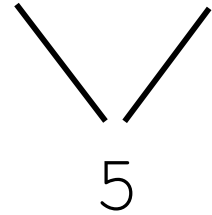
{5, 8, 3}

1 \* 5 \* 8 \* 3

# Example: arithmetic product

{ 5, 8, 3 }

1 \* 5 \* 8 \* 3

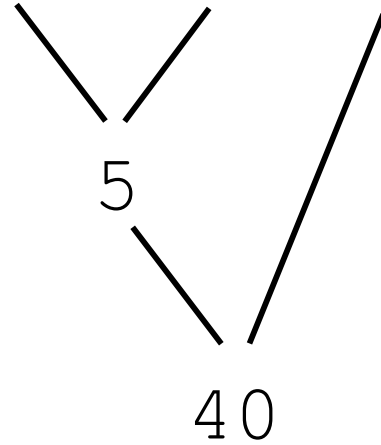




# Example: arithmetic product

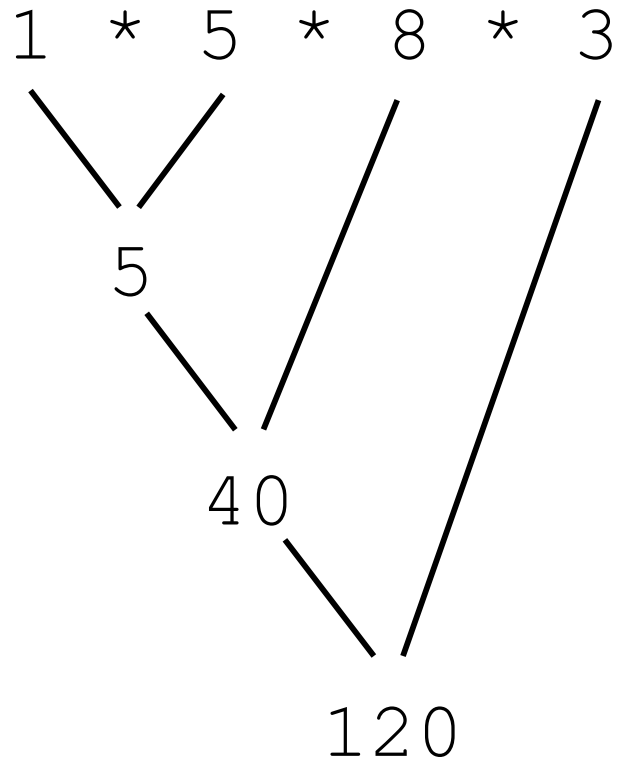
{ 5 , 8 , 3 }

1 \* 5 \* 8 \* 3



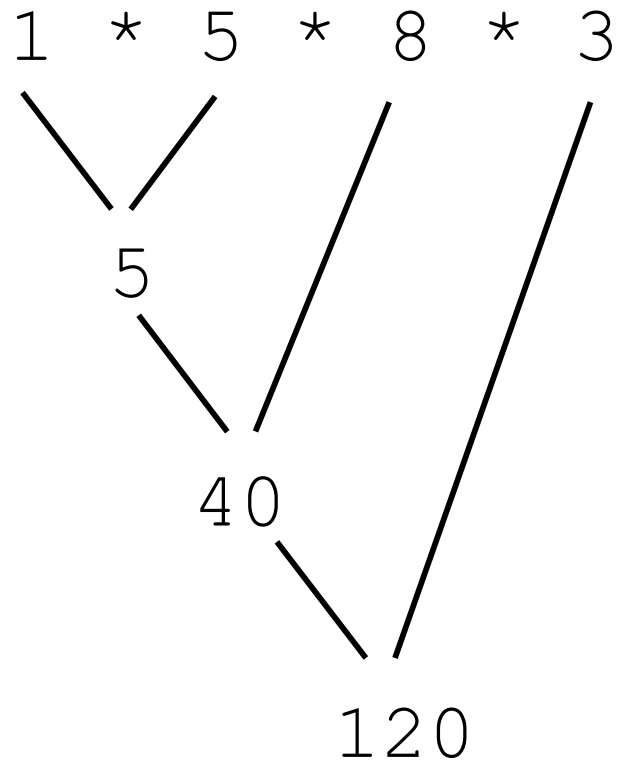
# Example: arithmetic product

{ 5, 8, 3 }



# In Code

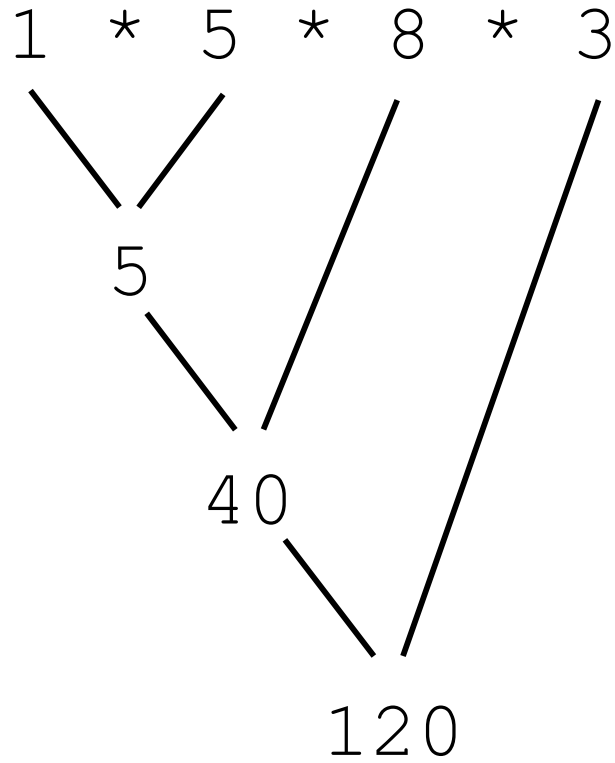
{ 5, 8, 3 }



Variables needed:

# In Code

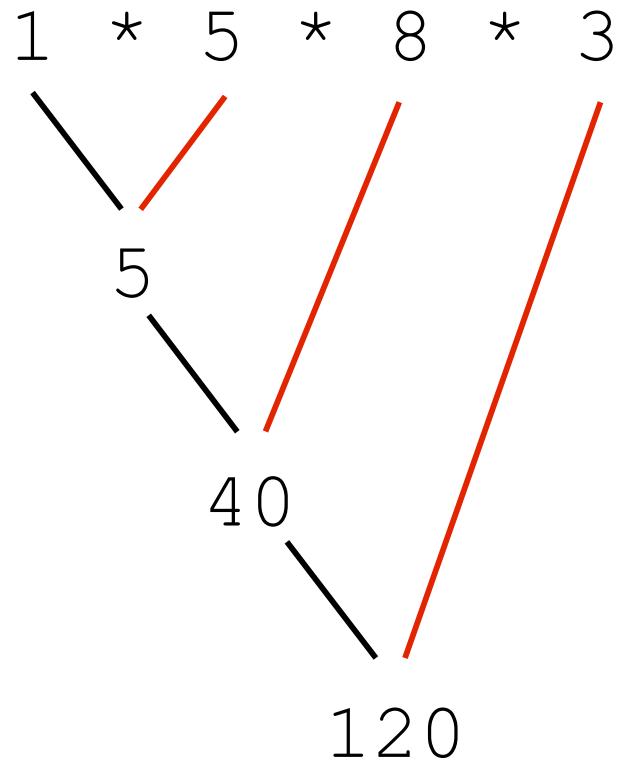
{5, 8, 3}



Variables needed: **array**

# In Code

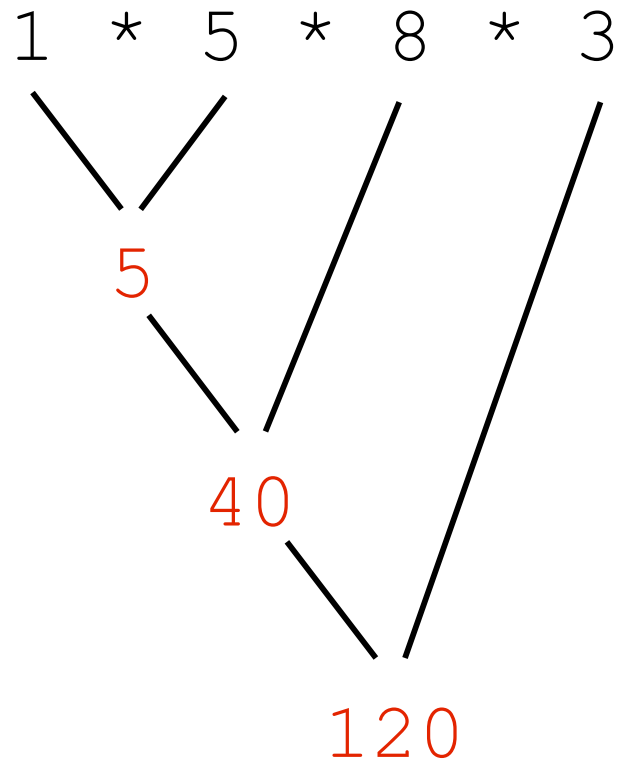
{5, 8, 3}



Variables needed: array, **position in array**

# In Code

{5, 8, 3}



Variables needed: array, position in array, **result**

# Example

- `Product.java`
- `ProductTest.java`

Another example: arithmetic sum



Another example: arithmetic sum

{ }

## Another example: arithmetic sum

{ }

0

## Another example: arithmetic sum

{ }

0

{ 2 }

## Another example: arithmetic sum

{ }

0

{ 2 }

0 + 2

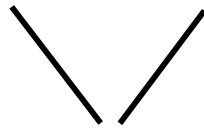
## Another example: arithmetic sum

{ }

0

{ 2 }

0 + 2



2

Another example: arithmetic sum

$\{2, 5\}$

## Another example: arithmetic sum

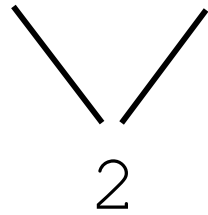
$$\{2, 5\}$$

$$0 + 2 + 5$$

## Another example: arithmetic sum

$$\{2, 5\}$$

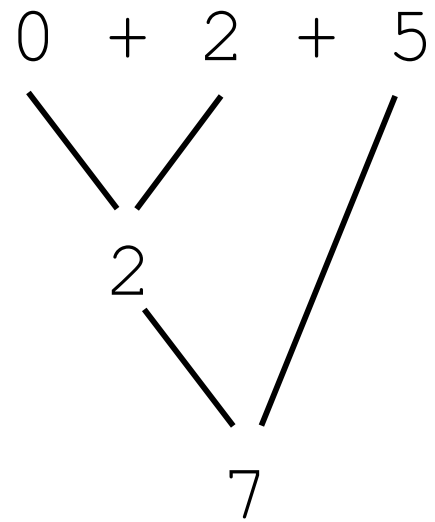
$$0 + 2 + 5$$





## Another example: arithmetic sum

$\{2, 5\}$



Another example: arithmetic sum

$\{2, 5, 9\}$

## Another example: arithmetic sum

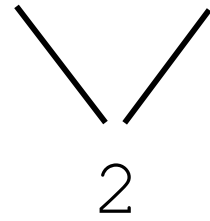
$\{2, 5, 9\}$

$$0 + 2 + 5 + 9$$

## Another example: arithmetic sum

$\{2, 5, 9\}$

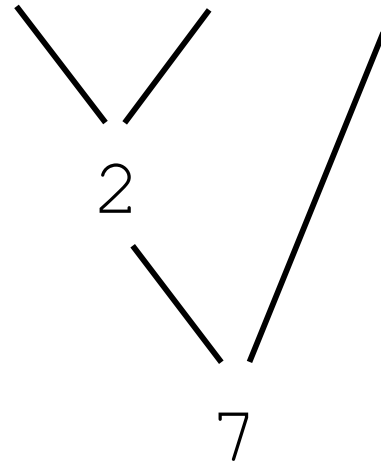
$0 + 2 + 5 + 9$



## Another example: arithmetic sum

$\{2, 5, 9\}$

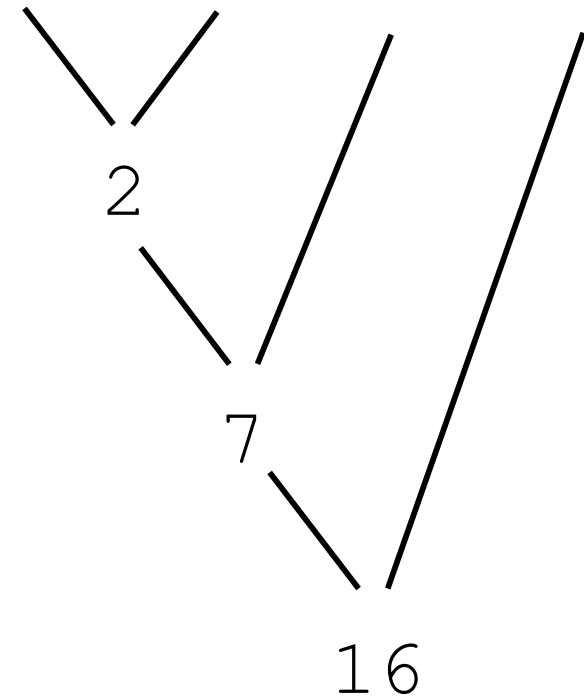
$0 + 2 + 5 + 9$



# Another example: arithmetic sum

{2, 5, 9}

0 + 2 + 5 + 9



# General Pattern

# General Pattern

```
ResultType result = initialResult;
```



# General Pattern

```
ResultType result = initialResult;  
for (int index = whereToStart;
```

# General Pattern

```
ResultType result = initialResult;  
for (int index = whereToStart;  
     index < whereToEnd;
```

# General Pattern

```
ResultType result = initialResult;  
for (int index = whereToStart;  
     index < whereToEnd;  
     index++) {
```

# General Pattern

```
ResultType result = initialResult;  
for (int index = whereToStart;  
     index < whereToEnd;  
     index++) {  
    result = oneStep(array[index],  
                    result);  
}
```