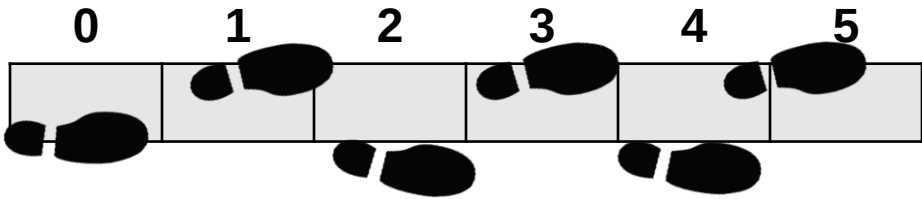
Structured Data in Java
The `foreach` Loop

Array Processing

- **Array traversal**: Accessing each element of an array

```
for (int i=0; i<arr.length; i=i+1) {
    do something with arr[i];
}
```



- Examples:
  - printing the elements
  - searching for a specific value
  - computing the sum or product
  - etc.

The For-Each Loop

- Java added another loop construct that can simplify some array loops:

```
for (type var : arr) {
    statements using var;
}
```

- This is equivalent to:

```
for (int i=0; i<arr.length; i++) {
    type var = arr[i];
    statements using var;
}
```

The For-Each Loop

- Example: compute the average of all the numbers

```
int total=0;
for (int num : numbers) {
  total += num;
}
double average = (double)total/numbers.length;
```

Read as: “for each integer **num** in the array **numbers** do…”

“add num to the total”

Another Example

```
public void process() {
    int[] points = {125, 132, 95, 116, 110};
    int hiScore = max(points);
    out.println("The highest score was " + hiScore);
}

public static int max(int[] numbers) {
    int maxSoFar = numbers[0];
    for (int num : numbers) {
        if (num > maxSoFar) {
            maxSoFar = num;
        }
    }
    return maxSoFar;
}
```

The For Loop vs. For-Each Loop

- Use of the `for` loop is always safe
- Use of the `for-each` loop can make code more readable
- The `for-each` loop is applicable to other collections in Java, even when they do not support indexing
- The `for-each` loop is preferred over the `for` loop in most cases, but it has some limitations…

For-Each Limitations

- `For-each` loops are not appropriate when you want to modify the array:

```
for (int num : numbers) {
  num = num*2;  // only changes num, not the array
}
```

- `For-each` loops are not appropriate when you need the array index

```
for (int num : numbers) {
  if (num == target) {
     return ???;   // do not know the index of num
  }
}
```

For-Each Limitations

- `For-each` only iterates forward over the array in single steps

```
// cannot be converted to a for-each loop
for (int i=numbers.length-1; i>=0; i--) {
  out.println(numbers[i]);
}
```

- `For-each` cannot process two structures at once

```
// cannot be easily converted to a for-each loop
for (int i=0; i<numbers.length; i++) {
  if (numbers[i] == arr[i]) {
     ...
  }
}
```