CS210 Discussion

Week 3

Attendance

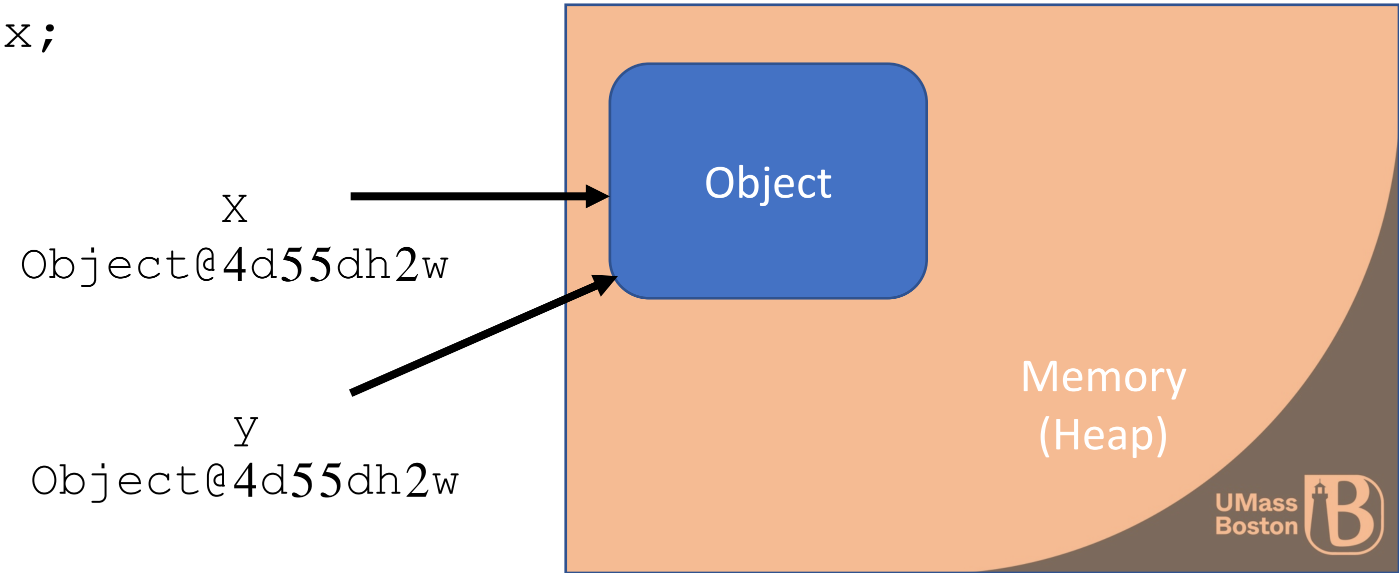
Today

- References
- Linked lists
- Generics
- Iterators
- Finish through exercise 3
  - Show us passing Gradescope tests to leave early

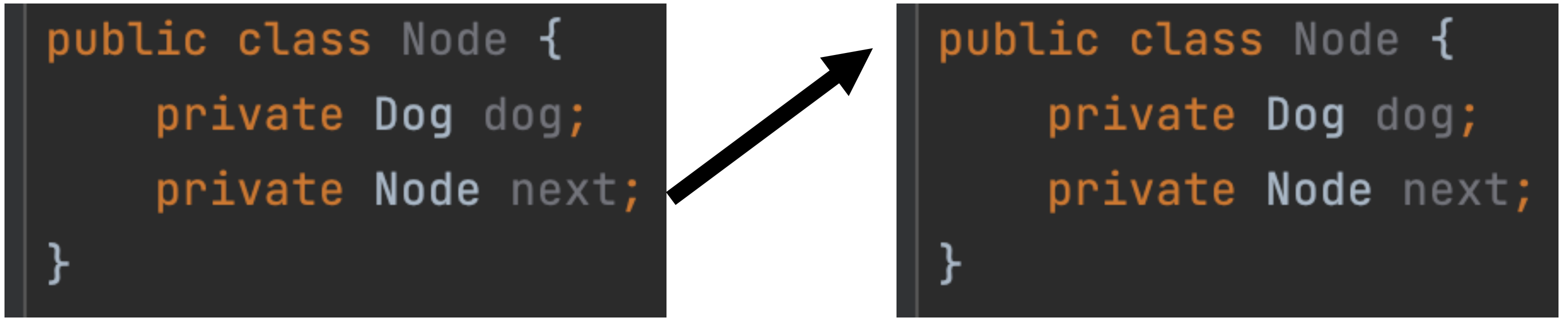
References in Java

- Doesn’t have explicit pointer data types like in C
- Object variables are essentially pointers
  - Hence why they’re also called “reference” types
  - You can treat them like the objects they point to

```
Object x = new Object();
Object y = x;
```



Linked List of Dogs

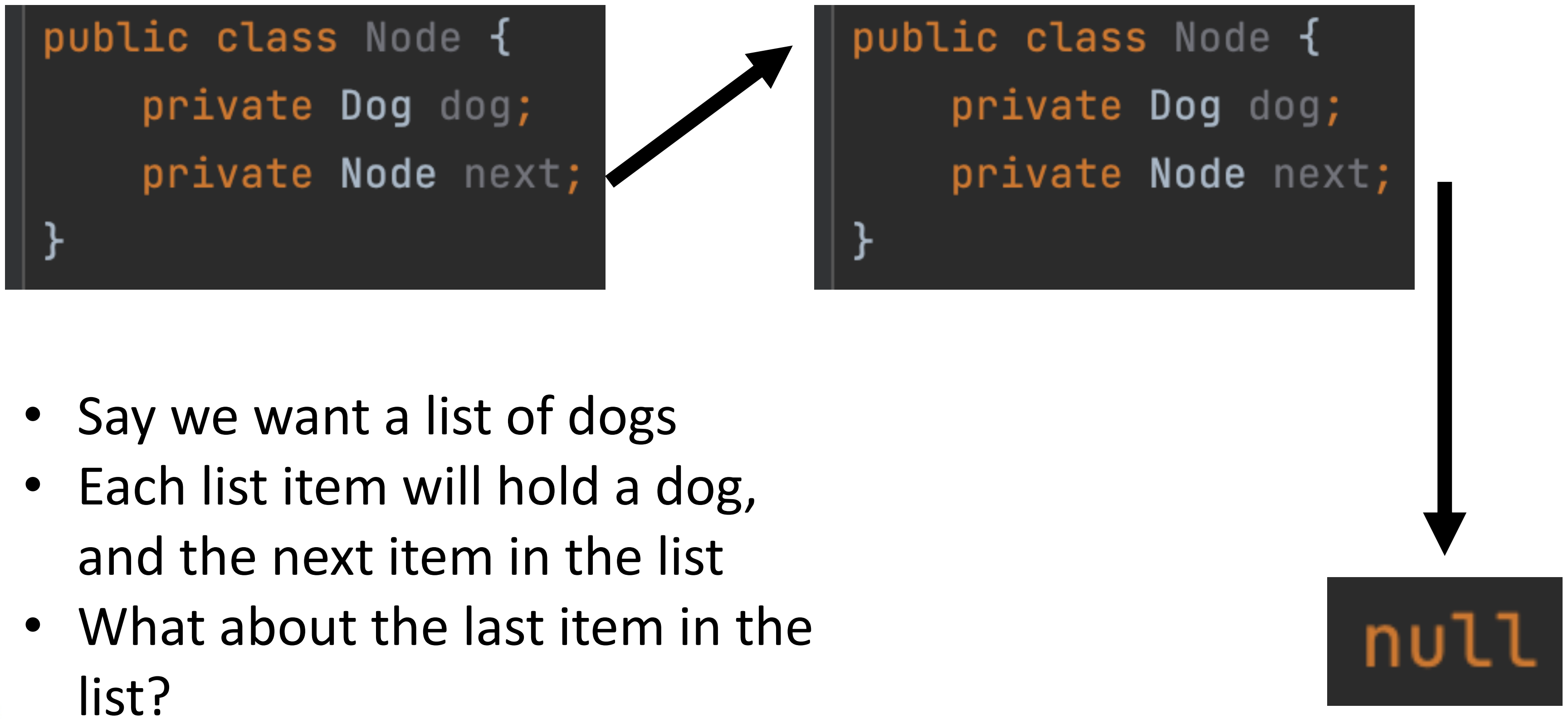


```java
public class Node {
    private Dog dog;
    private Node next;
}
```

```java
public class Node {
    private Dog dog;
    private Node next;
}
```

- Say we want a list of dogs
- Each list item will hold a dog, and the next item in the list
- What about the last item in the list?

Linked List of Dogs



```
public class Node {
    private Dog dog;
    private Node next;
}
```

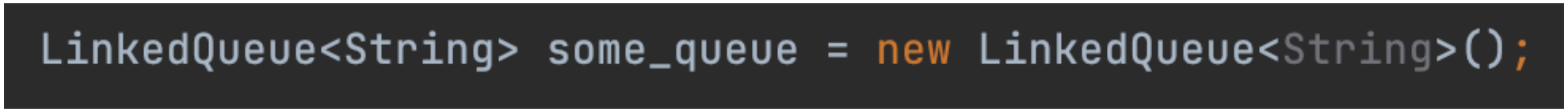
```
public class Node {
    private Dog dog;
    private Node next;
}
```

```
null
```

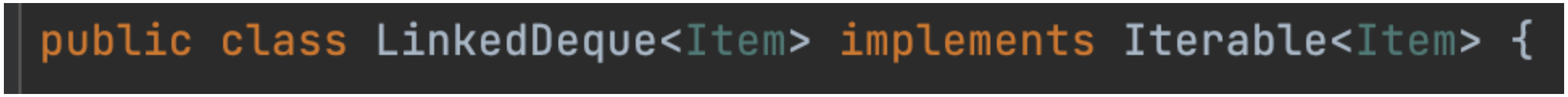
- Say we want a list of dogs
- Each list item will hold a dog, and the next item in the list
- What about the last item in the list?

Generics

- Placeholder types
- Allows you to define classes/methods that operate on any type

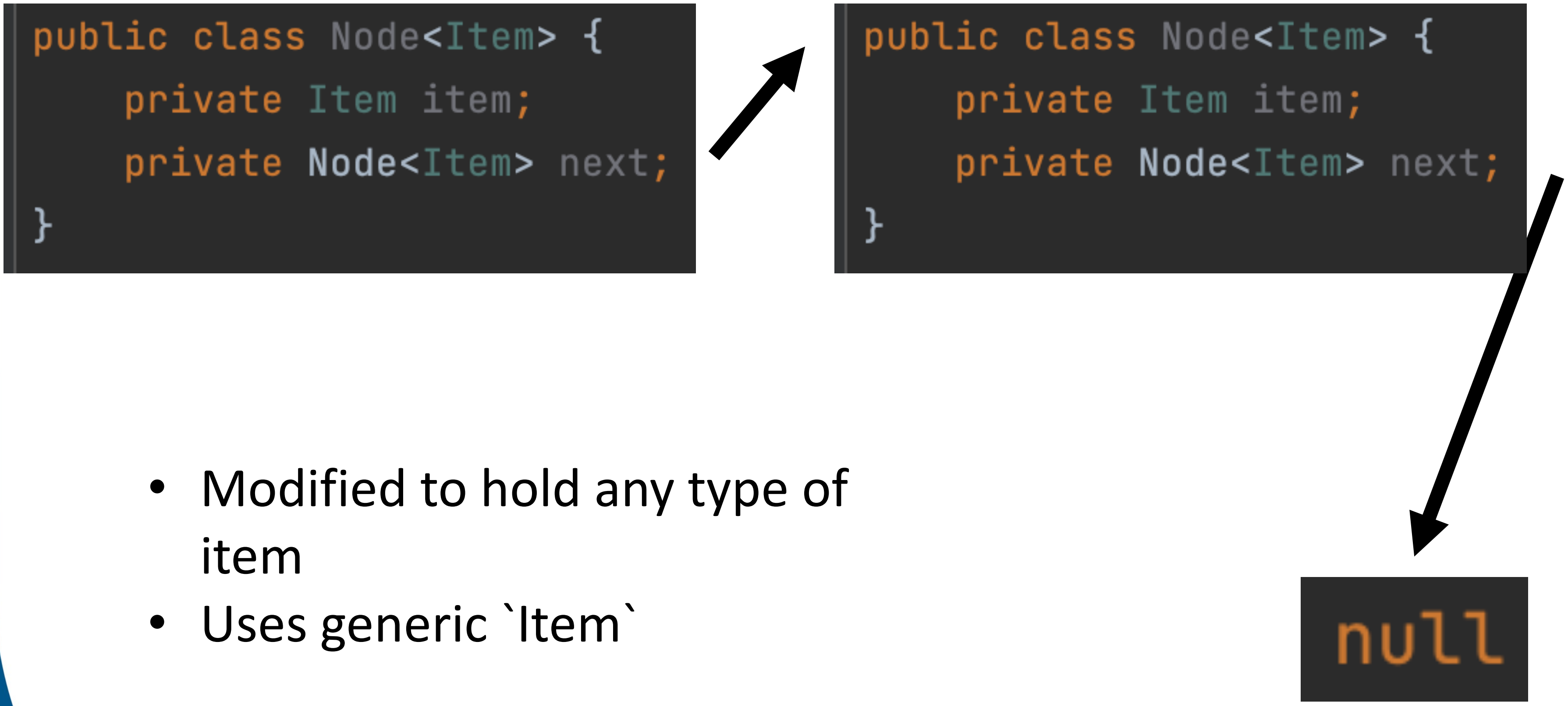


```
LinkedQueue<String> some_queue = new LinkedQueue<String>();
```



```
public class LinkedDeque<Item> implements Iterable<Item> {
```

Linked List of Anything



```java
public class Node<Item> {
    private Item item;
    private Node<Item> next;
}
```

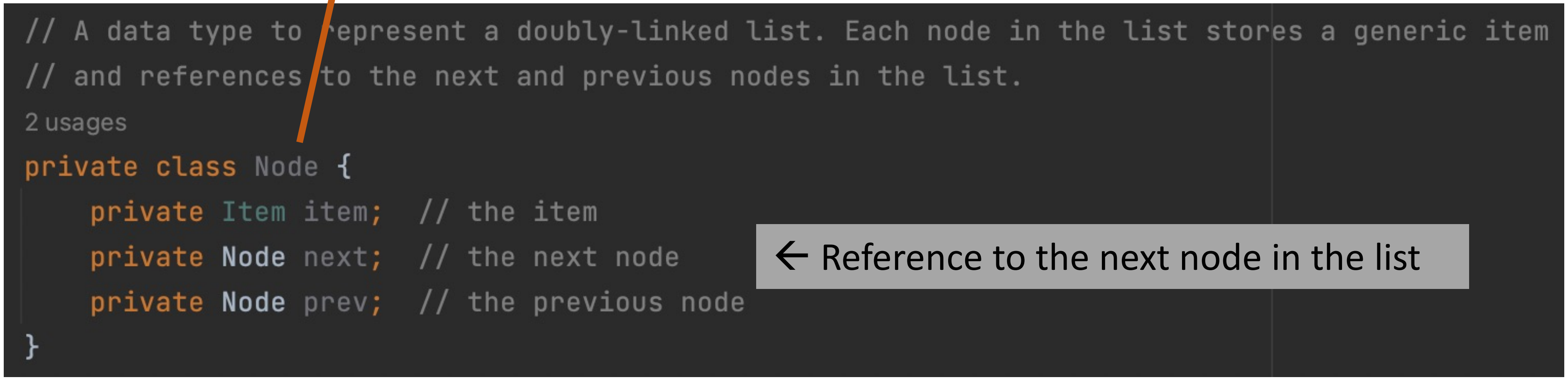
```java
public class Node<Item> {
    private Item item;
    private Node<Item> next;
}
```

```java
null
```

- Modified to hold any type of item
- Uses generic `Item`

Doubly Linked List

$$\begin{tikzcd}
 & \text{First} & & & & \text{Last} & \\
\texttt{null} & \boxed{item_1} \arrow[l] \arrow[r, leftrightarrow] & \boxed{item_2} \arrow[r, leftrightarrow] & \boxed{item_3} \arrow[r, leftrightarrow] & \cdots \arrow[r, leftrightarrow] & \boxed{item_n} \arrow[r] & \texttt{null}
\end{tikzcd}$$



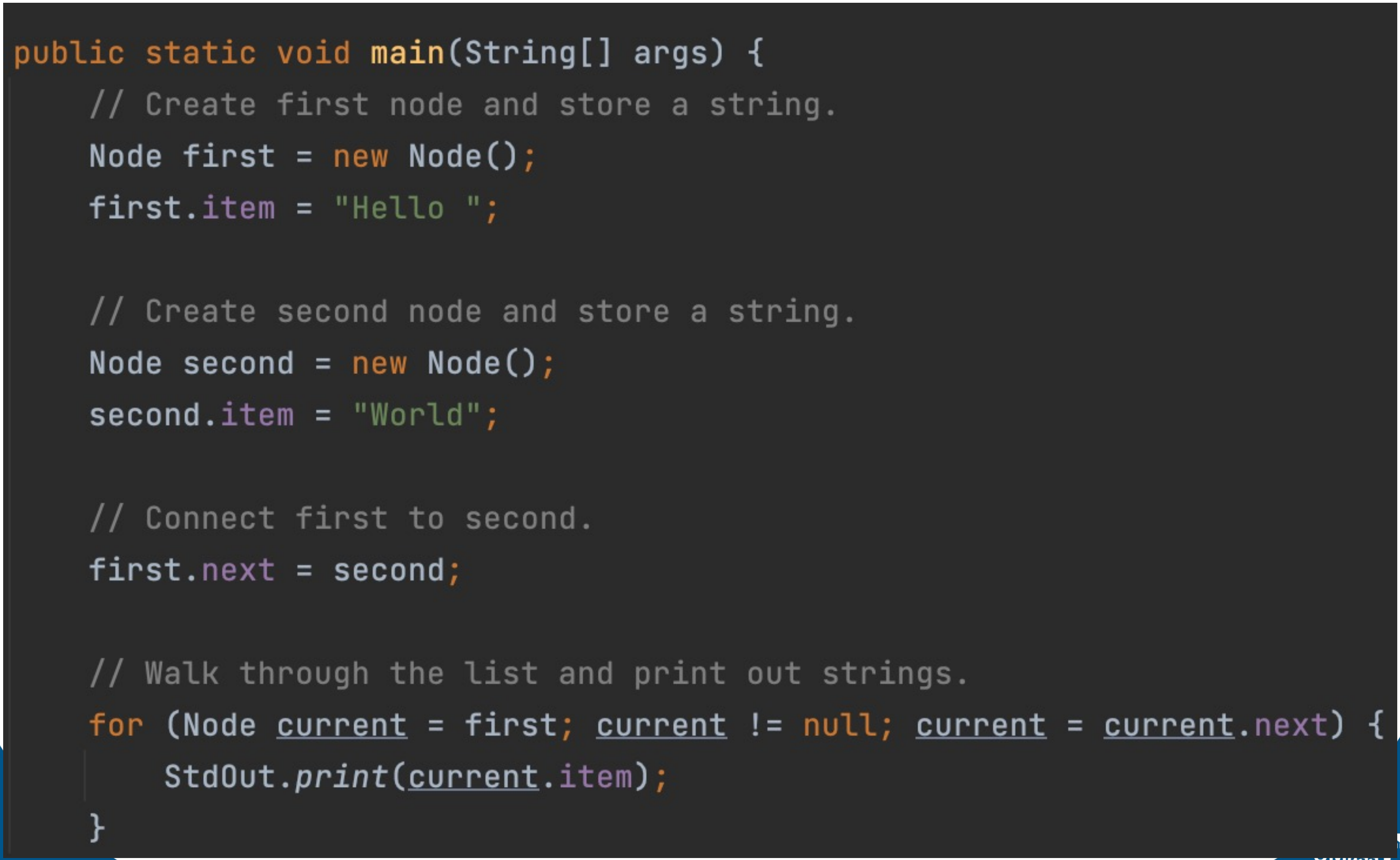
```java
// A data type to represent a doubly-linked list. Each node in the list stores a generic item
// and references to the next and previous nodes in the list.
2 usages
private class Node {
    private Item item;   // the item
    private Node next;   // the next node
    private Node prev;   // the previous node
}
```

← Reference to the next node in the list

Which item does ‘next’ point to?

What about ‘prev’?

Traversal Example

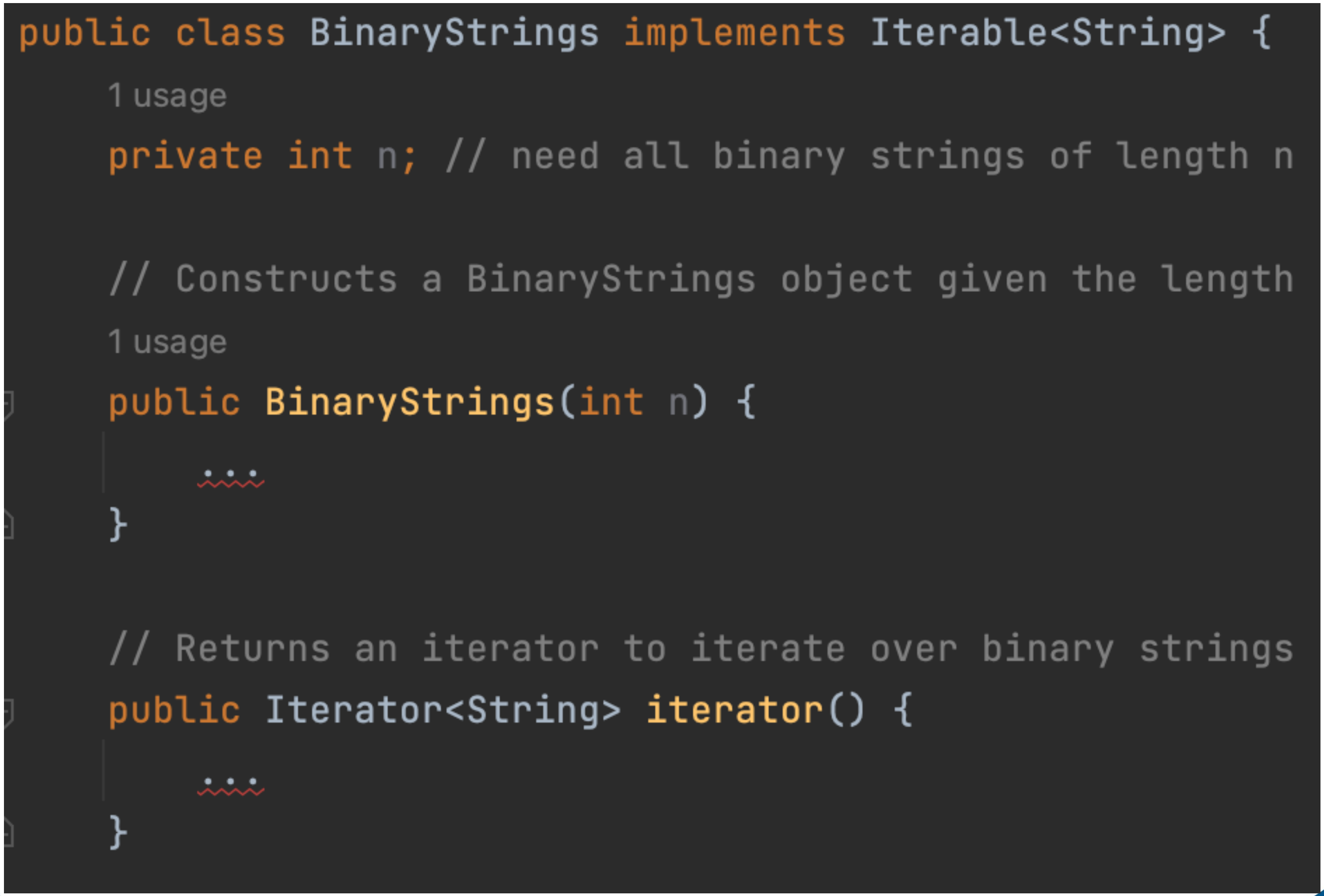


```
public static void main(String[] args) {
    // Create first node and store a string.
    Node first = new Node();
    first.item = "Hello ";

    // Create second node and store a string.
    Node second = new Node();
    second.item = "World";

    // Connect first to second.
    first.next = second;

    // Walk through the list and print out strings.
    for (Node current = first; current != null; current = current.next) {
        StdOut.print(current.item);
    }
```

Iterable and Iterator

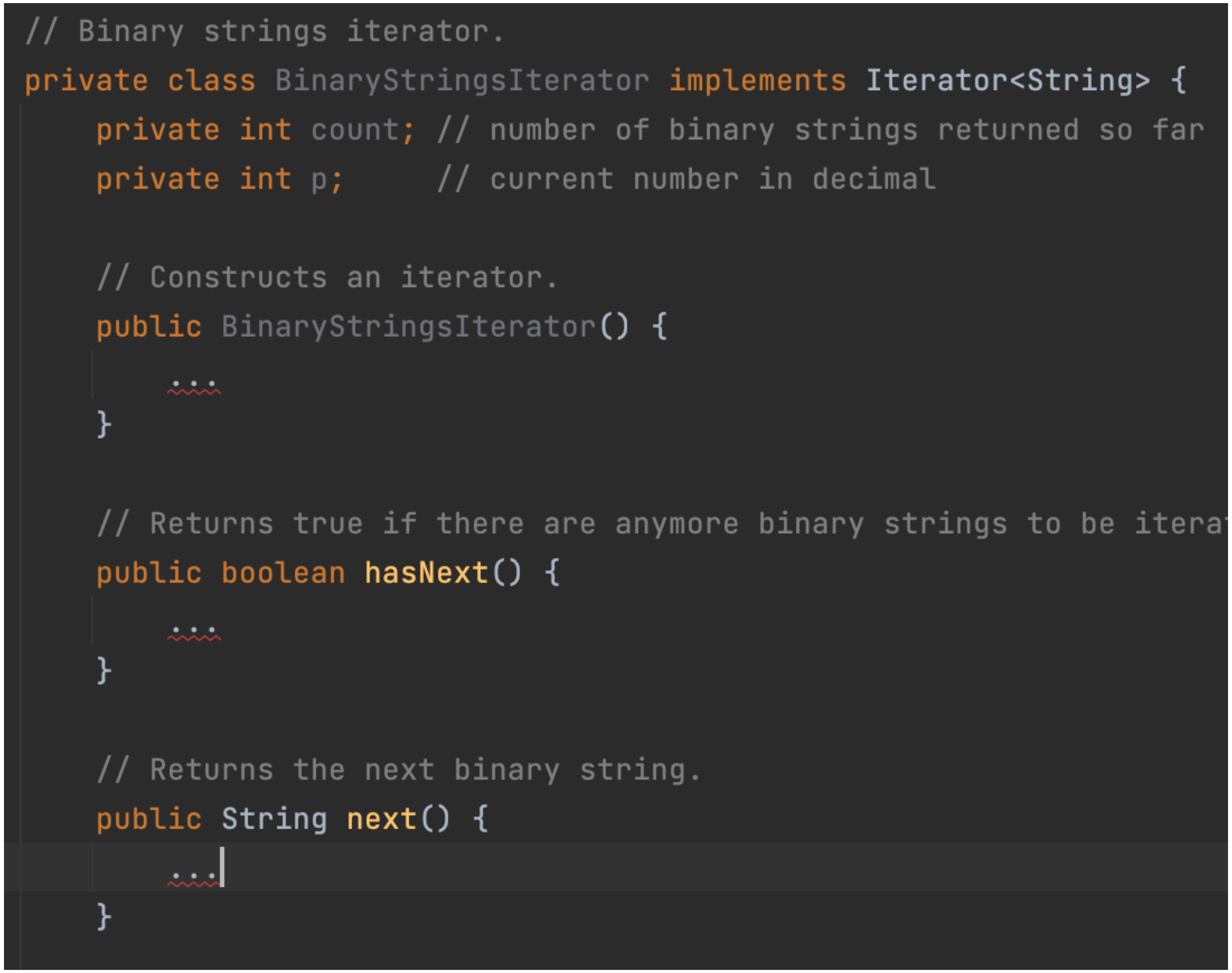
- Java interfaces
    - Contracts
- Represents a sequence of values
- Must implement certain methods
- Iterable can produce an iterator
- Iterator can produce a sequence of values
    - Does the work of iterating

Iterable



```
public class BinaryStrings implements Iterable<String> {
    1 usage
    private int n; // need all binary strings of length n

    // Constructs a BinaryStrings object given the length
    1 usage
    public BinaryStrings(int n) {
        ...
    }

    // Returns an iterator to iterate over binary strings
    public Iterator<String> iterator() {
        ...
    }
```

Iterator



```
// Binary strings iterator.
private class BinaryStringsIterator implements Iterator<String> {
    private int count; // number of binary strings returned so far
    private int p;     // current number in decimal

    // Constructs an iterator.
    public BinaryStringsIterator() {
        ...
    }

    // Returns true if there are anymore binary strings to be itera
    public boolean hasNext() {
        ...
    }

    // Returns the next binary string.
    public String next() {
        ...
    }
```

Questions?

Exercise Hints

- List iteration with a `for` loop

```
for (Node current = first; current != null; current = current.next) {
```

  - Can also be done with a `while` loop
- Iterable contains the data, iterator lists it out
- Iterable mostly just needs to return an iterator

```
public Iterator<Dog> iterator() { return new DogIterator(); }
```

- Nested classes can refer to the instance variables of the outer class
- Smallest and largest integer values possible in Java
  - `Integer.MIN_VALUE`
  - `Integer.MAX_VALUE`