THE BINARY HEAP
IMPLEMENTATION

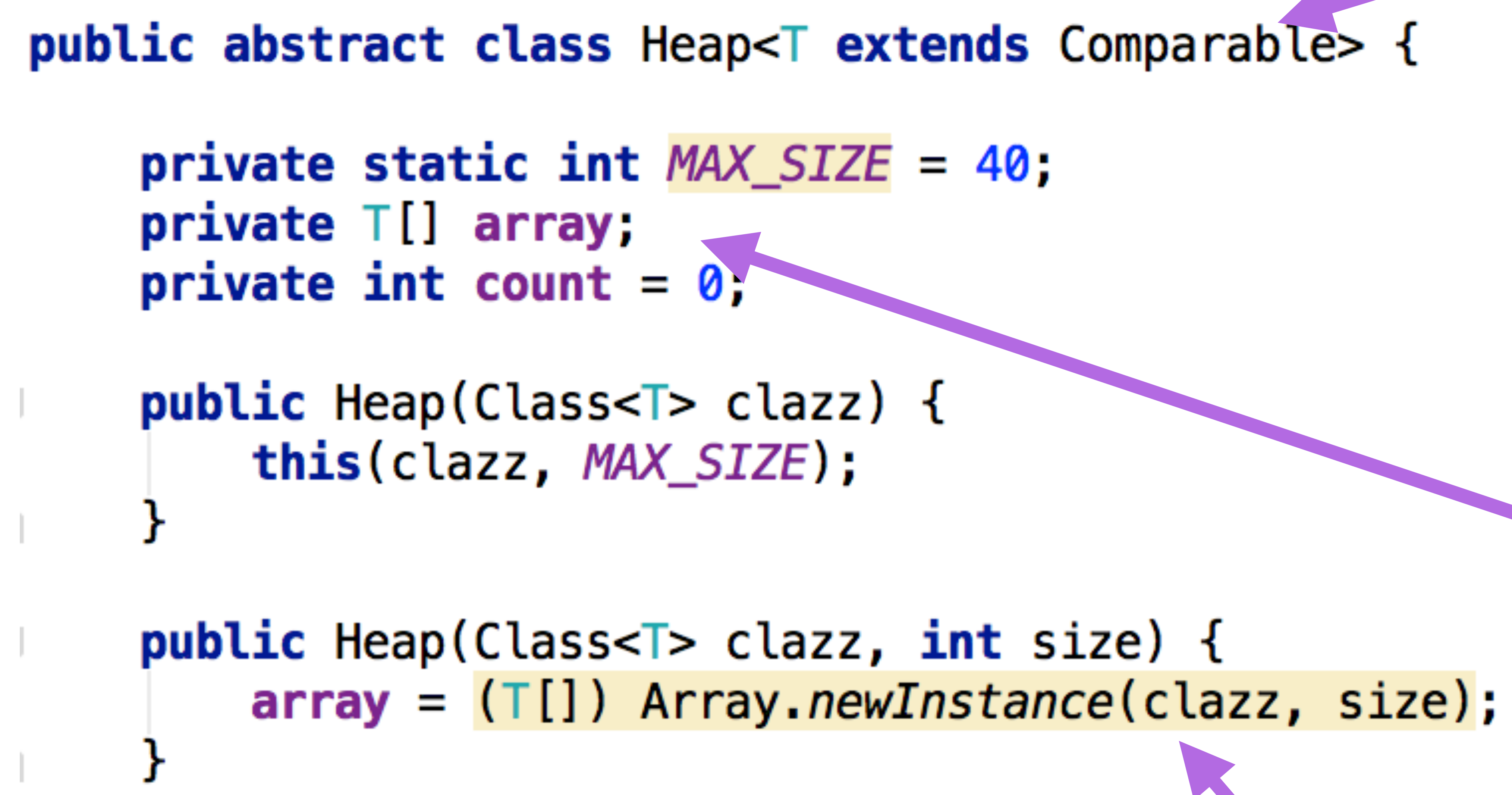
NOW LET'S SEE SOME CODE...

WE'LL START WITH THE HEAP BASE CLASS - THIS CONTAINS METHODS USED IN BOTH THE MIN HEAP AND THE MAX HEAP

HEAP - BASE CLASS

A GENERIC HEAP CAN HOLD DATA
OF OF ANY TYPE

NOTE THAT THE GENERIC TYPE HAS
TO EXTEND COMPARABLE - THIS IS HOW WE CHECK
FOR THE HIGHEST PRIORITY



```java
public abstract class Heap<T extends Comparable> {

    private static int MAX_SIZE = 40;
    private T[] array;
    private int count = 0;

    public Heap(Class<T> clazz) {
        this(clazz, MAX_SIZE);
    }

    public Heap(Class<T> clazz, int size) {
        array = (T[]) Array.newInstance(clazz, size);
    }
```

USE AN ARRAY TO STORE THE
HEAP ELEMENTS

THIS IS HOW YOU INSTANTIATE A
GENERIC ARRAY IN JAVA

BASE CLASS METHODS - GET LEFT CHILD INDEX

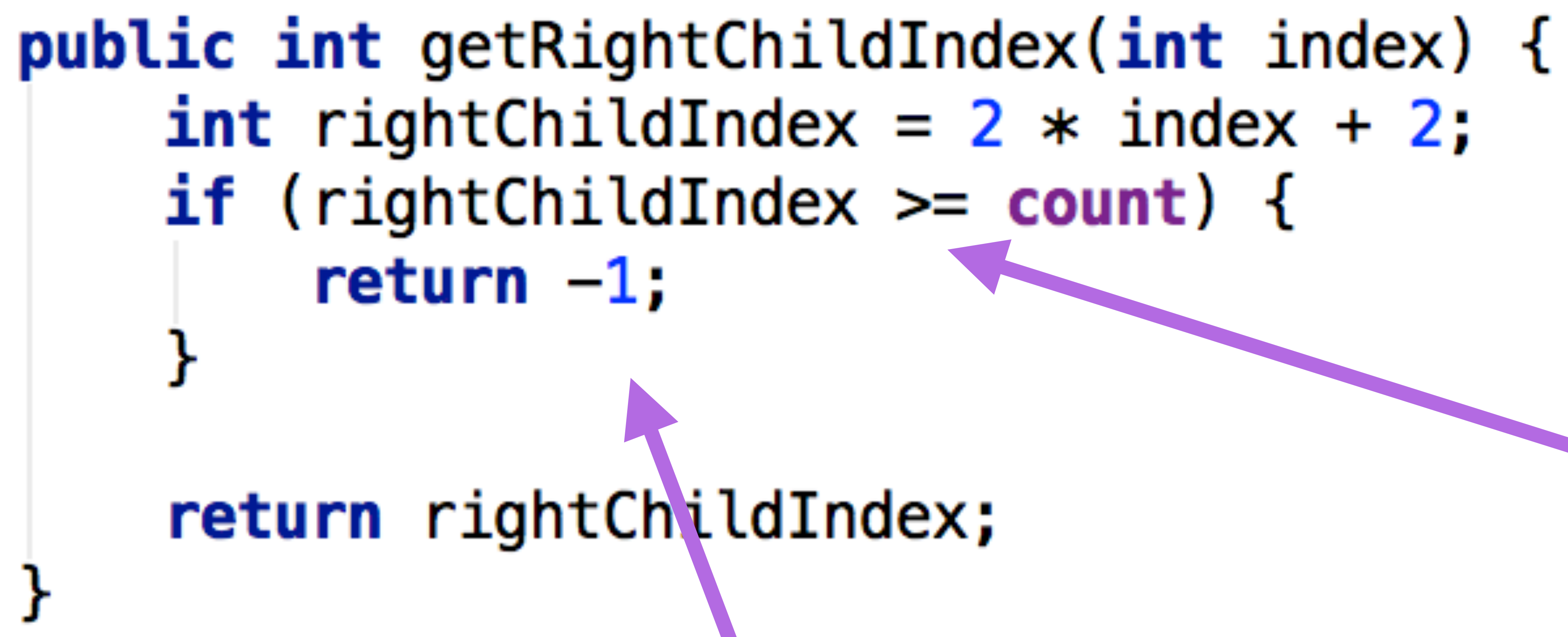
```java
public int getLeftChildIndex(int index) {
    int leftChildIndex = 2 * index + 1;
    if (leftChildIndex >= count) {
        return -1;
    }

    return leftChildIndex;
}
```

CALCULATE THE LEFT CHILD INDEX USING THE FORMULA

CHECK TO SEE IF A LEFT CHILD OF THIS NODE IS PRESENT. IF IT'S LESS THAN COUNT (THE NUMBER OF NODES) THEN IT'S A VALID LEFT CHILD

RETURN -1 IF A VALID LEFT CHILD WAS NOT FOUND

BASE CLASS METHODS - GET RIGHT CHILD INDEX



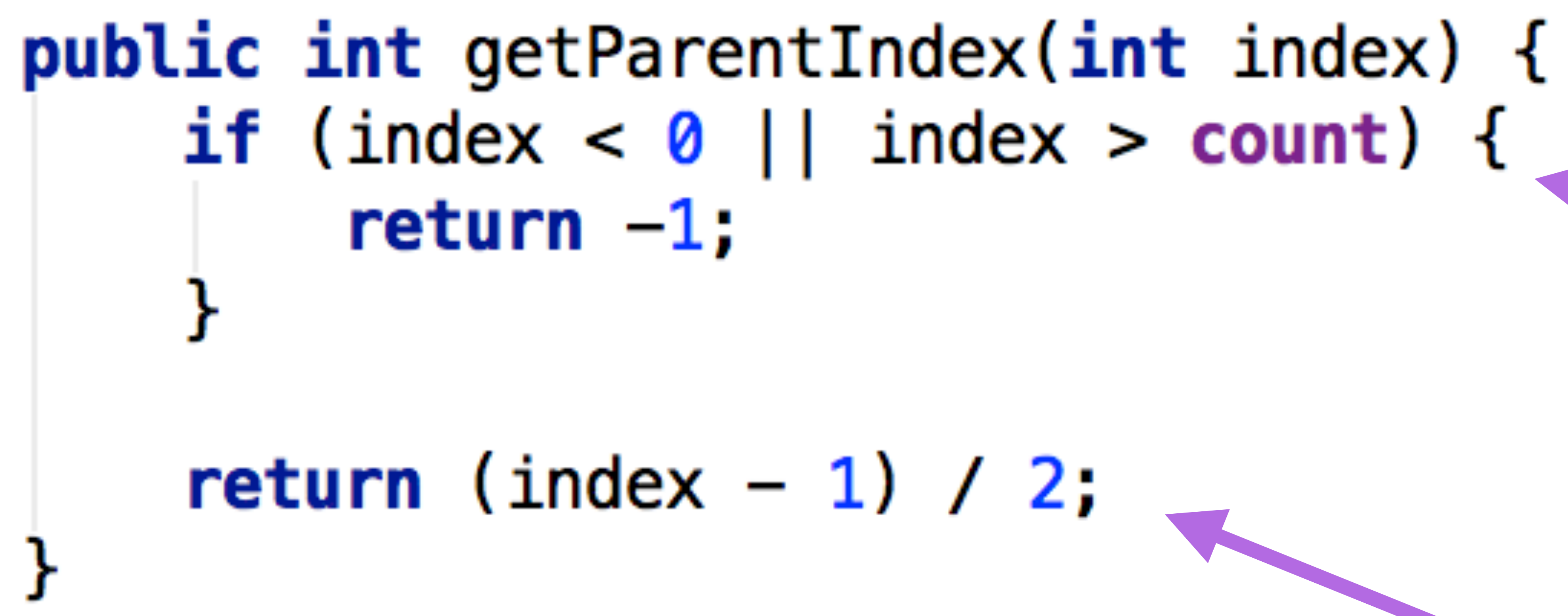
```java
public int getRightChildIndex(int index) {
    int rightChildIndex = 2 * index + 2;
    if (rightChildIndex >= count) {
        return -1;
    }

    return rightChildIndex;
}
```

CALCULATE THE RIGHT CHILD INDEX USING THE FORMULA

CHECK TO SEE IF A RIGHT CHILD OF THIS NODE IS PRESENT. IF IT'S LESS THAN COUNT (THE NUMBER OF NODES) THEN IT'S A VALID RIGHT CHILD

RETURN -1 IF A VALID RIGHT CHILD WAS NOT FOUND

BASE CLASS METHODS - GET PARENT INDEX



```java
public int getParentIndex(int index) {
    if (index < 0 || index > count) {
        return -1;
    }

    return (index - 1) / 2;
}
```

CHECK THAT THE INDEX IS NOT OUT OF RANGE

USE THE FORMULA TO GET THE PARENT INDEX

BASE CLASS METHODS - HELPERS

```java
protected void swap(int index1, int index2) {
    T tempValue = array[index1];
    array[index1] = array[index2];
    array[index2] = tempValue;
}

public int getCount() {
    return count;
}

public boolean isEmpty() {
    return count == 0;
}

public boolean isFull() {
    return count == array.length;
}

public T getElementAtIndex(int index) {
    return array[index];
}
```

SWAP 2 ELEMENTS IN THE HEAP ARRAY

WHOLE BUNCH OF OTHERS TO CHECK IF A HEAP IS FULL, EMPTY, GET THE NUMBER OF ELEMENTS ETC.