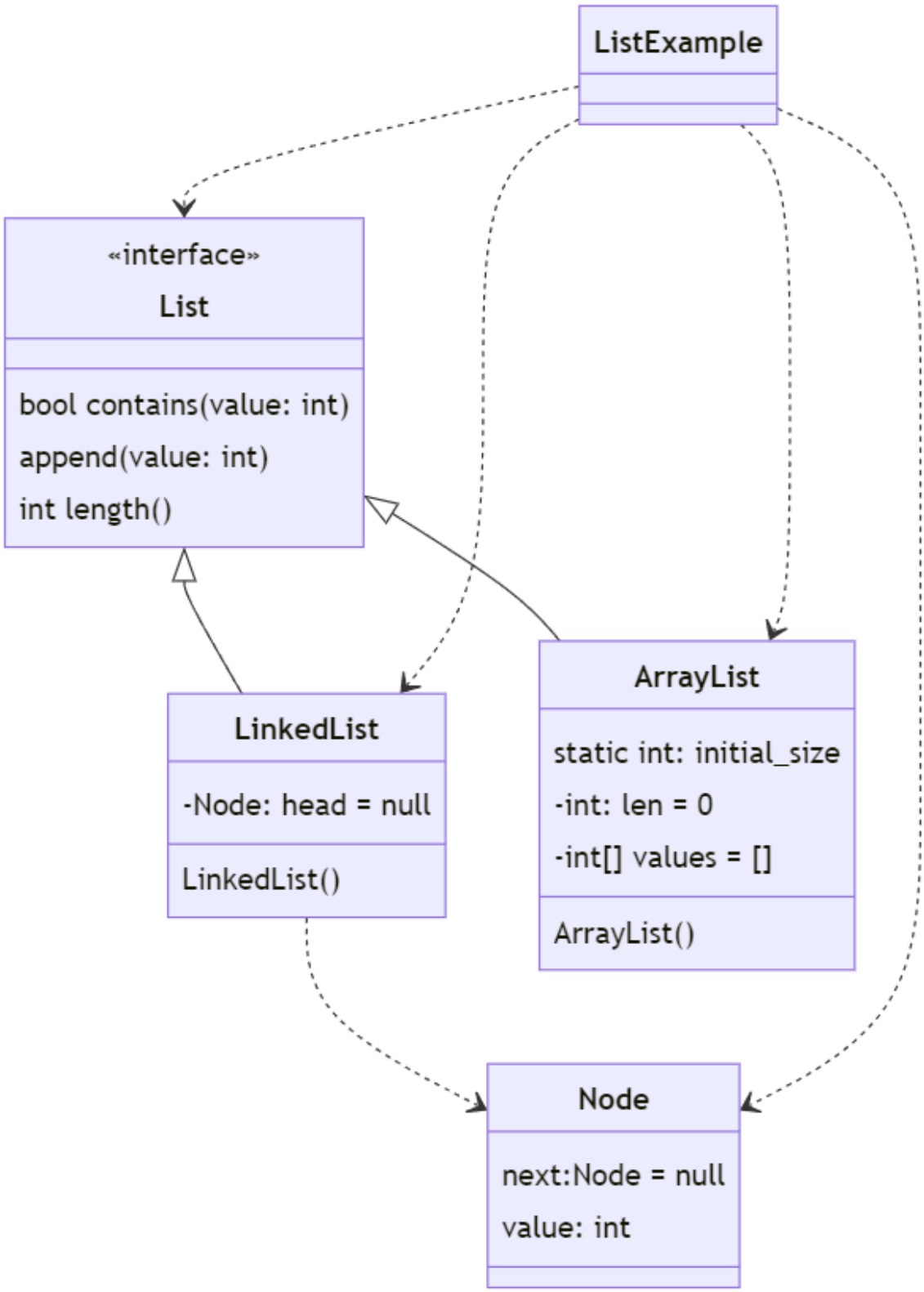

ListExample
«interface»
List
bool contains(value: int)
append(value: int)
int length()
LinkedList
-Node: head = null
LinkedList()
ArrayList
static int: initial_size
-int: len = 0
-int[] values = []
ArrayList()
Node
next:Node = null
value: int

Exercise: Hello Generics (2)

Im guessing that it would print:

Null

World

World

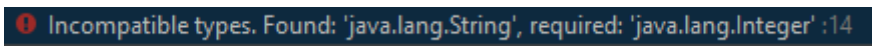
This is because it tries to print the most recent object before any objects are added, resulting in null, then will print out world twice since its getting the most recent object after the word “World” is added in.

Output:

```
C:\Users\GH\.jdks\openjdk-17.0.1-2\bin\java.exe
null
World
World

Process finished with exit code 0
```

When uncommenting line 14, it the IDE will give a warning saying that you cannot add a string into a Integer, as we declared mro to be able to hold `String`



Exercise: Odd Range Iterator (2)

I changed the Constructer such that if the start of the range is even, it will decrease the current by 1, and if its odd, reduce by 2, I have also done the same conditions to the end values This changes the range such that the start and end values will always be odd.

The reason why it reduces it by 1 or 2 because next() function when changed to increment by 2, seems to skip the first 2 values while incrementing it by 1 increase

```java
public RangeIterator(int start, int end) {
    this.current = start;
    this.end = end;
}
```

```java
public OddRangeIterator(int start, int end) {
    start = ChangeToOdd(start);
    end = ChangeToOdd(end);
    this.current = start;
    this.end = end;
}
```

```java
private int ChangeToOdd(int value)
{
    if(value % 2 == 0)
    {
        value--;
    }
    else
    {
        value -= 2;
    }
    return value;
}
```

I also changed the next function so that it increments by 2 instead of 1 such that it only prints out odd or even numbers depending on the starting number (which will always be odd due to the if statements in the constructor changing values to only be odd)

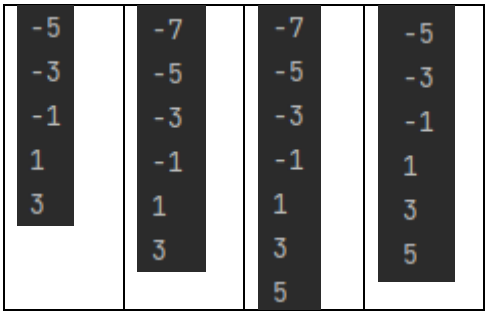
```java
public Integer next() {
    return current++;
}
```

```java
public Integer next() {
    return current += 2;
}
```

Range version is on the left, OddRange version is on the right

OutputTable for different ranges to make sure it works like in the example.

Even Odd	Odd Odd	Odd Even	Even Even
(-6,5)	(-7,5)	(-7, 6)	(-6,6)



Integer List Iterator

Array List implementation along with results compared to the linked list version

```
class ArrayList implements IntegerList
```

```
@Override
public Iterator<Integer> iterator() {
    return new ArrayList.ArrayListIterator(this);
}

private static class ArrayListIterator implements Iterator<Integer> {
    private ArrayList values;
    private int indexCurrent;
    private int indexEnd;

    public ArrayListIterator(ArrayList list) {
        values = list;
        indexCurrent = 0;
        indexEnd = list.len - 1;
    }

    @Override
    public boolean hasNext() { return indexCurrent != indexEnd; }

    @Override
    public Integer next() {
        int value = values.values[indexCurrent];
        indexCurrent++;
        return value;
    }
}
```

```
true
true
false
2
10
20
true
true
false
2
10
20
```

Result above the red line is the linked list while the bottom is the arraylist

Exercise: Generic Lists (2)

Append function of GenericLinkedList (efficient version since it tells us to use the efficient version of linkedlist in the first exercise)

```java
public void append(T value) {
    GenericNode<T> newNode = new GenericNode<T>(value);
    if (head == null) {
        head = newNode;
        tail = head;
    } else {
        tail.next = newNode;
        tail = tail.next;
    }
    len++;
}
```

Class for GenericArrayList (efficient version)

```java
class GenericArrayList<T> implements GenericList<T> {
    static int initialCapacity = 10;

    T[] values = (T[]) new Object[initialCapacity];
    int len = 0;

    public boolean contains(T value) {
        for (int i = 0; i < len; i++) {
            if (value == values[i]) {
                return true;
            }
        }
        return false;
    }

    public void append(T value) {
        if (len == values.length) {
            T[] newValues = (T[]) new Object[(len * 2) + 1];
            for (int i = 0; i < len; i++) {
                newValues[i] = values[i];
            }
            newValues[len] = value;
            values = newValues;
        } else {
            values[len] = value;
        }
        len++;
    }
```

```java
public int length() { return len; }

@Override
public Iterator<T> iterator() {
    return new GenericArrayListIterator<T>(this);
}

private static class GenericArrayListIterator<T> implements Iterator<T> {
    private GenericArrayList<T> values;
    private int indexCurrent = 0;
    private int indexEnd;

    public GenericArrayListIterator(GenericArrayList list) {
        values = list;
        indexEnd = list.len - 1;
    }

    @Override
    public boolean hasNext() { return indexCurrent != indexEnd; }

    @Override
    public T next() {
        T value = values.values[indexCurrent];
        indexCurrent++;
        return value;
    }
}
```

Output of the results (appended 20 times to cover all code in the append function for GenericArrayList)

Test Code (Types)	Result

```
static void testList(GenericList<Integer> list)
{
    for(int i = 0; i < 20; i++)
    {
        //String value = "String" + i;
        if(i == 3)
        {
            continue;
        }
        list.append(i);
    }
    System.out.println(list.contains(1));
    System.out.println(list.contains(2));
    System.out.println(list.contains(3));
    System.out.println(list.length());
      for(int j: list)
    {
        System.out.println(j);
    }*/
}

public static void main(String[] args) {
    testList(new GenericArrayList<Integer>());
    testList(new GenericLinkedList<Integer>());
}
```

```
true
true
false
19
true
true
false
19
```

Exercise: Marks Processing (2)

```
MarkRecord[studentId=0, moduleId=0, mark=33]
MarkRecord[studentId=0, moduleId=1, mark=46]
MarkRecord[studentId=1, moduleId=0, mark=93]
MarkRecord[studentId=1, moduleId=1, mark=20]
MarkRecord[studentId=2, moduleId=0, mark=41]
MarkRecord[studentId=2, moduleId=1, mark=9]
MarkRecord[studentId=3, moduleId=0, mark=12]
MarkRecord[studentId=3, moduleId=1, mark=83]
MarkRecord[studentId=4, moduleId=0, mark=43]
MarkRecord[studentId=4, moduleId=1, mark=89]

{0=lab2.GenericArrayList@5e9f23b4, 1=lab2.Gene

{0=lab2.GenericArrayList@378fd1ac, 1=lab2.Gene

Average of module 0 is : 44.4
Average of module 1 is : 49.4
Average of student 0 is : 39.5
Average of student 1 is : 56.5
Average of student 2 is : 25.0
Average of student 3 is : 47.5
Average of student 4 is : 66.0
```