Classwork 1T: main()

```java
import java.util.Scanner;

  public class UniqueWords {

      public static void main (String[] args) {
          Scanner input = new Scanner(System.in);    // C1
          System.out.println("Enter a sentence.");

          String[] words = input.nextLine().split(" "); // n
          int count = 0;          //C2
          boolean unique;
          for (int i = 0; i < words.length; i++) { //n
              unique = true;     //C3
              for (int j = 0; j < words.length; j++) { //n
                  if (i != j && words[i].equals(words[j])) { //C4
                      unique = false;      //C5
                      break;
                  }
              }
              if (unique) {count++;} //C6
          }
          System.out.println("There are " + count + " unique words in that 
sentence.");               //C7
          input.close();
      }

  }
```

$T(n) = C1 + n + C2 + n(C3 + nC4 + nC5) + C6 + C7 \implies O(n^2)$

IntList:

insertAt:

```
    public void insertAt(int toAdd, int index) {
        indexValidityCheck(index, 0, size + 1); //C1
        size++; //C2
        checkAndGrow(); //C3n
        shiftRight(index); //C4n
        items[index] = toAdd; //C5
    }

private void checkAndGrow () {
        if (size < items.length) {  //C1
            return;
        }
        int[] newItems = new int[items.length * 2]; //C2
        for (int i = 0; i < items.length; i++) {   //nC3
            newItems[i] = items[i];     //nC4
        }
        items = newItems; //C5
    }

    private void indexValidityCheck (int index, int lower, int upper) {
        if (index < lower || index >= upper) { //C1
            throw new IndexOutOfBoundsException(); // C2
        }
    }

    private void shiftRight (int index) {
        for (int i = size; i > index; i--) { //nC1
            items[i] = items[i-1]; //nC2
        }
    }
```

checkAndGrow:

$$T(n) = C1 + C2 + nC3 + nC4 + C5 \implies O(n)$$

shiftRight:

$$T(n) = nC1 + nC2 \implies O(n)$$

InsertAt then is:

$$T(n) = C1 + C2 + n(C3 + C4) + C5 \implies O(n)$$

getAt:

```java
    public int getAt(int index) {
        indexValidityCheck(index, 0, size);
        return items[index];
    }
```

$T(n) = C1 \implies O(1)$

IntLinkedList:

Prepend:

```java
    public void prepend (int toAdd) {
        Node currentHead = head; //C1
        head = new Node(toAdd); // C2
        head.next = currentHead; //C3
        size++; //C4
    }
```

$T(n) = C1 + C2 + C3 + C4 \implies O(1)$

getIteratorAt:

```java
     public Iterator getIteratorAt (int index) {
         if (index > size || index < 0) { //C1
             throw new IllegalArgumentException();
         }
         Iterator it = new Iterator ();
         while (index > 0) { //nC5
             it.next(); // n(C2 + C3)
             index--; // n(C4)
         }
         return it;
     }

         public void next () {
             if (current == null) {return;} //C2
             current = current.next; // C3
         }
```

$T(n) = C1 + n(C2 + C3 + C4 + C5) \implies O(n)$