Arizona State University

School of Computing, Informatics, and Decision Systems Engineering

CSE 205 Lecture Notes

Ryan Dougherty
redoughe@asu.edu

To my friends and family

Table of Contents

Introduction

Welcome to these lecture notes for CSE205! This guide will follow up on the first lecture notes (for CSE110), and get more into the "algorithms and computation" side of programming. However, all examples will be in Java, to remain consistent between the two sets of notes. As we have said before, programming is becoming more of an important tool in the last few decades. Understanding of these technological shifts, therefore, is imperative.

The topics are split up according to the Table of Contents above, and their corresponding page numbers. At the end of each chapter are written and programming exercises. The written exercises are for testing your knowledge of the material, and the programming ones are to see if you can write programs using the material from that chapter.

If there are any questions/errors/comments that you want to send regarding the material, send an email to: Ryan.Dougherty [at] asu.edu.

Useful Bits & Pieces

In this section we will give a few "miscellaneous" bits of information that are not big enough topics to warrant a whole section, but are nevertheless important. Knowledge from the first set of lecture notes is assumed.

2.1 String.split

There is a useful method in the `String` class called `split(String)` - it returns a `String` array (`String[]`) that consists of the `String` being called on without all instances of the passed in `String`. The passed in `String` is often called the "delimiter." Here we give some examples:

```
1 String original = "I am going to learn Java!";
2 String[] allWords = original.split(" ");
3 // allWords = {"I", "am", "going", "to", "learn", Java!"};
4 String[] words2 = original.split("a");
5 // allWords = {"I ", "m going to le", "rn J", "v", "!"};
```

2.2 ArrayList

Dealing with ordinary arrays in Java (or any programming language) is quite a hassle. Therefore, there is a class in the `java.util` package called `ArrayList`, which is an object wrapper for an array. It is a *templated* type, which means it has angle brackets to denote what type is allowed in the `ArrayList` (this type must be an `Object`, not a primitive type). Here are some examples of its use:

```
1  ArrayList<Integer> ints = new ArrayList<Integer>(); // example
     of initialization - must be Integer, not int
2  ints.add(0);
3  ints.add(2);
4  ints.add(0); // can add 0 again
5  ints.add(1, 2); // add 2 at index 1
6  ints.remove(3); // remove element at index 3
7  int value = ints.get(1); // gets element at index 1
8  ints.set(0, 5); // sets index 0 to 5
9  if (ints.isEmpty()) {
10       System.out.println("The array is empty!");
11 }
12 // for loop over the array
13 for (int i=0; i<ints.size(); i++) {
```

```
14        System.out.println(ints.get(i));
15    }
16    ints.clear(); // ints has no elements
```

There is another class called `Vector`, also in the same package, that provides the same interface as `ArrayList`, but is thread-safe, whereas `ArrayList` is not. This means `Vector` is useful in multithreaded environments, as we will see in a later chapter.