Name:

YOU MAY READ THIS PAGE BEFORE THE EXAM BEGINS

1. You have 75 minutes to create and submit the 2 programs in this exam.
2. When instructed to start, download and extract the following IntelliJ Project under the `~/workspace` folder if you do not have it there already.

   `https://www.cs.umb.edu/~siyer/teaching/dsaj.zip`

3. Open the project in IntelliJ. To create a program, right-click on the `src` folder under `dsaj` in the top-left window and then select the *New* → *File* menu. Enter the name of the program in the pop-up window. Note that the name is case-sensitive and must match the suggested name exactly.
4. You may use the text, your notes, your code from the assignments, and the code on the CS210 course website. No form of communication is permitted (eg, talking, texting, etc.) during the exam, except with the course staff.
5. Submit your programs (`.java` files) on Gradescope under the assignment named `Sample Programming Exam 1`.
6. Return this exam sheet to the course staff with your name written at the top. Failing to do so will void your exam submission on Gradescope.
7. You are *not* allowed to leave the exam hall before the official end time even if you are done early.
8. Your programs will be graded based on correctness, clarity, and efficiency.
9. Discussing the exam contents with anyone who has not taken the exam is a violation of the academic honesty code.

DO NOT READ FURTHER UNTIL SO INSTRUCTED

Problem 1. (7 Points) Implement the function `private static int sum(int[][] a)` in `MatrixSum.java` such that it returns the sum of the elements in `a`.

MatrixSum.java

```java
import stdlib.StdArrayIO;
import stdlib.StdOut;

public class MatrixSum {
    private static int sum(int[][] a) {
        ...
    }

    // Entry point.
    public static void main(String[] args) {
        StdOut.println(sum(StdArrayIO.readInt2D()));
    }
}
```

>_ ~/workspace/dsaj

```
$ javac -d out src/MatrixSum.java
$ java MatrixSum
2 3
1 2 3 4 5 6
21
```

Problem 2. (18 Points) Implement a comparable, iterable data type called `Sentence` that represents a sentence, which is a sequence of words. The data type must the following API:

Sentence implements Comparable<Sentence>, Iterable<String>

`Sentence(String s)`	constructs a `Sentence` object from the sentence `s`
`int charCount()`	returns the number of characters in this sentence
`int wordCount()`	returns the number of words in this sentence
`boolean equals()`	returns `true` if this sentence is the same as `other`, and `false` otherwise
`String toString()`	returns a string representation of this sentence in `"<word count>:<sentence>"` format; for example, the sentence "she sells sea shells" is represented as `"4:she sells sea shells"`
`int compareTo(Sentence other)`	returns a comparison of this and `other` sentence based on their character counts
`static Comparator<Sentence> wordCountOrder()`	returns a comparator for comparing sentences based on their word counts
`Iterator<String> iterator()`	returns an interator for iterating over the words in this sentence in *reverse* order

Sentence.java

```java
import java.util.Comparator;
import java.util.Iterator;
import stdlib.StdOut;

public class Sentence implements Comparable<Sentence>, Iterable<String> {
    private String s; // the sentence
    private String[] words; // words in the sentence

    public Sentence(String s) { ... }

    public int charCount() { ... }

    public int wordCount() { ... }

    public boolean equals(Object other) {
        if (other == null) {
            return false;
        }
        if (other == this) {
            return true;
        }
        if (other.getClass() != this.getClass()) {
            return false;
        }
        ...
    }
    public String toString() { ... }

    public int compareTo(Sentence other) { ... }

    public static Comparator<Sentence> wordCountOrder() { ... }

    public Iterator<String> iterator() { ... }

    private static class WordCountOrder implements Comparator<Sentence> {
        public int compare(Sentence s1, Sentence s2) { ... }
    }

    private class ReverseWordIterator implements Iterator<String> {
        private int i; // index of the current letter

        public ReverseWordIterator() { ... }

        public boolean hasNext() { ... }

        public String next() { ... }
    }

    public static void main(String[] args) {
        Sentence s1 = new Sentence("abc def ghi jkl mno");
        Sentence s2 = new Sentence("abcdefg hijklmn opqrst");
        Sentence s3 = new Sentence("abc def ghi jkl mno");
        StdOut.println(s1);
        StdOut.println(s2);
        StdOut.println(s3);
        StdOut.println(s1.charCount());
        StdOut.println(s1.equals(s3));
        StdOut.println(s1.compareTo(s2));
        StdOut.println(Sentence.wordCountOrder().compare(s1, s2));
        for (String word : s3) {
            StdOut.print(word + " ");
        }
        StdOut.println();
    }
}
```

```
>_ ~/workspace/dsaj
$ javac -d out src/Sentence.java
$ java Sentence
5:abc def ghi jkl mno
3:abcdefg hijklmn opqrst
5:abc def ghi jkl mno
19
true
-3
2
mno jkl ghi def abc
```

Hint: use `s.split("\\s+")` to split a string into an array of tokens. For example, if `s = "she sells sea shells"`, then `s.split("\\s+")` returns the array `{"she", "sells", "sea", "shells"}`.

Files to Submit

1. `MatrixSum.java`
2. `Sentence.java`

Solutions

MatrixSum.java

```java
import stdlib.StdArrayIO;
import stdlib.StdOut;

public class MatrixSum {
    // Returns the sum of the elements in a.
    private static int sum(int[][] a) {
        int sum = 0;
        for (int i = 0; i < a.length; i++) {
            for (int j = 0; j < a[0].length; j++) {
                sum += a[i][j];
            }
        }
        return sum;
    }

    // Entry point.
    public static void main(String[] args) {
        StdOut.println(sum(StdArrayIO.readInt2D()));
    }
}
```

Sentence.java

```java
import java.util.Comparator;
import java.util.Iterator;

import stdlib.StdOut;

// This comparable, iterable data type that represents a sentence, which is a sequence of words.
public class Sentence implements Comparable<Sentence>, Iterable<String> {
    private String s; // the sentence
    private String[] words; // words in the sentence

    // Constructs a Sentence object from the sentence s.
    public Sentence(String s) {
        this.s = s;
        words = s.split("\\s+");
    }

    // Returns the number characters in this sentence.
    public int charCount() {
        return s.length();
    }

    // Returns the number of words in this sentence.
    public int wordCount() {
        return words.length;
    }

    // Returns true if this sentence is the same as other, and false otherwise.
    public boolean equals(Object other) {
        if (other == null) {
            return false;
        }
        if (other == this) {
            return true;
        }
        if (other.getClass() != this.getClass()) {
            return false;
        }
        Sentence a = (Sentence) this, b = (Sentence) other;
        return a.s.equals(b.s);
    }

    // Returns a string representation of this sentence.
    public String toString() {
        return wordCount() + ":" + s;
    }

    // Returns a comparison of this and other sentence based on their lengths (ie, character
    // counts).
    public int compareTo(Sentence other) {
        return this.charCount() - other.charCount();
    }

    // Returns a comparator for comparing sentences based on their word count.
    public static Comparator<Sentence> wordCountOrder() {
        return new WordCountOrder();
    }

    // Returns an iterator for iterating over this sentence in reverse order.
    public Iterator<String> iterator() {
        return new ReverseWordIterator();
    }

    // A comparator for comparing sentences based on their word counts.
    private static class WordCountOrder implements Comparator<Sentence> {
        // Returns a comparison of sentences s1 and s2 based on their word count.
        public int compare(Sentence s1, Sentence s2) {
            return s1.wordCount() - s2.wordCount();
        }
    }

    // An iterator for iterating over a sentence in reverse order.
    private class ReverseWordIterator implements Iterator<String> {
        private int i; // index of current letter

        // Constructs an iterator.
        public ReverseWordIterator() {
            i = wordCount() - 1;
        }

        // Returns true if there are more words in the sentence, and false otherwise.
        public boolean hasNext() {
            return i >= 0;
        }

        // Returns the next word in the sentence.
        public String next() {
            return words[i--];
        }
    }

    // Unit tests the data type [DO NOT EDIT].
    public static void main(String[] args) {
        Sentence s1 = new Sentence("abc def ghi jkl mno");
        Sentence s2 = new Sentence("abcdefg hijklmn opqrst");
        Sentence s3 = new Sentence("abc def ghi jkl mno");
        StdOut.println(s1);
        StdOut.println(s2);
        StdOut.println(s3);
        StdOut.println(s1.charCount());
        StdOut.println(s1.equals(s3));
        StdOut.println(s1.compareTo(s2));
        StdOut.println(Sentence.wordCountOrder().compare(s1, s2));
        for (String word : s3) {
            StdOut.print(word + " ");
        }
        StdOut.println();
    }
}
```