Duration: **50 minutes**
Aids Allowed: **The Java API: An Introduction for Students**

**Student Number:** |_|_|_|_|_|_|_|_|_|

**Last Name:** ______________________________

**First Name:** ______________________________

**Lecture Section:** ____________ **Instructor: Danny Heap**

---

*Do **not** turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above,
and read the instructions below *carefully*.)

---

This midterm test consists of 2 questions on 4 pages (including this one), printed on one side of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete.* If you need more space, use the reverse side of the page and *indicate clearly the part of your work that should be marked.*

MARKING GUIDE

# 1: ______/25

# 2: ______/ 7

TOTAL: ______/32

*Good Luck!*

## Question 1. [25 MARKS]

Note: read parts (a) and (b) completely before answering.

### Part (a) [9 MARKS]

Consider the following partially-implemented class, `Advertisement`. Write the two missing methods `setCharacterRate`, `addText`, and the `constructor`.

```
/** An Advertisement holds text in currentText, maximumText is the
 *  maximum number of characters allowed in an Advertisement.  All
 *  Advertisements are charged a standard characterRate.
 */
public class Advertisement {

  private static int characterRate; // cents-per-character charged on every Advertisement
  private int maximumText; // maximum number of characters in this Advertisement
  private String currentText; // text currently in this Advertisement

  // [3 marks]
  // setCharacterRate: set the characterRate to r for every Advertisement.
  public static void setCharacterRate(int r) {
    characterRate= r;
  }

  // getCharactersRemaining: return the number of characters that can
  // be added to the current text without exceeding maximumText characters.
  public int getCharactersRemaining() {
    return maximumText - currentText.length();
  }

  // [3 marks]
  // constructor: create an Advertisement with room for up to
  // n characters.
  public Advertisement(int n) {
    maximumText= n;
  }

  // [3 marks]
  // addText: add text t to the end of currentText.
  // There is no need to check whether you have exceeded
  // the maximum number of characters allowed.
  public void addText(String t) {
    if (currentText == null) {currentText= "";}
    currentText= currentText + t;
  }
}
```

**Part (b)** [6 MARKS]

Follow the instructions given in the comments for the class `ClassifiedAds` below. This class uses `Advertisement` from part (a).

```
/** ClassifiedAds models a collection of classified ads, and uses
 *  the class Advertisement.
 */
public class ClassifiedAds {
  public static void main(String[] args) {
    // [1 mark] Write a line of Java to set the
    // charge-per-character of every Advertisement to 10
    Advertisement.setCharacterRate(10);

    Advertisement sale= new Advertisement(75);
    String s= "Going out of business!  Everything must go!\n";

    // [1 mark] Write a line of Java that adds s to the text of sale.
    sale.addText(s);

    String moreText= "All socks 50% off, shoes even less!\n";

    // [4 marks] write a code fragment (several lines of code) that
    // checks to see whether there is enough room to add moreText
    // to the text of sale.  If there is, add moreText, and
    // print "Characters remaining: X" where X is the number of
    // characters remaining after you have added moreText.  Otherwise,
    // print "Not enough room." and do not add moreText.
    if (sale.getCharactersRemaining() >= moreText.length()) {
      sale.addText(moreText);
      System.out.println("Characters remaining: " +
                         sale.getCharactersRemaining());
    }
    else {
      System.out.println("Not enough room.");
    }
  }
}
```

**Part (c)** [1 MARK]

Write the output of `ClassifiedAds`. Assume that `s.length()` is 44 and `moreText.length()` is 36.

```
Not enough room.
```

**Part (d)** [2 MARKS]

Write the name of each **instance** variable in `Advertisement` and `ClassifiedAds`.

`maximumText, currentText`

**Part (e)** [3 MARKS]

Write the name of each **local** variable in `Advertisement` and `ClassifiedAds`.

`sale, s, moreText`

**Part (f)** [3 MARKS]

Write the name of each **parameter** in `Advertisement` and `ClassifiedAds`.

`r, n, t, args`

**Part (g)** [1 MARK]

Write the name of each **static** variable in `Advertisement` and `ClassifiedAds`.

`characterRate`

## Question 2. [7 MARKS]

**Part (a)** [2 MARKS]

The class `String` is a standard part of Java, and a `String` is immutable (unchangeable). In light of this, explain why (or why not) the following code fragment is possible in Java:

```
String s= "hi";
String t= "there";
s= t;
```

This fragment first assigns the address of `String "hi"` to variable `s`, then assigns the address of `String "there"` to variable `t`, and finally assigns the address contained in `t` to `s`. No `String` objects are changed, simply the address stored in variable `s`.

**Part (b)** [5 MARKS]

Complete the method `chopString` below.

```
/** chopString removes the last s.length()/2 characters from the
  * end of s, adds them to the beginning (in the same order), and
  * returns the result.  You may assume that s is not null.
 */
public static String chopString(String s) {
  int mid= s.length() - s.length()/2;
  return s.substring(mid) + s.substring(0,mid);
}
```