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

Student Number: ___________

Last Name: ___________________________

First Name: ___________________________

Lecture Section: __________ **Instructor:** ___________________

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 3 questions on 6 pages (including this one), printed on **both** sides of the paper. *When you receive the signal to start, please make sure that your copy of the test is complete.* Answer each question directly on the test paper, in the space provided. If you need more space for one of your solutions, use the final page and *indicate clearly the part of your work that should be marked.*

Be aware that concise, well thought-out answers will be rewarded over long rambling ones. Also, unreadable answers will be given zero (0) so write legibly. In your answers, you may use without justification any facts given during the course, *as long as you state them clearly*. You must justify any other facts needed for your solution.

General Hint: We were careful to leave ample space on the test paper to answer each question. If you think that you need much more room that what is available, take the time to stop and think about what you are doing instead: there is probably a shorter, easier solution. Also, remember that hints are just hints: you are not required to follow them if you can think of a different solution.

MARKING GUIDE

BONUS MARKS:	____/ 4
# 1:	____/ 6
# 2:	____/14
# 3:	____/ 8
TOTAL:	____/28

Good Luck!

Question 1. [6 marks]

Write the output of the following loops in the space provided. Recall that the % operator is the modulo operator: **a % b** returns the remainder from dividing **a** by **b**. For example, 5 % 3 = 2, 5 %7 = 5, 27 % 3 = 0, and so on.

Part (a) [3 marks]

```
String someString = "Bread";
for(int i = 0; i != someString.length(); i++){
        System.out.println(someString.charAt(i/2));
}
```

Part (b) [3 marks]

```
String someString = "lever";
for(int i = 1; i <= someString.length(); i++){
        System.out.println(someString.charAt((i*4)%5));
}
```

Question 2. [14 MARKS]

The code below is similar to some of the code we talked about in class in that there is a **CourseList** class which contains an array of type **Course**. However, in this version there is also an array of type `int` in the class **CourseList** which stores the number of courses offered by each department, where departments are numbered from 0 to NUM_DEPARTMENTS - 1. Each department has a three letter prefix for its course names which is different from the three letter prefix for every other department. At UofT for example, computer science courses have the prefix **CSC** while biochemistry courses have the prefix **BCH**.

```
class Course{
    private String name;

    public  Course(String cname) { name = cname; }
    public boolean isOfType(String FirstThreeLetters) {
        return ((name.substring(0,3)).equals(FirstThreeLetters));
    }
  }

class CourseList{
    public static final int MAX_COURSES = 5;
    public static final int NUM_DEPARTMENTS = 3;

    private static int [] numCoursesOfDept = new int[NUM_DEPARTMENTS];
    private static int numCourses = 0;

    private static Course [] courseArray = new Course[MAX_COURSES];

    public static Course addCourse(String name, int Dept){
        if (numCourses >= MAX_COURSES) {
            System.out.println("Too many Courses. ");
            return null;
        }
        courseArray[numCourses] = new Course(name);
        numCoursesOfDept[Dept]++;
        numCourses++;
        return courseArray[numCourses-1];
    }







}
```

Part (a) [9 MARKS]

Write a method for the class **CourseList** called **getList()** which has a parameter of type `int` (the number of the department), a parameter of type **String** (the three letter prefix) and returns an array of type **Course** containing references to all the courses offered by that department. The length of the array must be exactly the number of courses offered by that department.

You may not add any instance or class variables to either class, nor may you add any methods to the class **Course**. Note that there is a method **isOfType()** in the class **Course** which could prove useful.

Part (b) [5 MARKS]

Describe how you would test the method **addCourse()**. You may write lines of code, but you must describe why those lines of code are appropriate for testing the method.

Question 3. [8 MARKS]

An electronic combination lock requires the user to input 3 integers, each of which is two digits and in the range 00 – 99. Here is a picture of the lock's display after a user has entered three integers:

Enter guess:	43	22	07

The lock has a combination of 3 integers each in the range 00 – 99 which is unkown to the user. This combination is set by the creator of the lock, and the lock is locked when it is first created. Once a lock is created, there is no direct access to the combination. The three integers displayed above are guesses made by a user trying to unlock the lock. Each of the three integers can be guessed independently of the others — thus they can be changed in any order. When each guessed integer is equal to the hidden combination, the lock is unlocked. A lock can be reset, meaning the three displayed guesses are set to 0 (the combination doesn't change, and the lock is locked) Every lock is either locked or unlocked and the lock can be asked which of these two states it is in.

Design a class to model such electronic combination locks. In your design, do not write any code. Specify the names and responsibilities of instance and class variables and of all methods in your class along with the parameters and return values of all methods. You should design your class so that someone could write a `main` method which could call your methods and create an interactive simulation of the electronic lock; however, you do not need to worry about details such as writing prompts for the user. A programmer should be able to look at your design and write the simulation *even though you haven't actually implemented your class.*

Bonus. [4 MARKS]

Name and briefly describe the stages of software development in the waterfall model. Write in point form and give no more than 3 points per stage.

Total Marks = 28