Mid-Term Test — Section L0101 (Afternoon)

Duration: *50 minutes* (3:10pm–4:00pm)
Aids Allowed: NONE (not even calculators)

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

Last Name: ____________________

First Name: ____________________

Tutorial Section: (circle one)	Iris MP 203	Theophanis MP 118	Wei SS 2111

---

*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 term test consists of 3 questions on 6 pages (including this one). *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, and *use the reverse side of the pages for rough work.* (If you need more space for one of your solutions, use the reverse side of the page and indicate **clearly** which part of your work 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 any facts given during the course, without proof or justification, *as long as you state them clearly*. You must justify any other facts needed for your solutions.

General Hint: We were careful to leave ample space on the test paper to answer each question, so if you find yourself using much more room than what is available, you're probably missing something. Also, remember that hints are just hints: you are not required to follow them!

1: ______/15
2: ______/20
3: ______/20
TOTAL: ______/55

Good Luck!

Question 1. [15 MARKS]

Part (a) [2 MARKS]

Write your student number **legibly** at the top of every page of this test, *except page 1.*

Part (b) [6 MARKS]

For each statement below, check the appropriate box.

TRUE	FALSE	
☐	☐	If I have a class `A` and I want to make it accessible to classes in another package, I can achieve this by giving it `protected` accessibility.
☐	☐	A Java `interface` can optionally specify method bodies, as long as they do not refer to any instance variables and only make calls to other methods in the interface.
☐	☐	It is possible to use class `java.util.Vector` without having an import statement.

Part (c) [3 MARKS]

State the two elements that must be specified to define an Abstract Data Type.

Part (d) [4 MARKS]

Give an example of how one would create a new type of `Exception`.

Question 2. [20 MARKS]

Consider the following Java interface.

```
// A SortedList stores Strings in non-decreasing sorted order.
public interface SortedList {

    // Inserts the given String at the correct position in this list.
    public void insert(String data);

} // interface SortedList
```

Part (a) [5 MARKS]

Fill-in a representation invariant for class VectorSortedList given below.

```
import java.util.Vector;

abstract public class VectorSortedList implements SortedList {

    private Vector items = new Vector();

    ////  WRITE YOUR REPRESENTATION INVARIANT HERE.  ////









    abstract public void insert(String data);

} // class VectorSortedList
```

Part (b) [15 MARKS]

Implement the insert method for class LinkedSortedList given on the next page, and write a complete method specification for the method. Recall that class String implements the interface Comparable, so that two Strings can be compared using the .compareTo() method. You do not have to write internal comments for your code, but you can get part marks even if your code is incorrect as long as you have comments that clearly show you have the right idea.

Question 2. (CONTINUED)

Part (b) (CONTINUED)

```
class Node {
    public String data;
    public Node link;
    public Node(String d) { data = d; }
} // class Node

public class LinkedSortedList implements SortedList {
    private Node head;

    ////  WRITE YOUR METHOD SPECIFICATION HERE, THEN IMPLEMENT THE METHOD.  ////






    public void insert(String data) {










    } // void insert(String)
} // class LinkedSortedList
```

Question 3. [20 MARKS]

Consider the following code.

```
class VWCar {
    int numPersons = 0;

    int getNumPersons() { return numPersons; }
    void stuff(int n) { numPersons += n; }
    void unstuff(int n) { numPersons -= n; }
    void transfer(VWCar c, int n) {
        c.stuff(n);
        unstuff(n);
    }
} // class VWCar

class Jetta extends VWCar {
    void stuff(int n) { numPersons = Math.min(4, numPersons + n); }
}

class PassatWagon extends VWCar {
    void stuff(int n) { numPersons = Math.min(6, numPersons + n); }
}

public class Driver {
    public static void main(String[] args) {
        VWCar c1 = new PassatWagon();
        VWCar c2 = new Jetta();

        c1.stuff(15);
        c2.stuff(2 * c1.getNumPersons());
        System.out.println("1: " + c1.getNumPersons()
            + " " + c2.getNumPersons());

        VWCar c3 = new VWCar();
        c3.stuff(20);
        c1.unstuff(2);
        c3.transfer(c1, 1);
        System.out.println("2: " + c1.getNumPersons()
            + " " + c3.getNumPersons());

        c1 = c2;
        c3 = c1;
        c1.transfer(c3, 4);
        System.out.println("3: " + c1.getNumPersons()
            + " " + c3.getNumPersons());
    }
} // class Driver
```

[Continued on the next page...]

Question 3. (CONTINUED)

Part (a) [10 MARKS]

Write the output produced by running this code.

Part (b) [10 MARKS]

Draw the Call Stack at the point during the execution of the code just *before* line 1 of method `transfer` is executed for the **second** time.