CSC 150H1 F October Midterm 2005
L0101
Duration — 50 minutes

Student Number: |__|__|__|__|__|__|__|__|__|

Family Name: ______________________ **Given Name:** ______________________

*Do **not** turn this page until you have received the signal to start.*
(Please fill out the identification section above,
and read the instructions below.) *Good Luck!*

This midterm consists of 2 questions on 7 pages (including this one). *When you receive the signal to start, please make sure that your copy is complete.*

For 1 bonus mark write your student number at the bottom of pages 2-7.

If you do any rough work, indicate clearly what you want marked.

# 1:	______/34
# 2:	______/ 8
TOTAL:	______/42

Question 1. [34 marks]

Consider the following version of the class `Rational` from lecture. The writer noticed, just as we did, that updating `n` requires the original value of `d`. Unfortunately there is still an error, when an object is added to itself.

```
class Rational implements Addable {

  private int n;
  private int d;

  public Rational(int n, int d) {
    this.n = n;
    this.d = d;
  }

  // Requires r to be Rational
  public void add(Addable r) {
    Rational r1 = (Rational) r;
    // Remember original d for when we update n
    int d = this.d;
    this.d = d * r1.d;
    n = n * r1.d + d * r1.n;
  }

  public boolean equals(Object o) {
    return o instanceof Rational
      && n * ((Rational) o).d == d * ((Rational) o).n;
  }

  public String toString() {
    return n + ‘‘/’’ + d;
  }

}
```

Part (a) [8 MARKS]

Show the state of memory when the following code is executed, just before `add` returns (in other words, after `add`'s last line is executed but it and its local variables still exist).

```
class C {
  public static void main(String[] args) {
    Rational a = new Rational(2, 3);
    Rational b = a;
    a.add(b);
  }
}
```

Part (b) [2 MARKS]

What would then be returned by `a.toString()`?

Part (c) [8 MARKS]

The writer of `Rational` finds out that many users need a rational equal to 1 and decides to save them memory by providing a method called `one`. This method keeps returning the same object when called, except it starts using a new one each time the current one is no longer 1 (because someone called `add` on it). Even though this method is not actually a good idea, write it.

Part (d) [6 MARKS]

Suppose now that Rational gets fixed and works properly.

Recall the following code from lecture:

```
class Add {
  // Add all the elements of l to its first element
  //  and return that element.
  // Requires l to be non-empty and contain only Addables.
  public static Addable addUp(LinkedList l) {
    Addable sum = (Addable)(l.get(0));
    Iterator i = l.iterator();
    i.next(); // skip first element since already accounted for
    while (i.hasNext()) {
      sum.add((Addable)(i.next()));
    }
    return sum;
  }
}
```

Unfortunately, **addUp** also has an error, similar to the error in (a). Explain the error and write a single JUnit test case that fails because of it.

```
public class StringTest extends junit.framework.TestCase {

  public void testAddUp() {










  }
}
```

Part (e) [5 marks]

Many kinds of addable objects have a "zero": an object that doesn't change objects it's added to. Show the modifications to Addable and Rational to allow users access to a zero object. Note: you will be using this in (f). And don't worry about consistency with your answer to (c); in particular always give users a new zero.

```
interface Addable {

  void add(Addable a);



}
```

Part (f) [5 marks]

Assuming the modifications from part (e), write a version of addUp that doesn't modify the first element, and that takes an Iterator instead of a LinkedList (and that works correctly).

Question 2. [8 MARKS]

Consider the following classes:

```
class C {
  public String m() {
    return "Hi";
  }
}
class D extends C {
  public String m() {
    return "Bye";
  }
}
```

Beside each of the following statements, executed in the Interactions Pane of DrJava, write either "Compile error", "Exception" or the return value, as appropriate.

```
C c = new C();

c.m()

((D)c).m()

C c = new D();

D d = new D();

d.m()

((C)).m()

D d = new C();
```

Total Marks = 42