CS201 Final Exam **Name** ______________________________
100 Total Points
Open Book, Open Notes

Short Answer. (20 pts) Provide a **brief** answer to the following questions.

1. Every class that we create is automatically derived from what other class?

2. If we compare two array variables using ==, (e.g. ary1 == ary2) what is Java really comparing?

3. Under what circumstances might we want to declare a class as **abstract**?

4. Why is a typecast necessary when extracting an element from a Vector?

5. Can a private method in a base class be accessed by name from a derived class?

6. **(12 pts) Potpourri**. Give the output of the following code snippets.

a)

```
char[] arr1 = {‘a’,’b’,’c’};
char[] arr2 = arr1;
arr1[0]=’z’;
System.out.println(arr2[0]);
```

b)

```
class A
{
 public void foo() {
  System.out.println("In A");
 }
}

class B extends A
{
 public void foo() {
  System.out.println("In B");
 }
}

class Foo
{
 public static void main(String[] args)
 {
  A myObj = new B();
  myObj.foo();
 }
}
```

c)

```
class A
{
 public int num;
 public A(int x) { num = x; }
 public boolean equals(Object other)
 {
   A otherA = (A) other;
   return ((otherA.num + this.num)==7);
 }
}

class Foo
{
 public static void main(String[] args)
 {
   Vector v = new Vector();
   v.addElement(new A(3));
   v.addElement(new A(4));
   v.addElement(new A(5));
   System.out.println(v.indexOf(new A(2)));
   System.out.println(v.indexOf(new A(4)));
 }
}
```

7. **(12 pts) Arrays.** Use the following method heading for FindMin. This method is passed a 2D array of integers and returns an object of type Coord:

```
public static Coord FindMin(int[][] arr)
```

Here is the Coord class:

```
public class Coord
{
      public int row, column;
      public Coord(int r, int c) {
            row = r; column = c;
      }
}
```

Write the method **FindMin** so that it returns the coordinate of the smallest value in the 2D array.

For example, if your method is used with the following code:

```
public static void main(String[] args)
{
      int[][] arr = new int[10][10];
      Coord c = null;

      // Initialize array to zero
      for (int i=0; i<10; i++)
            for (int j=0; j<10; j++)
                  arr[i][j]=0;
      arr[5][3] = -4;                // Set one value to -4
      c = FindMin(arr);              // ← You write FindMin

      System.out.println("Min at " + c.row + " " + c.column +
                         " Val =" + arr[c.row][c.column]);
}
```

This program should output:

```
Min at 5  3  Val = -4
```

8. **(10 pts) Graphics and Loops.** Draw the output of the following program:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class WindowDestroyer extends WindowAdapter
{
    public void windowClosing(WindowEvent e)
    {
        System.exit(0);
    }
}

class MyWindow extends JFrame
{
      public MyWindow()
      {
            this.setSize(300, 300);
            this.addWindowListener(new WindowDestroyer());
            this.setTitle("Fun Stuff");
            this.setVisible(true);

      }

      public void paint(Graphics g)
      {
            for (int i=0; i<255; i++) {
                  g.setColor(new Color(i,0,0));
                  g.drawLine(25,i+25,i+25,i+25);
            }
      }
}

class DrawSomething
{
      public static void main(String[] args)
      {
            MyWindow w = new MyWindow();
      }
}
```

9. **(11 pts) File I/O**. You have given monetary donations to a number of individuals. They are stored in the text file “donations.txt” where the name is stored first followed by the amount of money donated. An “*” for the name signals the end of the file. For example, the file might contain:

```
Nephew George
400.00
Salvation Army
200.00
United Way
100.00
Kenrick Mock
150000.00
*
```

Write a method that reads this file and outputs the sum of the donations. For the above file, the code should output 150700.

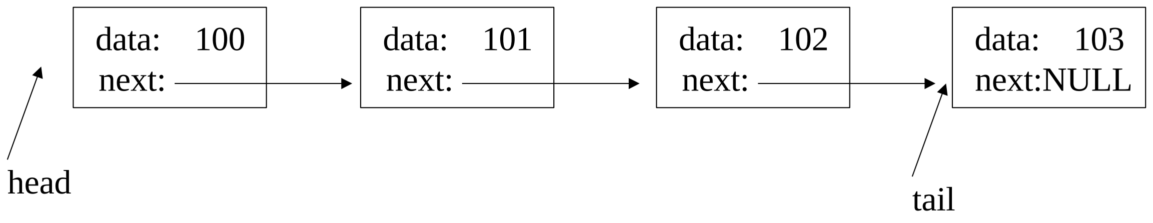
10. **(12 pts). Inheritance**. Give the output of the program below.

```
class Stuff
{
      private int x;
      public Stuff() {
            x = 5;
      }
      public Stuff(int i) {
            x = i;
      }
      public int getVal() {
            return x;
      }
}

class MoreStuff extends Stuff
{
      private int y;
      public MoreStuff() {
            y = 2;
      }
      public MoreStuff(int i, int j) {
            super(i);
            y = j;
      }
      public void setY(int j) {
            y = j;
      }
      public int getVal() {
            return (super.getVal() * y);
      }
}

class Inherit
{
      public static void main(String[] args) {
            Stuff s1 = new Stuff();
            Stuff s2 = new Stuff(4);
            MoreStuff s3 = new MoreStuff();
            MoreStuff s4 = new MoreStuff(6,7);

            PrintStuff(s1);
            PrintStuff(s2);
            PrintStuff(s3);
            PrintStuff(s4);
            s1 = s3;
            s3.setY(3);
            PrintStuff(s3);
            PrintStuff(s1);
      }

      public static void PrintStuff(Stuff s) {
            System.out.println(s.getVal());
      }
}
```

11. **(12 pts) Pointer Data Structures**. The code below constructs a singly linked list of four elements:

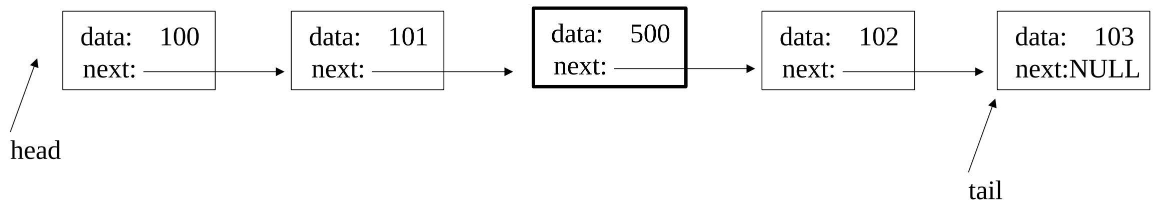
```
class Node
{
    public int data;
    public Node next;
    public Node(int i) {
          data = i;
          next = null;
    }
}

class LinkedList
{
    public static void main(String[] args)    {
          Node head=null, newData = null, tail = null;
          int i;

          for (i=0; i<4; i++) {
                newData = new Node(100+i);
                if (head == null) {
                      head = newData;
                      tail = newData;
                }
                else {
                      tail.next = newData;
                      tail = newData;
                }
          }
          // Code for you to write would start here
    }
}
```

Graphically, the list that is created looks like this:

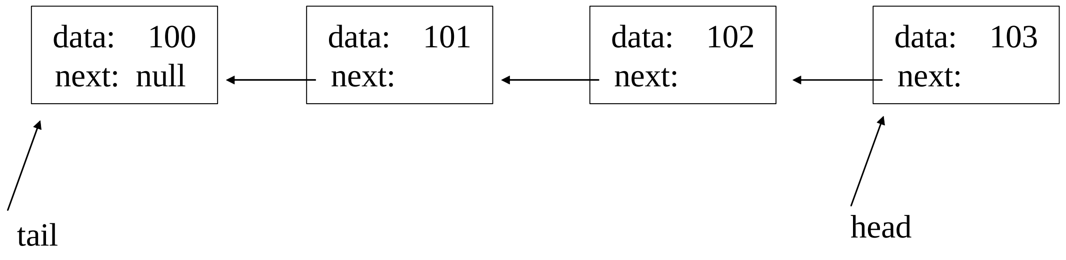


(problem continued on the next page)

a) Write code that inserts a node with the data value 500 in between the second and third nodes. The new linked list should look like this:



b) Write code that reverses the linked list. The easiest way to do this is to change the "next" pointers around, although it is also possible to swap values among nodes. Your code should work on an arbitrary number of nodes, not for just this case of four nodes. For example, if the original list were reversed, it would result in the following graphical picture:



12. **(11 pts) Recursion - Handshake problem**.

We have *n* people in a room, where *n* is an integer greater than or equal to 2. Each person shakes hands once with every other person. What is the total number *h(n)* of handshakes?

Write a recursive method to solve this problem with the following header:

```
public static int handshake(int n)
```

Where handshake(n) returns the total number of handshakes for *n* people in the room.

To get you started, if there are only one or two people in the room, then:

handshake(1) = 0
handshake(2) = 1

If a third person enters the room, she must shake hands with each of the two persons already there. Add these two handshakes to the previous number of handshakes and we get:

handshake(3) = handshake(2) + 2

If a fourth person enters the room, she must shake hands with each of the three persons already there. Add these three handshakes to the previous number of handshakes and we get:

handshake(4) = handshake(3) + 3

If you can generalize this to *n* handshakes then it should help you write the recursive solution.

(blank page for extra space)