CS100B: Prelim 2 Sample
October 19, 1999
7:30 PM – 9:00 PM

(Print your name)

(Signature)

(Student ID)

Sections 10 11 12 13 14 15
(circle one) Mon Mon Mon Tue Tue Tue
1:25 2:30 3:35 10:10 2:30 3:35

Instructions:
• CIRCLE YOUR SECTION! Otherwise, your test will be sent to Carpenter.
• Answer all questions by yourself! Respect academic integrity.
• Sign or initial each page.
• Show all work and comment code fragments to receive partial credit
• Use the back of each page if you need more space.
• Remember to breathe! Relax, it’s only a test.

Points:

1. _____ (10 points)

2. _____ (15 points)

3. _____ (20 points)

4. _____ (25 points)

5. _____ (30 points)

Subtotal: _____/100 points

6. _____ (2 possible bonus points)

Total: ______
**Problem 1  (10 points) Short Answer questions**

Answer the following questions. Try to keep your answers concise. (Know the definitions to all the following terms to answer the 5 questions I’m not showing.) Example: Describe the client-server relationship. (Keep answers brief!)

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>abstraction</td>
<td>new</td>
</tr>
<tr>
<td>actual parameter</td>
<td>object</td>
</tr>
<tr>
<td>address</td>
<td>object oriented programming</td>
</tr>
<tr>
<td>alias</td>
<td>parameter</td>
</tr>
<tr>
<td>attribute</td>
<td>pass by reference</td>
</tr>
<tr>
<td>behavior</td>
<td>pass by value</td>
</tr>
<tr>
<td>call</td>
<td>private</td>
</tr>
<tr>
<td>class</td>
<td>procedural programming</td>
</tr>
<tr>
<td>class variable</td>
<td>public</td>
</tr>
<tr>
<td>class</td>
<td>reference</td>
</tr>
<tr>
<td>client</td>
<td>reference variable</td>
</tr>
<tr>
<td>constructor</td>
<td>return type</td>
</tr>
<tr>
<td>final</td>
<td>return value</td>
</tr>
<tr>
<td>flow of control</td>
<td>server</td>
</tr>
<tr>
<td>formal parameter</td>
<td>service method</td>
</tr>
<tr>
<td>garbage collection</td>
<td>state</td>
</tr>
<tr>
<td>identity</td>
<td>static</td>
</tr>
<tr>
<td>information hiding</td>
<td>static variable</td>
</tr>
<tr>
<td>instance variable</td>
<td>string</td>
</tr>
<tr>
<td>instantiation</td>
<td>String</td>
</tr>
<tr>
<td>interval analysis</td>
<td>structured programming</td>
</tr>
<tr>
<td>invoke</td>
<td>support method</td>
</tr>
<tr>
<td>Java</td>
<td>type</td>
</tr>
<tr>
<td>local variable</td>
<td>visibility</td>
</tr>
<tr>
<td>method</td>
<td>visibility modifier</td>
</tr>
<tr>
<td>method definition</td>
<td>void</td>
</tr>
<tr>
<td>modifier</td>
<td></td>
</tr>
<tr>
<td>method invocation</td>
<td></td>
</tr>
</tbody>
</table>
Problem 2  (15 points) Tracing

What is the output of the following code? Hint: Yes, this code does in fact compile and run.

// Class for an integer Number
class Num{
    int value;
    Num() {}    
    public void add(Num number){
        number.value++;    
    }
}

// Demonstrates the effects of parameter passing
class Parameter_Passing{
    public static void print(int value1, Num value2){
        System.out.println("value1= "+value1);
        System.out.println("value2= "+value2.value);
        System.out.println();
    } // method print

    public static void change1(int value1, Num value2){
        value1++;
        value2.value++;
        print(value1,value2);   // 2nd Output
    } // method change1

    public static int change2(int value1, Num value2){
        value1++;
        value2.value++;
        print(value1,value2);   // 4th Output
        return value1;
    } // method change2

    public static void main (String[] args){
        int number1=5;
        Num number2=new Num();
        number2.value=10;
        print(number1,number2); // 1st Output

        change1(number1, number2);
        print(number1,number2); // 3rd Output

        number1=change2(number1, number2);
        print(number1,number2); // 5th Output

        number2.add(number2);
        print(number1,number2); // 6th Output
    } // method main
} // class Parameter_Passing
Output from program:

```plaintext
value1=  5
value2=  10

value1=  6
value2=  11

value1=  5
value2=  11

value1=  6
value2=  12

value1=  6
value2=  12

value1=  6
value2=  13
```
Problem 3 (20 points) Nested Loops

Write a program that will generate the following tabular output for a user entered integer n. The main “diagonal” (the elements whose rows and columns match) must values of 1. All other elements are zero. See example output for different values of n.

\[ \begin{align*}
    n = 1 & \rightarrow 1 \\
    n = 2 & \rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \\
    n = 3 & \rightarrow \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}
\end{align*} \]

Your output won’t have the \( n = \# \rightarrow \) portion, just the table of 0s and 1s. Though not required, documenting your code will help you earn partial credit.

```java
public class Identity {
    public class static void main(String args[]) {
        // complete the code
        int nn; // size of square matrix
        int row; // loop index
        int col; // loop index

        // initialize parameters
        nn = 5;

        // print an nn x nn identity matrix

        // for each row
        for (row = 1; row <= nn; row++) {
            // print the columns (spaces) in that row
            for (col = 1; col <= nn; col++) {
                if (row == col)
                    System.out.print ("1 ");
                else
                    System.out.print ("0 ");
            }
            // go to the next row (line)
            System.out.print ("\n");
        }
        System.out.flush();
    }
}
```
**Problem 4**  
(25 points) Complete the class!

We're already familiar with interval arithmetic for adding two intervals. Now we wish to extend this notion to add three intervals. Using the code below as a starting point, write a method called `addInterval` which can be invoked on an `Interval` object. This method should take two `Interval` objects as arguments and return the sum of the two `Interval` objects and the `Interval` object that invoked the addition method.

Note: Though not required, documenting your code will earn partial credit.

```java
public class Interval {

    public double minimum;
    public double maximum;

    public Interval(double min, double max) {
        minimum = min;
        maximum = max;
    }

    // complete the addition code here

    public Interval addInterval(Interval intOne, Interval intTwo) {
        // add min values
        double tempMin = minimum + intOne.minimum + intTwo.minimum;

        // add max values
        double tempMax = maximum + intOne.maximum + intTwo.maximum;

        Interval tempInterval = new Interval(tempMin, tempMax);
        return tempInterval;
    } // method addInterval

    // END OF ANSWER

} // class Interval
```
Problem 5 (30 points) Program (20 points correctness, 5 points style, 5 points documentation: all mandatory!)
Write a program that performs interval subtraction and division between two intervals. Assume that a user enters the lower and upper bounds of both intervals. The user must also choose the operation. You should use a class that called Interval that contains all appropriate methods and variables. You do not need to employ encapsulation. Be sure to check if the user enters a zero and chooses division.

// PL2 problem5 sample

class Interval {
    // instance variables
    private double min;
    private double max;

    // constructors

    // default:
    Interval() {
    } // constructor Interval

    // user can set values with this constructor:
    Interval(double x, double y) {

        // check if user entered values in correct order
        if (x < y) {
            min = x;
            max = y;
        }

        // user entered values backwards, so swap:
        else {
            System.out.println("You entered your interval backwards!");
            System.out.println("Reversing interval values....");
            min = y;
            max = x;
        }
    } // constructor Interval

    // Interval subtraction
    // [a1,a2] - [b1,b2] = [a1-b2,a2-b1]
    public Interval intsub(Interval I2) {
        double I3min = min - I2.max;
        double I3max = max - I2.min;
        Interval I3 = new Interval(I3min, I3max);
        return I3;
    } // method intsub

    // Interval division
    // [a1,a2] / [b1,b2] = [min(a1/b1,a1/b2,a2/b1,a2/b2),
    // max(a1/b1,a1/b2,a2/b1,a2/b2)]
public Interval intdiv(Interval I2) {
    // check if dividing by zero (I2 contains zero)
    if (I2.min <= 0 && I2.max >= 0) {
        System.err.println("Dividing by zero not allowed, fool!");
        System.exit(0); // exit from program
    }
    // alternative check
    if (I2.min*I2.max <= 0) {
        System.err.println("Dividing by zero not allowed, fool!");
        System.exit(0); // exit from program
    }
    // otherwise, continue with division
    // all combinations to check:
    double LL,LU,UL,UU,I3min,I3max;
    LL = min/I2.min;
    LU = min/I2.max;
    UL = max/I2.min;
    UU = max/I2.max;
    // pick initial values
    I3min = I3max = LL;
    // sort out the min
    if (LU < I3min)
        I3min = LU;
    if (UL < I3min)
        I3min = UL;
    if (UU < I3min)
        I3min = UU;
    // sort out the max
    if (LU > I3max)
        I3max = LU;
    if (UL > I3max)
        I3max = UL;
    if (UU > I3max)
        I3max = UU;
    // return solution
    Interval I3 = new Interval(I3min,I3max);
    return I3;
} // method intdiv

public void intprint() {
public class interval {

    public static void main(String args[]) {

        // set up user input
        TokenReader in = new TokenReader(System.in);
        System.out.println("Welcome to wonderful world of Interval Arithmetic!");
        System.out.println();

        // enter and assign interval 1
        System.out.print("Enter interval 1 min: ");
        double I1min = in.readDouble();
        System.out.print("Enter interval 1 max: ");
        double I1max = in.readDouble();
        Interval I1 = new Interval(I1min,I1max);
        System.out.print("Interval 1: ");
        I1.intprint();
        System.out.println();

        // enter and assign interval 2
        System.out.print("Enter interval 2 min: ");
        double I2min = in.readDouble();
        System.out.print("Enter interval 2 max: ");
        double I2max = in.readDouble();
        Interval I2 = new Interval(I2min,I2max);
        System.out.print("Interval 2: ");
        I2.intprint();
        System.out.println();

        // create interval 3 to store results of I1 (op) I2
        Interval I3 = new Interval();

        // choose operation and computer result:
        System.out.print("Choose your operation [1=sub,2=div]. Choice? ");
        int choice = in.readInt();

        // subtraction
        if (choice == 1) {
            I3 = I1.intsub(I2);
            System.out.println("The result is ");
            I3.intprint();
        }

        // division
        else if (choice == 2) {

        }
    }
}

System.out.print("["+min","+max+"]\n");
} // method intprint

} // class Interval
I3 = I1.intdiv(I2);
System.out.print("The result is ");
I3.intprint();
System.out.println("Thank you for you patronage. Come again!");
}

else
    System.out.println("Don't understand choice. Try again!");

} // method main

} // class interval
Problem 6  
(2 points) Bonus problems (optional! REGRADE REQUESTS UNLIKELY TO BE ACCEPTED)

6a) (0.5 points) What does <something> stand for (in <something>)?

6b) (0.5 points) Describe <somethings> that guarantee <something> of <something>.

6c) (0.1 points) Who <did something> to <something> to <do something>?

6d) (0.1 points) What does the expression <something> mean?

6e) (0.1 points) What is the <something> given <something> and <something>?

6f) (0.1 points) What does <something> do?

6g) (0.1 points) What does <a different something> do?

6h) (0.5 point) What is the <something> of <someone’s> <something>?