

CIS 120

Midterm I — October 9, 2007

Name: _____

Student ID: _____
(from your PennCard)

Email _____ @seas.upenn.edu

Lab:(circle one)

201	202	203	204	205	206
W11	W12	W4	R10	R3	R4

- This exam text and attached code, the *Java Backpack Reference Guide*, and your brain are the only information sources you can use during this exam.
- You have 80 minutes to answer all of the questions. The entire exam is worth 100 points. The point value of each question is given.
- Write your answers on the exam pages. The back of the exam pages and the code handout can be used as scratch pad.
- The answers to the questions require only basic Java language facilities and the code given as appendix. No other classes or packages are needed.
- Questions during the exam should be about the wording of the exam only. If you have a question, raise your hand and we'll come to you. (This is less disruptive for others than if you come to us.)
- **Partial credit will be given where appropriate.**
- **DON'T PANIC!** If you find a question that you cannot solve right away, move on and return to it after you finish the other questions.

	Score	Max
1		18
2		16
3		16
4		28
5		22
Total		100

1. The following questions pertain to some proposed new methods for the `IntSeq` class discussed in the lectures and included in the code handout. We assume that the methods have been successfully added to and compiled with the class in DrJava, and that the following DrJava command was given:

```
> IntSeq u = new IntSeq(new int[] {1, -3, 5, -7, 4});
```

After each of the new methods, we give a new DrJava interaction involving the method, and your job is to select the result of the interaction.

(a)

```
public int m1() {
    int t = 0;
    for (int i = 1; i < data.length; i++)
        if (data[i] > data[i-1])
            t += data[i] - data[i-1];
        else
            t += data[i-1] - data[i];
    return t;
}
```

4 points

Interaction:

```
> u.m1()
0    10    20    35
```

(b)

```
public int m2(IntSeq v) {
    int t = 0;
    for (int i = 0; i < size() && i < v.size(); i++)
        t += data[i] * v.get(i);
    return t;
}
```

4 points

Interaction:

```
> u.m2(u)
-16    0    100    420
```

(Two more questions for this section on the next page)

```
(c) public int m3() {
    if (data.length < 2) return 0;
    int x = data[0], y = data[0];
    for (int i = 1; i < data.length; i++) {
        if (data[i] < x) x = data[i];
        if (data[i] > y) y = data[i];
    }
    return y - x;
}
```

5 points

Interaction:

```
u.m3()
-7  5  11  12
```

```
(d) public int m4() {
    int x = 0, j = -1;
    for (int i = 1; i < data.length; i++)
        if (data[i] > data[i-1] + x) {
            x = data[i] - data[i-1];
            j = i - 1;
        }
    return j;
}
```

5 points

Interaction:

```
> u.m4()
-1  2  3  4
```

2. You compiled the following classes in DrJava

```
class I {
    protected double x;
    private String s;
    I(String s, double x) {
        this.s = s;
        this.x = x;
    }
    double m(int i) {
        return x*i;
    }
    public String toString() {
        return s;
    }
}
class J extends I {
    private double y;
    J(String s, double x, double y) {
        super(s + "J", x);
        this.y = y;
    }
    double m(int i) {
        return super.m(i) * (1-y);
    }
}
class K extends I {
    private int j;
    private double y;
    K(String s, double x, int j, double y) {
        super(s, x);
        this.j = j;
        this.y = y;
    }
    double m(int i) {
        int k = (i/j)*j;
        int l = i%j;
        return k*x*(1-y) + l*x;
    }
    public String toString() {
        return super.toString() + "K";
    }
}
```

followed by this interaction:

```
> I[] a = {new I("A", 2), new J("B", 2, .1), new K("C", 2, 10, .2)};
```

For each of the interactions on the next page, circle the correct output:

- (a) > a[0].m(15) *4 points*
2 15 30
- (b) > a[1] *4 points*
B J BJ
- (c) > a[1].m(15) *4 points*
15 27 30
- (d) > a[2].m(15) *4 points*
26 27 30

3. The Java source listing attached to the end of this exam gives an extension of the `Ticket` class discussed in the lectures. Assume that the DrJava interaction lines below are being typed one immediately after the other after the class was compiled. For each line, circle the correct response from DrJava. The response "OK" corresponds to a DrJava interaction that produces no output and causes no error. *2 points each*

- (a) `Ticket t1 = new Ticket();` error OK
- (b) `t1.setCurrentPrice(10);` error OK
- (c) `Ticket t2 = new Ticket();` error OK
- (d) `t1.getPrice()` error 0.0 5.0 10.0
- (e) `Ticket.revenue` error 5.0 10.0 15.0
- (f) `Ticket[] ts =`
 `new Ticket[10];` error OK
- (g) `for (int i = 0; i < 10; i++)` error OK
 `ts[i] = new Ticket();`
- (h) `Ticket.getIssued()` error 2 10 12

4. In these questions, you will add several new methods to the class `IntSeq`, which is provided in the code handout. Here are some interactions with the methods you will provide:

```
> IntSeq u = new IntSeq(new int [] {1,2,3,4});
> IntSeq v = new IntSeq(new int [] {6,7,8});
> IntSeq w = new IntSeq(new int [] {0});
> u
1 2 3 4
> u.splice(1,3,v)
> u
1 6 7 8 4
> u.splice(1,4,w)
> u
1 0 4
> u.rotate()
> u
0 4 1
> u.truncate(1)
> u
0
> u.splice(0,1,v)
> u
6 7 8
```

None of the methods you will provide depends on any of the others. For the methods that may change the size (number of elements) of the sequence, recall the recipe given in class, where a new array of appropriate length is created, the new sequence is stored in the array, and finally the new array is set as the data for the sequence.

- (a) Write below a method `void rotate()` that rotates the sequence it is invoked on one position “counterclockwise”. That is, the 0th element becomes the last element, the 1st element becomes the 0th element, the 2nd becomes the 1st, and in general the i th element becomes the $i - 1$ th element for $i > 0$. For example, the sequence 1 3 5 7 would become 3 5 7 1. The method should have no effect on a sequence with fewer than two elements. **8 points**

(Two more questions for this problem on the next page)

- (b) Write below a method `void truncate(int pos)` that changes the sequence to consist of the elements $0 \cdots \text{pos} - 1$ of the original sequence. For example, if the original sequence is 7 6 5 and `pos` is 2, the changed sequence will be 7 6. The parameter `pos` is assumed to range from 0 to the original size of the sequence; you don't need to worry about values outside that range in your code. **8 points**

- (c) Write below a method `void splice(int from, int to, IntSeq mid)` that replaces the elements in the given sequence starting at position `from` and ending at position `to - 1` by the elements of the sequence `mid`. For example, if the original sequence is 3 5 7 9, `from` is 1, `to` is 3, and `mid` is 4 6 8, the sequence becomes 3 4 6 8 9; if `from` and `to` are still 1 and 3, but `mid` is 0, then the sequence becomes 3 0 9. You should assume that $0 \leq \text{from} \leq \text{to} \leq \text{size}()$. If `from` and `to` are the same position, then `mid` is inserted into the sequence at that position. If `mid` is an empty sequence, then the elements starting at `from` and ending at `to - 1` are just removed from the sequence. (If you write the method cleanly, these two cases fall out with any additional programming.) **12 points**

5. This problem refers to shape classes that was discussed in the lectures and provided in the code handout.

(a) Each of the following DrJava interactions were typed into DrJava right after compiling the shape interfaces and classes. Select whether the interactions would cause an error or run normally.

i. `Displaceable[] d =
 {new Point(1,2),new Circle(new Point(3,4),1),new Square(new Point(0,0),1)};` **2 points**
`for (int i = 0; i < d.length; i++)
 d[i].move(-1, -1);`
error normal

ii. `Displaceable[] d =
 {new Point(1,2),new Circle(new Point(3,4),1), new Square(new Point(0,0),1)};` **2 points**
`double a = 0;`
`for (int i = 0; i < d.length; i++)
 a += d[i].getArea();`
error normal

iii. `Displaceable[] d =
 {new Point(1,2), new Circle(new Point(3,4),1), new Square(new Point(0,0),1)};` **2 points**
`double a = 0;`
`for (int i = 0; i < d.length; i++)
 a += ((Area)d[i]).getArea();`
error normal

iv. `Displaceable[] d =
 {new Point(1,2), new Circle(new Point(3,4),1), new Square(new Point(0,0),1)};` **2 points**
`double a = 0;`
`for (int i = 0; i < d.length; i++)
 if (d[i] instanceof Area)
 a += ((Area)d[i]).getArea();`
error normal

- (b) A `Ring` is a pair of concentric `Circles` specified by a center, the radius of the inner circle, and a width, which is the difference between the radius of the outer circle and the radius of the inner circle. The width is assumed to be non-negative. The incomplete class definition below should be completed to implement `Ring` as a `Displaceable` and an `Area`. The area of the ring is the difference between the area of its outer circle and the area of its inner circle. To move a ring, we move its two circles. 14 points

```
// Include appropriate implements or extends clauses in the class declaration
```

```
public class Ring {
```

```
    private Circle inner, outer;
```

```
    public Ring(Point center, double radius, double width) {  
        // create the inner and outer circles
```

```
    }
```

```
// implement the Displaceable methods
```

```
// implement the Area method
```

```
}
```