Unit 3 Exam Key

Code Tracing

#	Output
1	`[1, 8, 4, 6]`

2	`2 [0, 0, 1, 0]`
	`1 [0, 0, 1, 0]`
	`3 [0, 0, 1, 1]`
	`2 [0, 0, 1, 1]`

3	`runtime error or infinite loop`

4	`[0, 1, 2]`
	`[1, 2, 6]`
	`[2, 3, -3]`

5	`runtime error`
6	`[C, T, F]`
7	`18`
8	`=+=+*+=+=`

OOP with Beyonce

```
Epic
Epic 1   Revolution 2   Epic 1
Epic
Epic 1   Epic 2
Lemonade Homecoming
Homecoming 1   Revolution 2   Homecoming 1   Lemonade 2
Homecoming
Homecoming 1   Revolution 2   Homecoming 1
```

PROGRAMMING

Program - File Processing

```
20      public static double groceryTotal(Scanner scan) {
21          double red = 0;
22          double blue = 0;
23          double none = 0;
24          while (scan.hasNext()) {
25              scan.next();
26              String sale = scan.next();
27
28              if (sale.equalsIgnoreCase("red")) {
29                  red += scan.nextDouble();
30              } else if (sale.equalsIgnoreCase("blue")) {
31                  blue += scan.nextDouble();
32              } else {
33                  none += scan.nextDouble();
34              }
35          }
36          return red * .90 + blue * .75 + none;
37      }
```

Item	Line #	Item	Points
A	20	header	+1 return type +1 parameters
B	24	loop	+3 correct loop/check for EOF with next
C	26-33	reading items	+2 next() for item, consume it +2 next() for sale type +2 nextDouble() for price
D	28	sale type decision	+2 if statement for sale type +3 formula with adjustment for sale type
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)

Program - 2D Arrays

```
95      public static int[] getMaxSum(int[][] nums2D) {
96          int[] result = new int[2];
97          int max = Integer.MIN_VALUE;
98
99          for (int r = 0; r < nums2D.length - 1; r++) {
100             for (int c = 0; c < nums2D[0].length - 1; c++) {
101                 int sum = nums2D[r][c] + nums2D[r][c + 1] +
102                         nums2D[r + 1][c] + nums2D[r + 1][c + 1];
103                 if (sum > max) {
104                     max = sum;
105                     result[0] = r;
106                     result[1] = c;
107                 }
108             }
109         }
110         return result;
111     }
```

Item	Line #	Item	Points
A	95	header	+1 return value +1 parameters
B	96-97	loop prep	+1 create result array +2 initialize max
C	99-100	nested loops	+2 nested loops to traverse 2D array +2 adjustment (-1) to avoid out of bounds
D	101-102	sum calculation	+3 sum calculation
E	103-107	max update	+2 correct if statement (sum > max, no =) +1 max update +1 result update
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)

Program - ArrayLists

```
123     // Array List - Regular Test
124     public static ArrayList<String> removeWords(ArrayList<String> words,
125             String infrequent) {
126         ArrayList<String> allInfrequent = new ArrayList<String>();
127         int i = 0;
128         while (i < words.size()) {
129             String temp = words.get(i).toLowerCase();
130             boolean allInfreq = true;
131             int indexInString = 0;
132             while (allInfreq && indexInString < temp.length()) {
133                 allInfreq = infrequent.indexOf(temp.charAt(indexInString)) != -1;
134                 indexInString++;
135             }
136             if (allInfreq)
137                 allInfrequent.add(words.remove(i));
138             else
139                 i++;
140         }
141         return allInfrequent;
142     }
```

Item	Line #	Item	Points
A	124-125	header (any method name)	+1 return type +1 parameters
B	128	loop	+2 loop (initialize, test and increment)
C	130-135	all infrequent?	+2 loop - general +2 loop leaves early if possible +2 check each character
D	136-137	update ArrayLists	+2 remove allInfrequent from words +2 add allinfrequent to return ArrayList
E	141	return	+2 correct return
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)

Programming - OOP - Implementing a Class

```
1  public class Dragonfly extends Critter {
2      private int moves;
3      private int east;
4      private int maxEast;
5      private boolean up;
6
7      private static final Attack[] DRAGONFLY_ATTACKS =
8          { Attack.ROAR, Attack.POUNCE, Attack.SCRATCH, Attack.FORFEIT };
9
10     public Dragonfly() {
11         moves = 0;
12         east = 0;
13         maxEast = 1;
14         up = false;
15     }
16
17     public Attack fight() {
18         return (DRAGONFLY_ATTACKS[moves % DRAGONFLY_ATTACKS.length]);
19     }
20
21     public boolean eat() {
22         maxEast++;
23         return true;
24     }
25
26     public Direction getMove() {
27         moves++;
28         if (east > 0) {
29             east--;
30             return Direction.EAST;
31         } else {
32             east = maxEast;
33             up = !up;
34             if (up) {
35                 return Direction.NORTH;
36             } else {
37                 return Direction.SOUTH;
38             }
39         }
40     }
41 }
```

Item	Line #	Item	Points
A	1-5	header and fields	+1 public class and extends +1 private fields
B	10-15	constructor	+1 initializes values
C	17-18	fight()	+4 mechanism for order of attacks with wraparound
D	21-24	eat()	+2 increase max east +2 return true
E	26-41	getMove()	+1 increment moves (if needed) +2 handle moves east (keep count and return east) +2 handle moves north and south (reset max east, decide on north and south)
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)

Programming - OOP - Classes and Objects

```
129     public void advance(int mins) {
130         minute += mins; // add into 'minute' field, then
131         while (minute >= 60) {
132             minute -= 60;
133             hour++;
134             if (hour == 12) {
135                 if (amPm.equals("AM")) {
136                     amPm = "PM";
137                 } else {
138                     amPm = "AM";
139                 }
140             } else if (hour > 12) {
141                 hour = 1;
142             }
143         }
144     }
```

Item	Line #	Item	Points
A	129	header	+1 return type +1 parameters
B	130	update minutes	+2 update minutes
C	131	hours loop	+3 loop correct number of times
D	132-141	AM-PM	+3 break minutes into hours +3 if statement for AM-PM +3 reset hour at end of day
X	N/A	Instructions	MAX of -5 -2 Unnecessary code -2 Redundant or inefficient code -2 Seriously incorrect style guide issue -2 Java syntax not allowed -2-4 Not following instructions (if not reflected above already)