CSE 142, Summer 2010
Final Part A Exam Key

1. Array Mystery

Expression	Final Contents of Array
`int[] a1 = {42, 42};` `arrayMystery(a1);`	`{42, 0}`
`int[] a2 = {6, 2, 4};` `arrayMystery(a2);`	`{6, -4, 10}`
`int[] a3 = {7, 7, 3, 8, 2};` `arrayMystery(a3);`	`{7, 1, 2, -1, 8}`
`int[] a4 = {4, 2, 3, 1, 2, 5};` `arrayMystery(a4);`	`{4, -2, 3, 0, -2, 6}`
`int[] a5 = {6, 0, -1, 3, -2, 0, 4};` `arrayMystery(a5);`	`{6, -6, 4, -1, 5, -11, 17}`

2. Reference Semantics Mystery

```
2 [8, 16] 16 123
4 [8, 16] 16 123
3 [10, 17] 42 223
5 [10, 17] 42 223
```

3. Inheritance Mystery

```
kate
kate-L
kate-L   locke-F

jack
kate-L   jack-L
kate-L   jack-L   locke-F

jack
kate-L   jack-L
sawyer-F   kate-L   jack-L   locke-F

kate
kate-L
kate-F
```

4. File Processing

```
public static void triathlon(Scanner input) {
    String name = input.next();
    int totalTime, firstTime;
    totalTime = firstTime = input.nextInt() + input.nextInt() + input.nextInt();
    System.out.printf("%s: %d min\n", name, totalTime);

    while (input.hasNext()) {
        name = input.next();
        totalTime = input.nextInt() + input.nextInt() + input.nextInt();
        System.out.printf("%s: %d min (+%d min)\n", name, totalTime,
            totalTime - firstTime);
    }
}
```

5. Array Programming (5 Solutions Shown)

```
// single for loop with mod
public static String[] sandwich1(String bread, String[] filling, int fillFactor) {
    String[] sandwich = new String[2 + filling.length * fillFactor];

    sandwich[0] = bread;
    sandwich[sandwich.length - 1] = bread;

    for (int i = 0; i < filling.length * fillFactor; i++) {
        sandwich[i + 1] = filling[i % filling.length];
    }

    return sandwich;
}

// nested for loops, outer as fillFactor, inner as filling
public static String[] sandwich2(String bread, String[] filling, int fillFactor) {
    String[] sandwich = new String[2 + filling.length * fillFactor];

    sandwich[0] = bread;
    sandwich[sandwich.length - 1] = bread;

    for (int i = 0; i < fillFactor; i++) {
        for (int j = 0; j < filling.length; j++) {
            sandwich[j + 1 + (i * filling.length)] = filling[j];
        }
    }

    return sandwich;
}
```

```
// nested for loops, outer as sandwich incrementing with filling length,
// inner as filling
public static String[] sandwich3(String bread, String[] filling, int fillFactor) {
    String[] sandwich = new String[2 + filling.length * fillFactor];

    sandwich[0] = bread;
    sandwich[sandwich.length - 1] = bread;

    for (int i = 1; i < sandwich.length - 1; i += filling.length) {
        for (int j = 0; j < filling.length; j++) {
            sandwich[i + j] = filling[j];
        }
    }

    return sandwich;
}

// single for loop with additional counter
public static String[] sandwich4(String bread, String[] filling, int fillFactor) {
    String[] sandwich = new String[filling.length * fillFactor + 2];
    sandwich[0] = bread;
    sandwich[sandwich.length - 1] = bread;
    int count = 0;

    for (int i = 1; i < sandwich.length - 1; i++) {
        if (count >= (filling.length - 1)) {
            count = 0;
        }

        sandwich [i] = filling[count];
        count++;
    }

    return sandwich;
}

// nested for loops, outer as fillFactor, inner as filling, using counter
public static String[] sandwich5(String bread, String[] filling, int fillFactor) {
    String[] sandwich = new String[2 + filling.length * fillFactor];

    sandwich[0] = bread;
    sandwich[sandwich.length - 1] = bread;
    int i = 1;

    for (int j = 1; j <= fillFactor; j++) {
        for (int k = 0; k < filling.length; k++) {
            sandwich[i] = filling[k];
            i++;
        }
    }

    return sandwich;
}
```