# CSE 142, Autumn 2010
# Final Exam Key

## 1. Array Mystery

| Expression | Final Contents of Array |
|---|---|
| `int[] a1 = {42, 42, 99};`<br>`arrayMystery(a1);` | `{42, 47, 99}` |
| `int[] a2 = {6, 18, 4, 25};`<br>`arrayMystery(a2);` | `{11, 18, 9, 25}` |
| `int[] a3 = {5, 10, 20, 35, 40};`<br>`arrayMystery(a3);` | `{5, 15, 25, 35, 40}` |
| `int[] a4 = {-20, 20, 26, 32, 50, 3};`<br>`arrayMystery(a4);` | `{-15, 25, 31, 37, 50, 3}` |
| `int[] a5 = {5, 10, 16, 21, 27, 40, 55};`<br>`arrayMystery(a5);` | `{5, 10, 16, 26, 32, 45, 55}` |

## 3. Reference Semantics Mystery

```
2 6 112,122
2 7 112,122
4 6 119,129
4 7 119,129
```

## 3. Inheritance Mystery

```
skeletor-T    shera-A
shera-A
shera!

skeletor-T    shera-A    heman-A
shera-A    heman-A
heman!

shera-T
shera-A
shera!

battlecat-A    battlecat-T
battlecat-A
heman!
```

**4. File Processing (5 solutions shown)**

```
public static int frequentFlier(Scanner input) {
    int total = 0;
    while (input.hasNext()) {
        String cabin = input.next();
        int miles = input.nextInt();
        if (cabin.equals("coach")) {
            total += miles;
        } else if (cabin.equals("firstclass")) {
            total += miles * 2;
        }
    }
    return total;
}

public static int frequentFlier(Scanner input) {
    int miles = 0;
    while (input.hasNext()) {
        String kind = input.next();
        int mult = 0;
        if (kind.equals("firstclass")) {
            mult = 2;
        } else if (kind.equals("coach")) {
            mult = 1;
        }
        miles += (input.nextInt() * mult);
    }
    return miles;
}

public static int frequentFlier(Scanner input) {
    int total = 0;
    int count = 0;
    String kind = "";    // most recent kind of flight
    while (input.hasNext()) {
        if (count % 2 == 0) {
            kind = input.next();
        } else {
            int miles = input.nextInt();
            if (kind.equals("firstclass")) {
                total += 2 * miles;
            } else if (kind.equals("coach")) {
                total += 1 * miles;
            }
        }
        count++;
    }
    return total;
}
```

```
public static int frequentFlier(Scanner input) {
    int total = 0;
    while (input.hasNext()) {
        String type = input.next();
        if (type.equals("firstclass")) {
            total += 2 * input.nextInt();
        } else if (type.equals("coach")) {
            total += input.nextInt();
        } else {
            input.nextInt();   // throw it away
        }
    }
    return total;
}

public static int frequentFlier(Scanner input) {
    int firstClass = 0;
    int coach = 0;
    int discount = 0;
    while (input.hasNext()) {
        String type = input.next();
        if (type.equals("firstclass")) {
            firstClass += input.nextInt();
        } else if (type.equals("coach")) {
            coach += input.nextInt();
        } else {
            discount += input.nextInt();
        }
    }
    return 2 * firstClass + coach;
}
```

**5. Array Programming (5 solutions shown)**

```
public static int countCommon(String[] a1, String[] a2, String[] a3) {
    int common = 0;
    int minLength = Math.min(a1.length, Math.min(a2.length, a3.length));
    for (int i = 0; i < minLength; i++) {
        if (a1[i].equals(a2[i]) && a2[i].equals(a3[i])) {
            common++;
        }
    }
    return common;
}

public static int countCommon(String[] a1, String[] a2, String[] a3) {
    int common = 0;
    int minLength = a1.length;
    if (a2.length < minLength) {
        minLength = a2.length;
    }
    if (a3.length < minLength) {
        minLength = a3.length;
    }
    for (int i = 0; i < minLength; i++) {
        if (a1[i].equals(a2[i]) && a1[i].equals(a3[i]) && a2[i].equals(a3[i])) {
            common++;
        }
    }
    return common;
}

public static int countCommon(String[] a1, String[] a2, String[] a3) {
    int common = 0;
    for (int i = 0; i < a1.length; i++) {
        if (i < a2.length &&
             i < a3.length &&  // length tests must come first!
              a1[i].equals(a2[i]) &&
               a2[i].equals(a3[i])) {
            common++;
        }
    }
    return common;
}

public static int countCommon(String[] a1, String[] a2, String[] a3) {
    int common = 0;
    for (int i = 0; i < a1.length; i++) {
        if (i >= a2.length || i >= a3.length) {
            return common;
        }
        if (a1[i].equals(a2[i]) && a2[i].equals(a3[i])) {
            common++;
        }
    }
    return common;
}

public static int countCommon(String[] a, String[] b, String[] c) {
    int count = 0;
    for (int i = 0; i < a.length && i < b.length && i < c.length; i++) {
        if (a1[i].equals(b[i]) && a[i].equals(c[i])) {
            count++;
        }
    }
    return count;
}
```

**6. Array Programming (5 solutions shown)**

```
public static int[] delta(int[] a) {
    int[] result = new int[a.length + (a.length - 1)];
    for (int i = 0; i < result.length; i++) {
        if (i % 2 == 0) {
            result[i] = a[i / 2];
        } else {
            result[i] = a[i / 2 + 1] - a[i / 2];
        }
    }
    return result;
}
```

```
public static int[] delta(int[] a) {
    int[] b = new int[2 * a.length - 1];
    for (int i = 0; i < a.length; i++) {
        b[2 * i] = a[i];
    }
    for (int i = 1; i < b.length; i += 2) {
        b[i] = b[i + 1] - b[i - 1];
    }
    return b;
}
```

```
public static int[] delta(int[] a) {
    int[] b = new int[a.length + (a.length - 1)];
    for (int i = 0; i < a.length; i++) {
        b[2 * i] = a[i];
        if (i < a.length - 1) {
            b[2 * i + 1] = a[i + 1] - a[i];
        }
    }
    return b;
}
```

```
public static int[] delta(int[] a) {
    int[] result = new int[a.length + a.length - 1];
    int place = 0;
    for (int i = 0; i <= result.length - 1; i++) {
        if (i % 2 == 0) {
            result[i] = a[place];
            place++;
        } else {
            result[i] = a[place] - a[place - 1];
        }
    }
    return result;
}
```

```
public static int[] delta(int[] a) {
    int[] b = new int[2 * a.length - 1];
    b[0] = a[0];

    for (int i = 0; i < a.length - 1; i++) {
        b[2 * i] = a[i];
        b[2 * i + 1] = a[i + 1] - a[i];
        b[2 * i + 2] = a[i + 1];
    }
    return b;
}
```

**7. Critters (2 solutions shown)**

```
public class Daisy extends Critter {
    private boolean goingEast, isFull;
    private int fullSteps, totalSteps;

    public Daisy() {
        goingEast = true;
        isFull = false;
        fullSteps = 0;
        totalSteps = 0;
    }

    public boolean eat() {
        isFull = true;
        fullSteps = 0;
        return true;
    }

    public Direction getMove() {
        if (isFull) {
            fullSteps++;
            if (fullSteps > 6) {
                isFull = !isFull;
                fullSteps = 0;
            } else if (fullSteps % 2 == 1) {
                return Direction.CENTER;
            }
        }

        totalSteps++;

        if (totalSteps % 5 == 0) {
            goingEast = !goingEast;
            return Direction.NORTH;
        } else if (goingEast) {
            return Direction.EAST;
        } else {
            return Direction.WEST;
        }
    }
}

public class Daisy extends Critter {
    private int moves = 0;   // no constructor needed
    private int food = 0;

    public boolean eat() {
        food = 6;
        return true;
    }

    public Direction getMove() {
        if (food > 0) {
            food--;
            if (food % 2 == 1) {
                return Direction.CENTER;
            }
        }

        moves = moves % 10 + 1;
        if (moves <= 4) {
            return Direction.EAST;
        } else if (moves == 5 || moves == 10) {
            return Direction.NORTH;
        } else {
            return Direction.WEST;
        }
    }
}
```

**8. Objects (4 solutions shown)**

```
public boolean isBefore(Date d) {
    return month < d.month || (month == d.month && day < d.day);
}

public boolean isBefore(Date d) {
    return 1000 * month + day < 1000 * d.month + d.day;
}

public boolean isBefore(Date d) {
    if (month > d.getMonth()) {
        return false;
    } else if (month < d.getMonth()) {
        return true;
    } else if (day >= d.getDay()) {
        return false;
    } else {
        return true;
    }
}

public boolean isBefore(Date d) {
    int thisAbsDay = 1;
    Date temp1 = new Date(1, 1);     // compute my absolute day
    while (!temp1.equals(this)) {
        temp1.nextDay();
        thisAbsDay++;
    }

    int dAbsDay = 1;
    Date temp2 = new Date(1, 1);     // compute d's absolute day
    while (!temp2.equals(d)) {
        temp2.nextDay();
        dAbsDay++;
    }

    return thisAbsDay < dAbsDay;
}
```