

1. 11
13
15
2. Invalid, Valid, Valid, Invalid / true / false
3. No / Yes
4. Translation to Switch

```
switch(x) {
    case 2:
        z = 50;
        break;
    case 1:
    case 3:
    case 4:
    case 5:
        z = 17;
        break;
    default:
        z = 22;
        break;
}
```
5. $y = (x \leq 4 ? x*2 : 9-x);$
6. Code – typo in the question – it says implement the main and then it says the parameter is an int. That is not possible so I am implementing the method (not the main):

```
public static void starMeth(int size){
    public static void drawStars(int size){
        char[][] flag = new char[size][size];
        for (int row = size; row >= 1; row/=2){
            int col = 0;
            for (; col < row; col++){
                flag[size-row][col] = '*';
            }
            for (; col < size; col++){
                flag[size-row][col] = ' ';
            }
        }
        System.out.println("first printing:");
        for (int row = size; row >= 1; row /=2){
            for (int col = 0; col < size; col++){
                System.out.print(flag[size-row][col]);
            }
            System.out.println();
        }
    }
}
```

```

        System.out.println("second printing:");
        for (int row = size; row >= 1; row /=2){
            for (int col = size-1; col >=0; col--){
                System.out.print(flag[size-row][col]);
            }
            System.out.println();
        }
    }
}

```

7. Deep/ Reference/Shallow

getterTwo & getterThree

getterTwo

8. Answer:

```

public static boolean subset(String[] set1, String[] set2){
    /* returns true if and only if every element of set1 also appears in
    set2*/
    for (int index = 0; index < set1.length; index++){
        if(!isIn(set1[index], set2)){
            return false;
        }
    }
    return true;
}

```

```

/* returns true if the string given is contained in the set given or
returns false if the string given is not contained in the set given */
private static boolean isIn(String s, String[] set){
    for (int index = 0; index < set.length; index++){
        if (s.equals(set[index])){
            return true;
        }
    }
    return false;
}

```

9. Answer:

```

public static boolean properSubset(String[] set1, String[] set2){
    /* returns true if and only if every element of set1 also appears in
    set2 but there is at least one member of set2 which is not present in
    set1 */
    if (subset(set1, set2) && !subset(set2, set1)){
        return true;
    } else {
        return false;
    }
}

```

10. Answer:

In F
Done F
After 1
In F
AE caught
Finally Done
After All

11. Answer (I am including a driver also just so you can see how it would be used):

```
public class ArrayCleanupExample {
    private static int keepVals(int val, int[] a){
        int count = 0;
        for (int i = 0; i < a.length; i++){
            if (a[i] != val){
                count++;
            }
        }
        return count;
    }
    public static int cleanup(int[][] arr, int val){
        int removeCount = 0;
        for (int row = 0; row < arr.length; row++){
            int newSize = keepVals(val, arr[row]);
            int[] tempRow = new int[newSize];
            for (int i = 0, j = 0; i < arr[row].length; i++){
                if (arr[row][i] != val){
                    tempRow[j++] = arr[row][i];
                } else {
                    removeCount++;
                }
            }
            arr[row] = tempRow;
        }
        return removeCount;
    }
}

public static void main(String[] args){
    int[][] arr = {
        {2,4,6,3,4},
        {3,4},
        {4,1,4,7,4,8},
    };
    int removed = cleanup(arr, 4);
    System.out.println("after removing "+removed);
    for (int row = 0; row < arr.length; row++){
        for (int col = 0; col < arr[row].length; col++){
```

```

        System.out.print(arr[row][col]+" ");
    }
    System.out.println();
}
}
}

```

12. String is immutable while StringBuffer is not. They both hold sequences of characters and have many similar operations (i.e. for appending and for finding a character in that list), but because the String is immutable, if it is passed to a method and modified there the one given back to the caller will still not be modified while this is not true for the StringBuffer.

13. Objects of any class that "implements MyInter" can be referenced by the elements of the array. MyInter objects themselves can not be referenced by the elements of the array since MyInter objects can't exist. Objects of classes that don't have the words "implements MyInter" at the top can not be referenced by the elements of the array.

14. **instance data member** **element of an arrays**

15. Last one uses the following class

ShoppingList class:

```

public class Toy{
    private int modelNum;
    (other possible private data here)
    public Toy(){ ... }
    public Toy(int modNum){...}
    public int getModelNum(){... }
    (other methods defined here)
}

public class ShoppingList {
    private Toy[] list;
    public ShoppingList(){ ... }
    public int getSize(){...}
    public boolean noDuplicates();
    (other methods defined here)
}

a)
public int countModel(int val){
    int count = 0;
    for (int i = 0; i < list.length; i++){
        if (list[i].getModelNum() == val){
            count++;
        }
    }
    return count;
}

```

```

    }
b)
public void testGetCount(){
    ShoppingList temp = newShoppingList();
    Toy t = new Toy(5);
    Toy s = new Toy(9);
    temp.add(t);
    assertTrue(temp.countModel(5) == 1);
    assertTrue(temp.countModel(3) == 0);
    temp.add(s);
    assertTrue(temp.countModel(5) == 1);
    assertTrue(temp.countModel(3) == 0);
    temp.add(t);
    assertTrue(temp.countModel(5) == 2);
    assertTrue(temp.countModel(3) == 0);
}
c)
public boolean noDuplicates(){
    for (int i = 0; i < list.length; i++){
        int currModel = list[i].getModelNum();
        if (countModel(currModel) > 1){
            return false;
        }
    }
    return true;
}
d)
public void testGetCount(){
    ShoppingList temp = newShoppingList();
    Toy t = new Toy(5);
    Toy s = new Toy(9);
    temp.add(t);
    assertTrue(temp.noDuplicates());
    temp.add(s);
    assertTrue(temp.noDuplicates());
    temp.add(t);
    assertFalse(temp.noDuplicates());
}
e)
public int removeBad(int n){
    if (list == null){
        return 0;
    } else {
        int badCount = 0;
        for (int i = 0; i < list.length; i++){
            if (list[i].getModelNum() == n){
                badCount++;
            }
        }
        int goodCount = list.length - badCount;
        Toy[] temp = new Toy[goodCount];
        int newI = 0;

```

```

        for (int i = 0; i < list.length; i++){
            if (list[i].getModelNum() != n){
                temp[newI++] = list[i];
            }
        }
    }
    return badCount;
}

f)
public Toy[][] getMasterList(int[] modNums, int quantArr){
    if (modNums == null || quantArr == null){
        return null;
    } else {
        int size1 = modNums.length;
        Toy[][] temp = new Toy[size1][];
        for (int mod=0; mod < size1; mod++){
            int size2 = quantArr[mod];
            temp[mod] = new Toy[size2];
            int model = modNums[mod];
            for (int i = 0; i < size2; i++){
                temp[mod][i] = new Toy(model);
            }
        }
        return temp;
    }
}

```