Solutions – Part 1, Version A (AA, AB, AC)

1. Expressions

```
9
10.0
2.5
6rural13juror
false
```

2. Array Simulation

```
[2, 3, 4]
[-1, 1, 2]
[-1, 1, 0, 4, 2, 6]
[-2, 4, 3, 3, 1]
[-1, -1, -2, -4, -7, -11]
```

3. Inheritance Mystery

```
leela 1
bender 1
leela 2

leela 1
zoidberg 1
zoidberg 2

fry
bender 1
bender 2

bender
bender 1
bender 2
```

4. **reportMax** - many solutions possible, the most common is shown below:

```
public static void reportMax(Scanner input) {
   int max = input.nextInt();
   while (input.hasNext()) {
      if (input.hasNextInt()) {
         max = Math.max(max, input.nextInt());
      } else {
         input.next();
         System.out.println("max so far: " + max);
      }
   }
   System.out.println("max: " + max);
}
```

Solutions – Part 1, Version B (AD, AE, AF, AG, AH)

1. Expressions

```
-1
10.0
2.0
13rural28juror
true
```

2. Array Simulation

```
[2, 3, 4]
[1, -1, 2]
[1, 1, 5, 2, 5, 1]
[-2, -2, 1, 2, 2]
[-1, -1, -2, -3, -5, -7]
```

3. Inheritance Mystery

```
michael
michael 1
lucille 2

buster
michael 1
buster 2

gob
gob 1
michael 2

michael
michael 1
michael 2
```

4. **scoreGame** - many solutions possible, the most common is shown below:

```
public static int scoreGame(Scanner input) {
    int sum = 0;
    while (input.hasNext()) {
        if (input.hasNextInt()) {
            sum += input.nextInt();
        } else {
            input.next();
            sum -= 10;
        }
    }
    return sum;
}
```

5. **hasSameParity/allEvenGaps** - the two problem 5s were technically the same question, but with different names and descriptions. However, the different descriptions tended to lead to different solutions. Here is the most common solution we saw for each wording:

```
public static boolean hasSameParity(int[] arr) {
   for (int i = 1; i < arr.length; i++) {
       if (Math.abs(arr[0]) % 2 !=
              Math.abs(arr[i]) % 2) {
           return false;
       }
   }
   return true;
}
```

```
public static boolean allEvenGaps(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        if ((arr[i] - arr[i - 1]) % 2 != 0) {
            return false;
        }
    }
    return true;
 }
```

6. **removeDuplicates** - many solutions possible, the most common ones shown below:

```
p s v removeDuplicates(ArrayList<Integer> list) {
    for (int i = 0; i < list.size() - 1; i++) {
        if (list.get(i) == list.get(i + 1)) {
            list.remove(i);
            i--;
        }
    }
}
```

```
p s v removeDuplicates(ArrayList<Integer> list){
    int i = 0;
    while (i < list.size() - 1) {
        if (list.get(i) == list.get(i + 1)) {
            list.remove(i);
        } else {
            i++;
        }
    }
}
```

```
p s v removeDuplicates(ArrayList<Integer> list) {
    for (int i = list.size() - 1; i > 0; i--) {
        if (list.get(i) == list.get(i - 1)) {
            list.remove(i);
        }
    }
}
```

6. **removeSmallerValues** - many solutions possible, the most common ones shown below:

```
p s v removeSmallerValues(ArrayList<Integer> list){
    int times = list.size() / 2;
    for (int i = 0; i < times; i++){ // or list.size()-1
        int first = list.get(i);
        int second = lest.get(i + 1);
        if (first <= second) {
            list.remove(i);
        } else {
            list.remove(i + 1);
        }
    }
}
```

Solutions – Part 2

7. **reportLongLines** – most solutions followed this general pattern:

```
public static void reportLongLines(Scanner input, int maxLength) {
    int lineCount = 0;
    int longLineCount = 0;
    while (input.hasNextLine()) {
        lineCount++;
        String line = input.nextLine();
        if (line.length() > maxLength) {
            longLineCount++;
            System.out.println("Line " + lineCount + " excess text: " + line.substring(maxLength));
        }
    }
    if (longLineCount == 0) {
        System.out.print("No");
    } else {
        System.out.print(longLineCount)
    }
    System.out.println(" long lines reported");
}
```

8. **copyRanges** – most solutions followed this general pattern:

```
public static int[] copyRanges(int[] values, int[] ranges) {
    int size = 0;
    for (int i = 0; i < ranges.length; i += 2) {
        size += (ranges[i + 1] - ranges[i]);
    }
    int[] result = new int[size];

    int resultIdx = 0;
    for (int i = 0; i < ranges.length; i += 2) {
        int start = ranges[i];
        int end = ranges[i + 1];
        for (int j = start; j < end; j++) {
            result[resultIdx] = values[j];
            resultIdx++;
        }
    }
    return result;
}
```

9. **Narwhal** – most solutions followed this general pattern:

```
public class Narwhal extends Critter {
    private String direction;
    private int count;

    public Narwhal() {
        direction = "o";
        count = -1;
    }

    public Action getMove(CritterInfo info) {
        if (info.getFront() == Neighbor.OTHER) {
            return Action.INFECT;
        } else {
            count++;
            if (count % 5 < 3) {
                return Action.HOP;
            } else {
                if (info.getDirection() == Direction.NORTH) {
                    direction = ">";
                } else if (info.getDirection() == Direction.SOUTH) {
                    direction = "<";
                } else if (info.getDirection() == Direction.EAST) {
                    direction = "v";
                } else {
                    direction = "^";
                }
                return Action.RIGHT;
            }
        }
    }

    public String toString() {
        return direction;
    }

    public Color getColor() {
        return Color.CYAN;
    }
}
```

10. **printReversed** – Many solutions possible, the most common ones we saw shown below:

```
public static void printReversed(String str) {
    for (int i = 0; i < str.length(); i++) {
        if (str.charAt(i) == ' ') {
            System.out.print(" ");
        } else {
            int j = i;
            while (j < str.length() && str.charAt(j) != ' ') {
                j++;
            }
            for (int k = j - 1; k >= i; k--) {
                System.out.print(str.charAt(k));
            }
            i = j;
        }
    }
    System.out.println();
}
```

```
public static void printReversed(String str) {
    Scanner input = new Scanner(str);
    for (int i = 0; i < str.length(); i++) {
        if (str.charAt(i) == ' ') {
            System.out.print(" ");
        } else {
            String word = input.next();
            for (int j = word.length() - 1; j >= 0; j--) {
                System.out.print(word.charAt(j));
            }
            i += word.length() - 1;
        }
    }
    System.out.println();
}
```

```
public static void printReversed(String str) {
    String build = "";
    for (int i = 0; i < str.length(); i++) {
        if (str.charAt(i) == ' ') {
            if (build.length() != 0) {
                System.out.print(build);
                build = "";
            }
            System.out.print(" ");
        } else {
            build = str.charAt(i) + build;
        }
    }
    System.out.println(build);
}
```