Key to Practice Final Self Assessment
Summer 2020

1. The program produces the following output:

```
108 [5, 10, 14]
8 [5, 10, 14]
20 8
```

2.
```
Original Array              Final Array
--------------              ----------------------------
[2, 3, 1]                   [2, 3, 1]
[2, 6, 3, 5]                [2, 6, 4, -2]
[3, 3, 7, 9]                [3, 3, 0, 9]
[2, 4, 5, 6, 8]             [2, 4, 2, 6, 4]
[1, 5, 8, 4, 10, 9]         [1, 5, 4, -1, 10, 11]
```

3. The program produces the following output:

```
Eins
Eins 1
Zwei 2 Eins 1

Eins
Eins 1
Eins 2

Vier Drei
Drei 1
Zwei 2 Drei 1 Vier 2

Drei
Drei 1
Zwei 2 Drei 1
```

4. Two possible solutions:

```
public static void split(ArrayList<Integer> list) {
   for (int i = 0; i < list.size(); i += 2) {
      int n = list.get(i);
      list.set(i, n / 2 + n % 2);
      list.add(i + 1, n / 2);
   }
}

public static void split(ArrayList<Integer> list) {
   for (int i = 0; i < list.size(); i += 2) {
      int n = list.get(i);
      list.remove(i);
      list.add(i, n / 2 + n % 2);
      list.add(i + 1, n / 2);
   }
}
```

5. Two possible solutions:

```
public static void formatList(Scanner input) {
   while (input.hasNextLine()) {
      String text = input.nextLine();
      if (!text.startsWith(".")) {
         System.out.println(text);
      } else {
         while (text.startsWith(".")) {
            System.out.print("    ");
            text = text.substring(1);
         }
         System.out.println("* " + text);
      }
   }
}

public static void formatList(Scanner input) {
   while (input.hasNextLine()) {
      String line = input.nextLine();
      int indent = 0;
      while (indent < line.length() && line.charAt(indent) == '.') {
         indent++;
      }
      line = line.substring(indent);
      for (int i = 0; i < indent * 4; i++) {
         System.out.print(" ");
      }
      if (indent > 0) {
         System.out.print("* ");
      }
      System.out.println(line);
   }
}
```

6. One possible solution:

```
public static void calculateGrade(Scanner file) {
   int hwTot = 0;
   int hwPoss = 0;
   int examTot = 0;
   int examPoss = 0;

   while (file.hasNext()) {
      String type = file.next();
      int score = file.nextInt();
      int poss = file.nextInt();

      if (type.equalsIgnoreCase("Homework")) {
         hwTot += score;
         hwPoss += poss;
      } else if (type.equalsIgnoreCase("Exam")) {
         examTot += score;
         examPoss += poss;
      }
   }

   double hwPct = (double) hwTot / hwPoss * 100;
   double examPct = (double) examTot / examPoss * 100;
   System.out.println("Homeworks: " + hwTot + " / " + hwPoss + " = " + hwPct);
   System.out.println("Exams: " + examTot + " / " + examPoss + " = " + examPct);
   System.out.println("Overall grade: " + ((hwPct + examPct) / 2));
}
```

7. Three possible solutions:

```
public static boolean sweep(int[] a) {
   boolean changed = false;
   for (int i = 0; i < a.length - 1; i++) {
      if (a[i] > a[i + 1]) {
         int temp = a[i];
         a[i] = a[i + 1];
         a[i + 1] = temp;
         changed = true;
      }
   }
   return changed;
}

public static boolean sweep(int[] a) {
   int count = 0;
   for (int i = 0; i < a.length - 1; i++) {
      if (a[i] > a[i + 1]) {
         int temp = a[i];
         a[i] = a[i + 1];
         a[i + 1] = temp;
         count++;
      }
   }

   if (count > 0) {
      return true;
   } else {
      return false;
   }
}

public static boolean sweep(int[] a) {
   int count = 0;
   for (int i = 1; i < a.length; i++) {
      int temp1 = a[i];
      int temp2 = a[i - 1];

      if (a[i - 1] > a[i]) {
         a[i - 1] = temp1;
         a[i] = temp2;
         count++;
      }
   }

   if (count > 0) {
      return true;
   } else {
      return false;
   }
}
```

8. One possible solution:

```
    public class Sponge extends Critter {
       private int dashCount;
       private int turnCount;

       public Sponge() {
          dashCount = 1;
          turnCount = 0;
       }

       public Action getMove(CritterInfo info) {
          if (info.getFront() == Neighbor.OTHER) {
             dashCount++;
             return Action.INFECT;
          } else if (info.getFront() == Neighbor.EMPTY) {
             return Action.HOP;
          } else {
             dashCount = Math.max(dashCount - 1, 1);
             turnCount++;
             if (turnCount % 3 == 1) {
                return Action.LEFT;
             } else {
                return Action.RIGHT;
             }
          }
       }

       public Color getColor() {
          return Color.YELLOW;
       }

       public String toString() {
          String result = "[";
          for (int i = 0; i < dashCount; i++) {
             result = result + "-";
          }
          return result + "]";
       }
    }
```

9. Two possible solutions:

```
public static int[] maxes(int[] arr1, int[] arr2) {
   int length = Math.max(arr1.length, arr2.length);
   int[] result = new int[length];

   for (int i = 0; i < length; i++) {
      if (i >= arr1.length) {
         result[i] = arr2[i];
      } else if (i >= arr2.length) {
         result[i] = arr1[i];
      } else if (arr1[i] > arr2[i]) {
         result[i] = arr1[i];
      } else {
         result[i] = arr2[i];
      }
   }

   return result;
}

public static int[] maxes(int[] arr1, int[] arr2) {
   int[] result = new int[Math.max(arr1.length, arr2.length)];
   int shorter = Math.min(arr1.length, arr2.length);

   for (int i = 0; i < shorter; i++) {
      if (arr1[i] > arr2[i]) {
         result[i] = arr1[i];
      } else {
         result[i] = arr2[i];
      }
   }

   for (int i = shorter; i < result.length; i++) {
      if (i >= arr1.length) {
         result[i] = arr2[i];
      } else {
         result[i] = arr1[i];
      }
   }

   return result;
}
```

10. Four possible solutions:

```
      public static String undouble(String s) {
         if (s.length() <= 1) {
            return s;
         }

         String res = "";
         for (int i = 0; i < s.length() - 1; i++) {
            char next = s.charAt(i);
            if (next == s.charAt(i + 1)) {
               i++;
            }
            res += next;
         }

         if (s.length() > 1) {
            if (s.charAt(s.length() - 1) != s.charAt(s.length() - 2)) {
               res += s.charAt(s.length() - 1);
            }
         }
         return res;
      }
      public static String undouble(String s) {
         String result = "";
         if (s.length() > 0) {
            char last = s.charAt(0);
            result += last;
            for (int i = 1; i < s.length(); i++) {
               if (s.charAt(i) != last) {
                  result += s.charAt(i);
               }
               last = s.charAt(i);
            }
         }
         return result;
      }

      public static String undouble(String str) {
         if (str.length() == 0) {
            return "";
         }
         String result = "";
         char prev = str.charAt(0);
         result += prev;

         for (int i = 1; i < str.length(); i++) {
            char curr = str.charAt(i);
            if (curr != prev) {
               result += curr;
            }
            prev = curr;
         }

         return result;
      }
      public static String undouble(String s) {
         for (int i = 0; i < s.length() - 1; i++) {
            if (s.charAt(i) == s.charAt(i + 1)) {
               s = s.substring(0, i) + s.substring(i + 1);
            }
         }
         return s;
      }
```