1. WordMatch
   1. scoreGuess

| public int scoreGuess (String guess) {  int numOfSubstringOccurrences = 0;  int squareOfLengthOfGuess = guess.length() \* guess.length();  int endValue = 0;  for (int i = 0; i < guess.length(); i++) {  if (secret.substring(i, i + guess.length()).equals(guess.substring(0, guess.length())) {  numOfSubstringOccurrences++;  }  else {  break;  }  }  this.endValue = numOfSubstringOccurences \* squareOfLengthOfGuess;  return endValue;  } |
| --- |

* 1. findBetterGuess

| public String findBetterGuess (String guess1, String guess2) {  int guess1Score = scoreGuess(guess1);  int guess2Score = scoreGuess(guess2);  if (guess1Score > guess2Score) {  return guess1Score;  }  else if (guess2Score > guess1Score) {  return guess2Score;  }  else if (guess1Score = guess2Score) {  if(guess1.compareTo(guess2) > 0) {  return guess1Score;  }  else{  return guess2Score;  }  }  } |
| --- |

1. SingleTable

| private int seats;  private int t1Height;  private int t2Height;  private int t1ViewQuality;  private int t2ViewQuality;  public CombinedTable (SingleTable t1, SingleTable t2) {  seats = t1.getNumSeats() + t2.getNumSeats() - 2;  t1Height = t1.getHeight();  t2Height = t2.getHeight();  t1ViewQuality = t1.getViewQuality();  t2ViewQuality = t2.getViewQuality();  }  public void canSeat (int people) {  people > seats ? return false: return seats;  }  public void getDesirability () {  if (t1Height == t2Height) {  return ((t1ViewQuality + t2ViewQuality) / 2);  }  else if (t1Height > t2Height || t2Height > t1Height) {  return ((t1ViewQuality + t2ViewQuality) / 2) - 10;  }  } |
| --- |

1. ClubMembers
   1. addMembers

| public void addMembers (String[] names, int gradYear) {  for (int i = 0, i <= names.size(); i++) {  memberList.add(MemberInfo(names[i], gradYear, true));  }  } |
| --- |

* 1. removeMembers

| public ArrayList < MemberInfo> removeMembers (int year) {  ArrayList<MemberInfo> graduates = new ArrayList();  for (int i = 0; i <= memberList.size(); i++) {  if(memberList[i].getGradYear() < year && memberList[i].inGoodStanding) {  graduates.add(memberList[i]);  }  membersList.remove(i);  i--;  }  return graduates;  } |
| --- |

1. ArrayResizer
   1. isNonZeroRow

| public static boolean isNonZeroRow(int[][] array2D, int r) {  int count = 0;  for (int j = 0; j <= array2D[r].length;j++) {  if (array2D[r][j] != 0) {  count++;  }  }  if(count == array2D[r].length) {  return true;  }  else{  return false;  }  } |
| --- |

* 1. resize

| public static int[][] resize( int[][] array2D) {  int[][] smaller = new int[array2D.numNonZeroRows][array2D[0].length]  for (int i = 0; i < array2D.length; i++) {  if (array2D[i].isNonZeroRow(array2D, i)) {  for (int j = 0; j < array2D[j].length; j++) {  for (int h = 0; h < smaller.length; h++) {  smaller[h][j] = array2D[i][j];  }  }  }  }  return smaller;  } |
| --- |