Report on s1709906 Inf1OP 2017 PE1 Programming Exam

Generated by Automarker on May 07, 2018

Question 1

Submitted Files: OK
Compiling Entropy.java alone: OK
Compiling Entropy.java with basic tests provided in the exam: OK
Running Basic Tests: All passed
Compiling Entropy.java with main tests: OK
Running Main Tests: Passed 26 of 27 main tests
Note: Disabled unnecessary basic test testJP for method jointProb.

Issue: 1b) Doesn't handle null. testNormalise0a failed. Impact: −1
Q1: 49 / 50

Question 2

Submitted Files: OK
Compiling FruitySmartPhone.java alone: OK
Compiling FruitySmartPhone.java with basic tests provided in the exam: OK
Running Basic Tests: All passed
Compiling FruitySmartPhone.java with main tests: OK
Running Main Tests: Passed 33 of 34 main tests

Issue: 2d) Lost track of memory usage. testInstallUseUninstallBook failed. Impact: −1
Q2: 49 / 50

Total Marks: 98/100

Submitted Files

Submitted: Entropy.java

```
import java.util.Arrays;
import java.util.HashMap;
import java.util.Hashtable;

public class Entropy {
 public static int[] charCount(String s) {

  if (s.equals(null) || s.isEmpty()) {
   return null;
  }

  char[] cs = s.toCharArray();
  Arrays.sort(cs);

  HashMap<Character,Integer> map = new HashMap<Character,Integer>();

  for (char c : cs) {
   if (!map.containsKey(c)) {
    map.put(c, 0);
   }
   map.put(c, map.get(c) + 1);

  }

  int[] result = new int[map.size()];
  int i = 0;
  for (int n : map.values()) {
   result[i] = n;
   i++;
  }

  return result;
 }

 public static double[] normalise(int[] c) {
  double[] p = new double[c.length];

  int sum = 0;

  for (int i = 0; i < c.length; i++) {
   sum += c[i];
  }

 // System.out.println(sum);
  //System.out.println(Arrays.toString(c));

  for (int i = 0; i < p.length; i++) {
   p[i] = (double) c[i]/sum;
  }

  return p;
 }

 public static double entropyOf(double[] p) {
  double result = 0;
  for (int i = 0; i < p.length; i++) {
```

```
   result = result - p[i]*Math.log(p[i]);
  }
  return result;
 }

 public static boolean containsChar(String s, char c) {
  for (int i = 0; i < s.length(); i++) {
   if (s.charAt(i) == c) {
    return true;
   }
  }
  return false;
 }

 public static String remove(String s, char c) {
  String rem = "" + c;
  String output = s.replaceAll(rem, "");
  return output;
 }

 public static int[][] charCountArray(String[] a) {
  int[] letters = new int[26];

  for ( int i = 0; i < a.length; i++) {
   for (char c = 'a'; c <= 'z'; c++) {
    if (containsChar(a[i], c)) {
     letters[c - 'a']++;
    }
   }
  }

  for (char c = 'a'; c <= 'z'; c++) {
   if (letters[c-'a'] > 1) {
    for (int i = 0; i < a.length; i++) {
     a[i] = remove(a[i],c);
    }
   }
  }

  int[][] targetArray = new int[a.length][];

  for (int i = 0; i < a.length; i++) {
   targetArray[i] = charCount(a[i]);
  }

  return targetArray;
 }



 public static void main(String[] args) {
  System.out.println("Character␣Probabilities␣in␣" + args[0] + "␣:␣" + Arrays.
 toString(normalise(charCount(args[0]))));
  System.out.println("Entropy␣of␣" + args[0] + "␣:␣" + entropyOf(normalise(
 charCount(args[0]))));
```

```
  System.out.println("Entropy␣of␣" + args[1] + "␣:␣" + entropyOf(normalise(
 charCount(args[1]))));
  String[] a = new String[args.length];
  for (int i = 0; i < args.length; i++) {
   a[i] = args[i];
  }
  int[][] counted = charCountArray(a);
  for (int i = 0; i < counted.length; i++) {
   System.out.println("Entropy␣of␣unique␣chars␣in␣" + args[i] + "␣:␣" +
 entropyOf(normalise(counted[i])));
  }
 }
}
```

Submitted: FruitySmartPhone.java

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;
import java.util.TreeMap;

class FruitySmartPhone extends Phone {

 private HashMap<String, Integer> installedApps = new HashMap<String, Integer
 >();
 private int cpuSpeed;
 private int freeMemory;

 public int getCpuSpeed() {
  return cpuSpeed;
 }

 public int getFreeMemory() {
  return freeMemory;
 }

 public FruitySmartPhone(String name, int freeMemory, int cpuSpeed) {
  // TODO Auto-generated constructor stub
  super(name);
  this.freeMemory = freeMemory;
  this.cpuSpeed = cpuSpeed;
 }

 public boolean installApp(App app) {
  // you can't install what is already there
  boolean a = installedApps.containsKey(app.getName());
  boolean b = app.getMemoryReq() > this.getFreeMemory();
  boolean c = app.getCpuReq() > this.getCpuSpeed();

  if (a || b || c) {
   return false;
  }

  installedApps.put(app.getName(), app.getMemoryReq());
  freeMemory = freeMemory - app.getMemoryReq();

```

```
  return true;
 }

 public boolean uninstallApp(App app) {
  // you can't uninstall what isn't there
  boolean a = installedApps.containsKey(app);

  if (!a) {
   return false;
  }

  installedApps.remove(app.getName());
  freeMemory = freeMemory + app.getMemoryReq();
  return true;
 }

 public boolean useApp(String s) {
  if (!installedApps.containsKey(s) || freeMemory < 1) {
   return false;
  }

  freeMemory = freeMemory - 1;
  installedApps.put(s, installedApps.get(s) + 1);
  return true;
 }



 public ArrayList<String> getInstalledApps() {
  ArrayList<String> apps = new ArrayList<String>();
  ArrayList<Integer> values = new ArrayList<Integer>(installedApps.values());
  Collections.sort(values);
  Collections.reverse(values);

  for (int v : values) {
   for (String s : installedApps.keySet()) {
    if (apps.contains(s) == false && installedApps.get(s) == v) {
     apps.add(s);
    }
   }
  }

  return apps;
 }

 public static void main(String[] args) {
  FruitySmartPhone p = new FruitySmartPhone("John Smith", 100, 10);
  p.installApp(new App("Camera",1,2));
  p.installApp(new App("Music",2,3));
  p.installApp(new App("Podcast",6,4));
  p.useApp("Music");
  System.out.println(p.getInstalledApps());
 }

}
```