My Project

AUTHOR
Version
10/04/2017

Table of Contents

Table of contents

Data Type Index

Design Unit Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Data Type Index

Data Types List

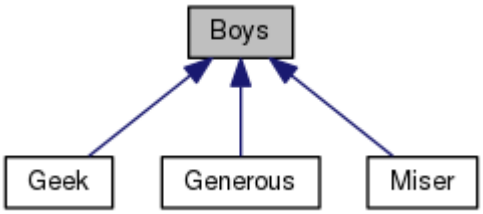
Here are the data types with brief descriptions:

Data Type Documentation

Boys Class Reference

Abstract Boys class.

Inheritance diagram for Boys:

IMAGE 

Package Functions

Package Attributes

Detailed Description

Abstract Boys class.

Member Function Documentation

void happiness (int *value*)[package]

Function to calculate happiness of a Boy.

```
23                                  {
24
25          this.happiness_boy = value;
26      }
```

boolean is_eligible (int *expense*, int *attar*)[package]

Function to check eligiblity of a Boy.

```
14                                                  {
15          if (this.budget >= expense && this.min_attr_req <= attar &&
this.status.equals("Single")){
16              return true;
17          }else{
18              return false;
19          }
20
21          }
```

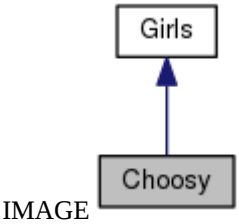
The documentation for this class was generated from the following file:

0 Boys.java

Choosy Class Reference

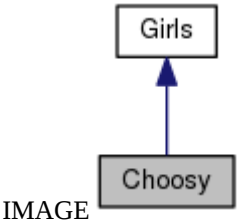
Type of Girls.

Inheritance diagram for Choosy:



IMAGE

Collaboration diagram for Choosy:



IMAGE

Package Functions

Additional Inherited Members

Detailed Description

Type of Girls.

Member Function Documentation

double happyness (Boys *boy*, gift *gf*[], int *k*)[package]

Function which calculates happiness of Choosy Girls.

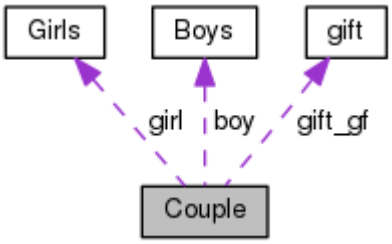
```
 4                                                    {
 5
 6     int i = 0,n = 0,sum = 0;
 7     double happy;
 8     for(i=0;i<n;i++){
 9             sum += gf[i].gift_price;
10             if(gf[i].gift_type.equals("Luxury")){
11                     sum += 2*(gf[i].gift_value);
12                 }
13             else{
14                 sum = sum + gf[i].gift_value;
15             }
16     }
17
18         happy = Math.log(sum - this.expense);
19     return happy;
20     }
```

The documentation for this class was generated from the following file:

1 Choosy.java

Couple Class Reference

Collaboration diagram for Couple:



IMAGE

Package Functions

Package Attributes

Detailed Description

The documentation for this class was generated from the following file:

2 Couple.java

---

csv_gen Class Reference

csv_gen Class to generate csv files

Package Functions

Detailed Description

csv_gen Class to generate csv files

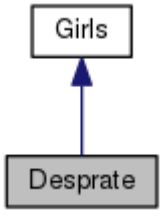
The documentation for this class was generated from the following file:

3 csv_gen.java

Desprate Class Reference

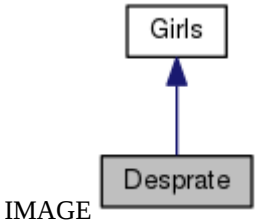
Type of Girls.

Inheritance diagram for Desprate:



IMAGE

Collaboration diagram for Desprate:



IMAGE

Package Functions

Package Attributes

Detailed Description

Type of Girls.

Member Function Documentation

double happyness (Boys *boy*, gift *gf*[], int *n*) [package]

Function which calculates happiness of Desparate Girls.

```
 5                                                    {
 6
 7         int sum = 0,i = 0;
 8         for(i=0;i<n;i++){
 9                 sum +=gf[i].gift_price;
10         }
11         if(sum > this.expense){
12             happy = Math.exp(sum - this.expense);
13         }
14         else{
15             happy = 0;
16         }
17         return happy ;
18 }
```

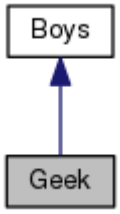
The documentation for this class was generated from the following file:

4 Desprate.java

Geek Class Reference

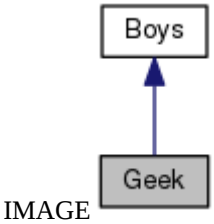
Type of a Boys.

Inheritance diagram for Geek:



IMAGE

Collaboration diagram for Geek:



IMAGE

Package Functions

Additional Inherited Members

Detailed Description

Type of a Boys.

Member Function Documentation

double happyness (Girls *gl*, double *sum*, double *happy_girl*)[package]

Function to calculate happiness of Geek boys.

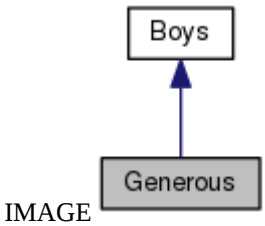
```
5     {
6         double happy;
7         happy = gl.intelligence;
8         return happy;
9     }
```

The documentation for this class was generated from the following file:

5 Geek.java

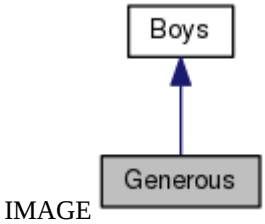
Generous Class Reference

Inheritance diagram for Generous:



IMAGE

Collaboration diagram for Generous:



IMAGE

Package Functions

Additional Inherited Members

Detailed Description

Member Function Documentation

double happyness (Girls *gl*, double *sum*, double *happy_girl*)[package]

Function which calculates happiness of Generous boys.

```
5     {
6         double happy;
7         happy = happy_girl;
8         return happy;
9     }
```

The documentation for this class was generated from the following file:

6 Generous.java

---

gift Class Reference

gift class

Package Attributes

Detailed Description

gift class

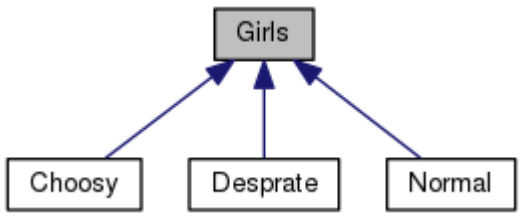
The documentation for this class was generated from the following file:

7 gift.java

Girls Class Reference

Inheritance diagram for Girls:

IMAGE



Package Functions

Package Attributes

Detailed Description

Member Function Documentation

abstract double happyness (Boys *boy*, gift *gf*[], int *n*)[abstract], [package]

Function to calculate happiness of a Girl.

boolean is_eligible (int *budget*)[package]

Function to calculate eligibility of a Girl.

```
15                                      {
16      if (this.expense <= budget) {
17              return true;
18          }
19      else {
20              return false;
21          }
22      }
```

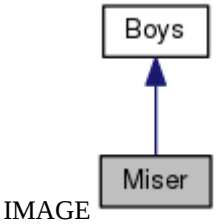
The documentation for this class was generated from the following file:

8 Girls.java

Miser Class Reference

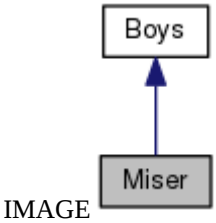
Type of a Boys.

Inheritance diagram for Miser:



IMAGE

Collaboration diagram for Miser:



IMAGE

Package Functions

Additional Inherited Members

Detailed Description

Type of a Boys.

The documentation for this class was generated from the following file:
9 Miser.java

MyLogger Class Reference

MyLogger class to Generate log file.

Static Package Functions

Detailed Description

MyLogger class to Generate log file.

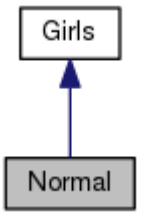
The documentation for this class was generated from the following file:
10 MyLogger.java

Normal Class Reference

Type of Girls.

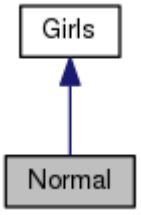
Inheritance diagram for Normal:

IMAGE



Collaboration diagram for Normal:

IMAGE



Package Functions

Package Attributes

Detailed Description

Type of Girls.

Member Function Documentation

double happyness (Boys *boy*, gift *gf*[], int *n*) [package]

Function which calculates happiness of Normal Girls.

```
 5                                                      {
 6
 7         int sum = 0,i = 0;
 8         for(i=0;i<n;i++){
 9                 sum = sum +gf[i].gift_price+gf[i].gift_value;
10         }
11         if(sum > this.expense){
12             happy = sum - this.expense;
13         }
14         else{
15             happy = 0;
16         }
17         return happy;
18     }
```

The documentation for this class was generated from the following file:

11 Normal.java

---

q3 Class Reference

Main Class.

Static Public Member Functions

Detailed Description

Main Class.

Member Function Documentation

static void main (String *args*[]) throws IOException [static]

Select between types of Boys

Select between types of Girls

```
11                                                                  {
12         int i,j;
13         i=0;
14         j=0;
15         int m=0;
16         String str = "";
17         Boys b[] = new Boys[1000];
18         csv_gen csv_files = new csv_gen();
19         csv_files.csv();
```

```
20          String csvFile = "boy.csv";
21          File out=new File("log.log");
22          String line = "";
23          String csvSplit = ",";
24          BufferedReader buff = null;
25          try{
26              buff = new BufferedReader(new FileReader(csvFile));
27              while((line = buff.readLine() )!= null){
28                  String[] boys_table = line.split(csvSplit);
29
30                  //b[i] = new Boys();
31                  str = boys_table[6];
32                  //  System.out.println(str);
33                      switch(str) {
35                      case "Geeky":
36                          b[i] = new Geek();
37                          break;
38                      case "Miser":
39                          b[i] = new Miser();
40                          break;
41                      case "Generous":
42                          b[i] = new Generous();
43                          break;
44                      }
45                  b[i].name = boys_table[0];
46                  b[i].intelligence = Integer.parseInt(boys_table[1]);
47                  b[i].attractivness = Integer.parseInt(boys_table[2]);
48                  b[i].min_attr_req = Integer.parseInt(boys_table[3]);
49                  b[i].budget = Integer.parseInt(boys_table[4]);
50                  b[i].status = boys_table[5];
51                  b[i].boy_type= boys_table[6];
52                  b[i].girlf="";
53                  b[i].happiness_boy = 0;
54                  i++;
55              }
56          }
57
58          catch(FileNotFoundException e){
59              e.printStackTrace();
60          }catch(IOException e){
61              e.printStackTrace();
62          }
63
64          Girls g[] = new Girls[1000];
65          csvFile = "girl.csv";
66          line = "";
67          csvSplit = ",";
68          buff = null;
69
70
71          try{
72              buff = new BufferedReader(new FileReader(csvFile));
73              while((line = buff.readLine() )!= null){
74                  String[] girls_table = line.split(csvSplit);
75
76                  str = girls_table[5];
77              //  System.out.println(str);
78                  switch(str) {
80                  case "Choosey":
81                      g[j] = new Choosy();
82                      break;
83                  case "Normal":
84                      g[j] = new Normal();
85                      break;
86                  case "Desprate":
87                      g[j] = new Desprate();
88                      break;
89                  }
90                  g[j].name = girls_table[0];
91                  g[j].attar = Integer.parseInt(girls_table[1]);
92                  g[j].expense = Integer.parseInt(girls_table[2]);
93                  g[j].intelligence = Integer.parseInt(girls_table[3]);
94                  g[j].status = girls_table[4];
95                  g[j].girl_type = girls_table[5];
96                  g[j].boyf = "";
97                  g[j].happiness_girl= 0;
98
```

```
 99                 j++;
100             }
101         }
102         catch(FileNotFoundException e){
103             e.printStackTrace();
104         }catch(IOException e){
105             e.printStackTrace();
106         }finally{
107                 if(buff != null){
108                     try{
109                         buff.close();
110                     }catch(IOException e){
111                     e.printStackTrace();
112                     }
113                 }
114         }
115 
116 
117         gift gf[] = new gift[1000];
118         csvFile = "gift.csv";
119         line = "";
120         csvSplit = ",";
121         buff = null;
122 
123 
124         try{
125             buff = new BufferedReader(new FileReader(csvFile));
126             while((line = buff.readLine() )!= null){
127                 String[] gifts_table = line.split(csvSplit);
128             //  System.out.println(boy[0] + boy[1] + boy[2] + boy [3]);
129                 gf[m] = new gift();
130                 gf[m].gift_name = gifts_table[0];
131                 gf[m].gift_price = Integer.parseInt(gifts_table[1]);
132 
133                 gf[m].gift_value = Integer.parseInt(gifts_table[2]);
134                 gf[m].gift_type = gifts_table[3];
135 
136                 m++;
137 
138             }
139         }
140         catch(FileNotFoundException e){
141             e.printStackTrace();
142         }catch(IOException e){
143             e.printStackTrace();
144         }finally{
145                 if(buff != null){
146                     try{
147                         buff.close();
148                     }catch(IOException e){
149                     e.printStackTrace();
150                     }
151                 }
152         }
153 
154     //  Arrays.sort( gft);
155 
156 
157         gift tem;
158         int m1,n;
159         for(m1=0;m1<m;m1++){
160             for(n=m1+1;n<m;n++){
161                 if(gf[m1].gift_price > gf[n].gift_price){
162                     tem= gf[m1];
163                     gf[m1]=gf[n];
164                     gf[n]=tem;
165                 }
166             }
167         }
168 
169 
170 
171         int count = 0;
172         int c,d;
173         Couple gbcouple[] = new Couple[100];
174         for(c=0;c<i;c++){
175             for(d=0;d<j;d++){
```

```
  176                 if(b[c].is_eligible(g[d].expense,g[d].attar) &&
b[c].status.equals("Single") && g[d].status.equals("Single")){
  177                     b[c].girlf = g[d].name;
  178                     g[d].boyf = b[c].name;
  179                     b[c].status = "Is_committed";
  180                     g[d].status = "Is_committed";
  181                     count++;
  182                     String df= new SimpleDateFormat("dd/MM/yy
HH:mm:ss").format(new Time(System.currentTimeMillis()));
  183                     String output =  df+" "+"Boy : "+b[c].name + " is
Committed with " +"Girl : "+g[d].name + "\n" ;
  184                     System.out.println(output);
  185                     try(BufferedWriter buffer=new BufferedWriter(new
FileWriter(out,true))){
  186                         buffer.write(output);
  187
  188                     }
  189                     gbcouple[count-1] = new Couple(b[c],g[d]);
  190
  191                 }
  192             }
  193
  194
  195
  196
  197
  198         }
  199
  200         for(n=0;n<count;n++){
  201             gbcouple[n].calc_happiness();
  202             gbcouple[n].calc_affinity();
  203             gbcouple[n].calc_gifts(gf,m);
  204 }
  205         Couple cu;
  206         int x,y;
  207         for(x=0;x<count;x++){
  208             for(y=x+1;y<count;y++){
  209                 if(gbcouple[x].happiness < gbcouple[y].happiness){
  210                     cu= gbcouple[x];
  211                     gbcouple[x]=gbcouple[y];
  212                     gbcouple[y]=cu;
  213                 }
  214             }
  215         }
  216         Random happi = new Random();
  217         int h = happi.nextInt(count) +1;
  218         System.out.println("Best " +h+ " happiest Couples :"+"\n");
  219         for(x=0;x<h;x++){
  220         System.out.println(gbcouple[x].boy.name+" and
"+gbcouple[x].girl.name+"\n");
  221
  222         String happy = "happiness : "+ gbcouple[x].boy.name+" and
"+gbcouple[x].girl.name+" & happiness is "+gbcouple[x].happiness +"\n";
  223         try(BufferedWriter buffer=new BufferedWriter(new
FileWriter(out,true))){
  224             buffer.write(happy);
  225
  226         }}
  227         for(x=0;x<count;x++){
  228             for(y=x+1;y<count;y++){
  229                 if(gbcouple[x].affinity < gbcouple[y].affinity){
  230                     cu= gbcouple[x];
  231                     gbcouple[x]=gbcouple[y];
  232                     gbcouple[y]=cu;
  233                 }
  234             }
  235         }
  236         System.out.println("Best " +h+ " Compaitable Couples :"+"\n");
  237
  238         for(x=0;x<h;x++){
  239             System.out.println(gbcouple[x].boy.name+" and
"+gbcouple[x].girl.name+"\n");
  240
  241         String Comp = "compaitable : "+ gbcouple[x].boy.name+" and
"+gbcouple[x].girl.name+" & compatibality is "+gbcouple[x].affinity +"\n";
  242         try(BufferedWriter buffer=new BufferedWriter(new
FileWriter(out,true))){
```

```
243             buffer.write(Comp);
244         }
245         }
246     }
```

The documentation for this class was generated from the following file:

12 q3.java

Index

INDE