My Project

AUTHOR
Version
10/04/2017

Table of Contents

Table of contents

Class Index

Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Class Documentation

Boys Class Reference

Boys class.

Detailed Description

Boys class.

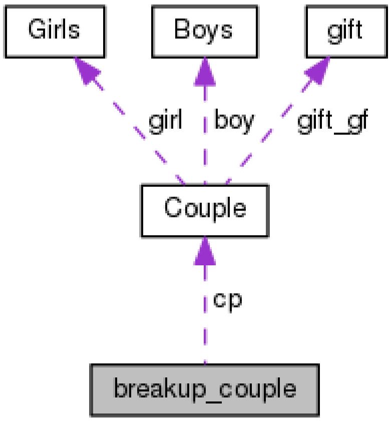
The documentation for this class was generated from the following file:

1 Boys.java

breakup_couple Class Reference

Contain breakup details.

Collaboration diagram for breakup_couple:



IMAGE

Detailed Description

Contain breakup details.

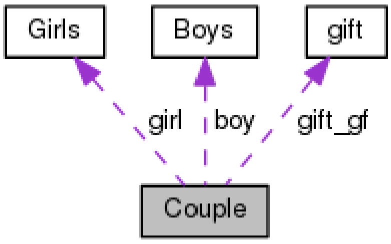
The documentation for this class was generated from the following file:

2 breakup_couple.java

Couple Class Reference

Couple class to make couple.

Collaboration diagram for Couple:



IMAGE

Detailed Description

Couple class to make couple.

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

3 Couple.java

csv_gen Class Reference

Class to generate csv file.

Detailed Description

Class to generate csv file.

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

4 csv_gen.java

gift Class Reference

gift class

Detailed Description

gift class

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

5 gift.java

Girls Class Reference

Girls Class.

Detailed Description

Girls Class.

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

6 Girls.java

MyLogger Class Reference

Class MyLogger.

Detailed Description

Class MyLogger.

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

7 MyLogger.java

q4 Class Reference

Main class q4.

Static Public Member Functions

static void main (String args[]) throws IOException

Detailed Description

Main class q4.

Member Function Documentation

static void q4.main (String *args*[]) throws IOException`[static]`

Sorting Gift according to price

```
                                                              {
        int i,j;
        i=0;
        j=0;
        int m=0;
        Boys b[] = new Boys[1000];
        csv_gen csv_files = new csv_gen();
        csv_files.csv();
        String csvFile = "boy.csv";
        File out=new File("log.log");
        String line = "";
        String csvSplit = ",";
        BufferedReader buff = null;
        try{
            buff = new BufferedReader(new FileReader(csvFile));
            while((line = buff.readLine() )!= null){
                String[] boys_table = line.split(csvSplit);

                b[i] = new Boys();
                b[i].name = boys_table[0];
                b[i].intelligence = Integer.parseInt(boys_table[1]);
                b[i].attractivness = Integer.parseInt(boys_table[2]);
                b[i].min_attr_req = Integer.parseInt(boys_table[3]);
                b[i].budget = Integer.parseInt(boys_table[4]);
                b[i].status = boys_table[5];
                b[i].boy_type= boys_table[6];
                b[i].girlf="";
                b[i].happiness_boy = 0;
                i++;
            }
        }

        catch(FileNotFoundException e){
            e.printStackTrace();
        }catch(IOException e){
            e.printStackTrace();
        }

        Girls g[] = new Girls[1000];
        csvFile = "girl.csv";
        line = "";
        csvSplit = ",";
        buff = null;


        try{
            buff = new BufferedReader(new FileReader(csvFile));
            while((line = buff.readLine() )!= null){
                String[] girls_table = line.split(csvSplit);

                g[j] = new Girls();
                g[j].name = girls_table[0];
                g[j].attar = Integer.parseInt(girls_table[1]);
                g[j].expense = Integer.parseInt(girls_table[2]);
                g[j].intelligence = Integer.parseInt(girls_table[3]);
                g[j].status = girls_table[4];
                g[j].girl_type = girls_table[5];
                g[j].boyf = "";
                g[j].happiness_girl= 0;

                j++;
            }
        }
        catch(FileNotFoundException e){
            e.printStackTrace();
        }catch(IOException e){
            e.printStackTrace();
        }finally{
                if(buff != null){
                    try{
                        buff.close();
                    }catch(IOException e){
                    e.printStackTrace();
                    }
                }
        }


        gift gf[] = new gift[1000];
        csvFile = "gift.csv";
        line = "";
        csvSplit = ",";
        buff = null;


        try{
            buff = new BufferedReader(new FileReader(csvFile));
            while((line = buff.readLine() )!= null){
                String[] gifts_table = line.split(csvSplit);
            //  System.out.println(boy[0] + boy[1] + boy[2] + boy [3]);
                gf[m] = new gift();
                gf[m].gift_name = gifts_table[0];
                gf[m].gift_price = Integer.parseInt(gifts_table[1]);

                gf[m].gift_value = Integer.parseInt(gifts_table[2]);
                gf[m].gift_type = gifts_table[3];

                m++;

            }
        }
        catch(FileNotFoundException e){
            e.printStackTrace();
        }catch(IOException e){
            e.printStackTrace();
        }finally{
                if(buff != null){
                    try{
                        buff.close();
                    }catch(IOException e){
                    e.printStackTrace();
                    }
                }
        }

        gift tem;
        int m1,n;
        for(m1=0;m1<m;m1++){
            for(n=m1+1;n<m;n++){
                if(gf[m1].gift_price > gf[n].gift_price){
                    tem= gf[m1];
                    gf[m1]=gf[n];
                    gf[n]=tem;
                }
            }
        }



        int count = 0;
        String output = "";
        int c = 0,d = 0;
        Couple gbcouple[] = new Couple[100];
```

```
    for(c=0;c<i;c++){
        for(d=0;d<j;d++){
            if(b[c].is_eligible(g[d].expense,g[d].attar) && b[c].status.equals("Single") && g[d].status.equals("Single")){
                b[c].girlf = g[d].name;
                g[d].boyf = b[c].name;
                b[c].status = "Is_committed";
                g[d].status = "Is_committed";
                count++;
                String df= new SimpleDateFormat("dd/MM/yy HH:mm:ss").format(new Time(System.currentTimeMillis()));
                output =  df+" "+"Boy : "+b[c].name + " is Committed with " +"Girl : "+g[d].name + "\n" ;
                System.out.println(output);
                try(BufferedWriter buffer=new BufferedWriter(new FileWriter(out,true))){
                    buffer.write(output);

                }
                gbcouple[count-1] = new Couple(b[c],g[d]);

            }
        }

    }
    int o;

    Random rn = new Random();
    o = rn.nextInt(count);
    breakup_couple bk = new breakup_couple(gbcouple, count, o,output);// Least n happy couples broke up
    String outp = "";
    outp = bk.break_print();
    for(i =0 ; i < o; i++)
    {
        for(j = 0;  j < 20; j++)
        {
            if(b[j].status == "Single" && gbcouple[i].girl.status == "Single"&& !((b[j].name).equals(gbcouple[i].boy.name)) && b[j].budget >= gbcouple[i].girl.expense && gbcouple[i].girl.attar >= b[j].min_attr_req) {
                b[j].status = "Is_committed";
                gbcouple[i].girl.status = "Is_committed";
                String fd= new SimpleDateFormat("dd/MM/yy HH:mm:ss").format(new Time(System.currentTimeMillis()));

                outp = outp.concat("\n"+fd+ " "+"New Commitment : "+gbcouple[i].girl.name +"   and  "+b[j].name+" \n\n");//New commitments of broke up girls

            //  System.out.println("New Commitment after break up: "+gbcouple[i].girl.name +"-"+b[j].name );
                gbcouple[i].boy = b[j];
                break;
            }
        }
    }
    System.out.println(outp);
    try(BufferedWriter buffer=new BufferedWriter(new FileWriter(out,true))){
    buffer.write(outp);

    }
}
```

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

8 q4.java

Index

INDE