```java
package taxpayer;

import java.util.ArrayList;
import receipt.Receipt;

public class Taxpayer {
    private String name = "nobody";
    private int afm = 0;
    private String familyStatus = "Single";
    private float income = 0;
    private ArrayList<Receipt> receipts = new ArrayList<Receipt>();
    private String taxpayerInformationString = "";
    private double basicTax;
    private double receiptTotalAmountPaid;
    private double taxIncrease;
    private double totalTax;
    private ArrayList<String> TaxpayerBunchOfData;
    private ArrayList<Double> percentageForSingleTaxpayer =
            new ArrayList<Double>();
    private ArrayList<Double>
    percentageForMarriedFilingSeparatelyTaxpayer =
    new ArrayList<Double>();
    private ArrayList<Double>
    percentageForMarriedFilingJointlyTaxpayer =
    new ArrayList<Double>();
    private ArrayList<Double> percentageForHeadOfHouseholdTaxpayer =
            new ArrayList<Double>();
    private ArrayList<Double> borderOfTaxForSingleTaxpayer =
            new ArrayList<Double>();
    private ArrayList<Double>
    borderOfTaxForMarriedFilingSeparatelyTaxpayer =
    new ArrayList<Double>();
    private ArrayList<Double>
    borderOfTaxForMarriedFilingJointlyTaxpayer =
    new ArrayList<Double>();
    private ArrayList<Double>
    borderOfTaxForHeadOfHouseholdTaxpayer =
    new ArrayList<Double>();
    private ArrayList<Double> staticTaxForSingleTaxpayer =
            new ArrayList<Double>();
    private ArrayList<Double>
    staticTaxForMarriedFilingSeparatelyTaxpayer =
    new ArrayList<Double>();
    private ArrayList<Double>
    staticTaxForMarriedFilingJointlyTaxpayer =
    new ArrayList<Double>();
    private ArrayList<Double> staticTaxForHeadOfHouseholdTaxpayer =
            new ArrayList<Double>();
    private ArrayList<Double> borderForTaxIncrease =
            new ArrayList<Double>();
    private ArrayList<Double> percentageForTaxIncrease =
            new ArrayList<Double>();
    ArrayList<Double> percentage = getPercentage();
    ArrayList<Double> borderOfTax = getBorderOfTax();
    ArrayList<Double> staticTax = getStaticTax();

    public Taxpayer(){
        initializePercentageTax();
        initializeBorderOfTax();
        initializeStaticTax();
        initializeBorderForTaxIncrease();
        initializePercentageForTaxIncrease();
```

```
63      }
64
65      public void initializeTaxpayer(ArrayList<String> myList){
66          name = myList.get(0);
67          myList.remove(0);
68          afm = Integer.parseInt(myList.get(0));
69          myList.remove(0);
70          familyStatus = myList.get(0);
71          myList.remove(0);
72          income = Float.parseFloat(myList.get(0));
73          myList.remove(0);
74          int i = 0;
75          while(i<myList.size()){
76              ArrayList<String> receiptDataList =
77                      new ArrayList<String>();
78              Receipt newReceipt = new Receipt();
79              ArrayList<String> junk = new ArrayList<String>();
80              junk.add("");
81              junk.add("<>");
82              junk.add("</>");
83              myList.removeAll(junk);
84              for(int j=i; j<i+9; j++){
85                  receiptDataList.add(myList.get(j));
86              }
87              newReceipt.initializeReceipt(receiptDataList);
88              receipts.add(newReceipt);
89              i=i+9;
90          }
91          initializeReceiptTotalAmountPaid();
92          createBasicTax();
93          createTaxIncrease();
94          createTotalTax();
95          createTaxpayerInformationString();
96          createTaxpayerBunchOfData();
97      }
98
99      private void initializePercentageTax(){
100         initializePercentageForSingleTaxpayer();
101         initializePercentageForMarriedFilingSeparatelyTaxpayer();
102         initializePercentageForMarriedFilingJointlyTaxpayer();
103         initializePercentageForHeadOfHouseholdTaxpayer();
104     }
105
106     private void initializeBorderOfTax(){
107         initializeBorderOfTaxForSingleTaxpayer();
108         initializeBorderOfTaxForMarriedFilingSeparatelyTaxpayer();
109         initializeBorderOfTaxForMarriedFilingJointlyTaxpayer();
110         initializeBorderOfTaxForHeadOfHouseholdTaxpayer();
111     }
112
113     private void initializeStaticTax(){
114         initializeStaticTaxForSingleTaxpayer();
115         initializeStaticTaxForMarriedFilingSeparatelyTaxpayer();
116         initializeStaticTaxForMarriedFilingJointlyTaxpayer();
117         initializeStaticTaxForHeadOfHouseholdTaxpayer();
118     }
119
120     private void initializeBorderForTaxIncrease(){
121         borderForTaxIncrease.add(0.0);
122         borderForTaxIncrease.add(20.0);
123         borderForTaxIncrease.add(40.0);
124         borderForTaxIncrease.add(60.0);
```

```
125         borderForTaxIncrease.add(Double.MAX_VALUE);
126     }
127 
128     private void initializePercentageForTaxIncrease(){
129         percentageForTaxIncrease.add(8.0);
130         percentageForTaxIncrease.add(4.0);
131         percentageForTaxIncrease.add(-15.0);
132         percentageForTaxIncrease.add(-30.0);
133     }
134 
135     private void initializePercentageForSingleTaxpayer(){
136         percentageForSingleTaxpayer.add(5.35);
137         percentageForSingleTaxpayer.add(7.05);
138         percentageForSingleTaxpayer.add(7.85);
139         percentageForSingleTaxpayer.add(7.85);
140         percentageForSingleTaxpayer.add(9.85);
141     }
142 
143     private void
144     initializePercentageForMarriedFilingSeparatelyTaxpayer(){
145         percentageForMarriedFilingSeparatelyTaxpayer.add(5.35);
146         percentageForMarriedFilingSeparatelyTaxpayer.add(7.05);
147         percentageForMarriedFilingSeparatelyTaxpayer.add(7.85);
148         percentageForMarriedFilingSeparatelyTaxpayer.add(7.85);
149         percentageForMarriedFilingSeparatelyTaxpayer.add(9.85);
150     }
151 
152     private void
153     initializePercentageForMarriedFilingJointlyTaxpayer(){
154         percentageForMarriedFilingJointlyTaxpayer.add(5.35);
155         percentageForMarriedFilingJointlyTaxpayer.add(7.05);
156         percentageForMarriedFilingJointlyTaxpayer.add(7.05);
157         percentageForMarriedFilingJointlyTaxpayer.add(7.85);
158         percentageForMarriedFilingJointlyTaxpayer.add(9.85);
159     }
160 
161     private void initializePercentageForHeadOfHouseholdTaxpayer(){
162         percentageForHeadOfHouseholdTaxpayer.add(5.35);
163         percentageForHeadOfHouseholdTaxpayer.add(7.05);
164         percentageForHeadOfHouseholdTaxpayer.add(7.05);
165         percentageForHeadOfHouseholdTaxpayer.add(7.85);
166         percentageForHeadOfHouseholdTaxpayer.add(9.85);
167     }
168 
169     private void initializeBorderOfTaxForSingleTaxpayer(){
170         borderOfTaxForSingleTaxpayer.add(0.0);
171         borderOfTaxForSingleTaxpayer.add(24680.0);
172         borderOfTaxForSingleTaxpayer.add(81080.0);
173         borderOfTaxForSingleTaxpayer.add(90000.0);
174         borderOfTaxForSingleTaxpayer.add(152540.0);
175         borderOfTaxForSingleTaxpayer.add(Double.MAX_VALUE);
176     }
177 
178     private void
179     initializeBorderOfTaxForMarriedFilingSeparatelyTaxpayer(){
180         borderOfTaxForMarriedFilingSeparatelyTaxpayer.add(0.0);
181         borderOfTaxForMarriedFilingSeparatelyTaxpayer.add(18040.0);
182         borderOfTaxForMarriedFilingSeparatelyTaxpayer.add(71680.0);
183         borderOfTaxForMarriedFilingSeparatelyTaxpayer.add(90000.0);
184         borderOfTaxForMarriedFilingSeparatelyTaxpayer.add(127120.0);
185         borderOfTaxForMarriedFilingSeparatelyTaxpayer.add(
186                 Double.MAX_VALUE);
```

```
187     }
188
189     private void
190     initializeBorderOfTaxForMarriedFilingJointlyTaxpayer(){
191         borderOfTaxForMarriedFilingJointlyTaxpayer.add(0.0);
192         borderOfTaxForMarriedFilingJointlyTaxpayer.add(36080.0);
193         borderOfTaxForMarriedFilingJointlyTaxpayer.add(90000.0);
194         borderOfTaxForMarriedFilingJointlyTaxpayer.add(143350.0);
195         borderOfTaxForMarriedFilingJointlyTaxpayer.add(254240.0);
196         borderOfTaxForMarriedFilingJointlyTaxpayer.add(
197                 Double.MAX_VALUE);
198     }
199
200     private void initializeBorderOfTaxForHeadOfHouseholdTaxpayer(){
201         borderOfTaxForHeadOfHouseholdTaxpayer.add(0.0);
202         borderOfTaxForHeadOfHouseholdTaxpayer.add(30390.0);
203         borderOfTaxForHeadOfHouseholdTaxpayer.add(90000.0);
204         borderOfTaxForHeadOfHouseholdTaxpayer.add(122110.0);
205         borderOfTaxForHeadOfHouseholdTaxpayer.add(203390.0);
206         borderOfTaxForHeadOfHouseholdTaxpayer.add(Double.MAX_VALUE);
207     }
208
209     private void initializeStaticTaxForSingleTaxpayer(){
210         staticTaxForSingleTaxpayer.add(0.0);
211         staticTaxForSingleTaxpayer.add(1320.38);
212         staticTaxForSingleTaxpayer.add(5296.58);
213         staticTaxForSingleTaxpayer.add(5996.80);
214         staticTaxForSingleTaxpayer.add(10906.19);
215     }
216
217     private void
218     initializeStaticTaxForMarriedFilingSeparatelyTaxpayer(){
219         staticTaxForMarriedFilingSeparatelyTaxpayer.add(0.0);
220         staticTaxForMarriedFilingSeparatelyTaxpayer.add(965.14);
221         staticTaxForMarriedFilingSeparatelyTaxpayer.add(4746.76);
222         staticTaxForMarriedFilingSeparatelyTaxpayer.add(6184.88);
223         staticTaxForMarriedFilingSeparatelyTaxpayer.add(9098.80);
224     }
225
226     private void initializeStaticTaxForMarriedFilingJointlyTaxpayer(){
227         staticTaxForMarriedFilingJointlyTaxpayer.add(0.0);
228         staticTaxForMarriedFilingJointlyTaxpayer.add(1930.28);
229         staticTaxForMarriedFilingJointlyTaxpayer.add(5731.64);
230         staticTaxForMarriedFilingJointlyTaxpayer.add(9492.82);
231         staticTaxForMarriedFilingJointlyTaxpayer.add(18197.69);
232     }
233
234     private void initializeStaticTaxForHeadOfHouseholdTaxpayer(){
235         staticTaxForHeadOfHouseholdTaxpayer.add(0.0);
236         staticTaxForHeadOfHouseholdTaxpayer.add(1625.87);
237         staticTaxForHeadOfHouseholdTaxpayer.add(5828.38);
238         staticTaxForHeadOfHouseholdTaxpayer.add(8092.13);
239         staticTaxForHeadOfHouseholdTaxpayer.add(14472.61);
240     }
241
242     private void createBasicTax(){
243         for(int i=0; i<percentage.size(); i++){
244             if (checkIncomeCategory(i)){
245                 basicTax = computeBasicTax(i);
246                 return;
247             }
248         }
```

```
249         System.out.println("Error can't create basic tax");
250         System.exit(-1);
251     }
252
253     private double computeBasicTax(int categoryNumber){
254         return staticTax.get(categoryNumber) + percentage.get(
255                 categoryNumber) * (income - borderOfTax.get(
256                         categoryNumber)) / 100;
257     }
258
259     private boolean checkIncomeCategory(int categoryNumber){
260         return borderOfTax.get(categoryNumber) >= income &&
261                 income < borderOfTax.get(categoryNumber+1);
262     }
263
264     private ArrayList<Double> getPercentage(){
265         if (familyStatus.equals("Married Filing Jointly")){
266             return percentageForMarriedFilingJointlyTaxpayer;
267         }else if(familyStatus.equals("Married Filing Separately")){
268             return percentageForMarriedFilingSeparatelyTaxpayer;
269         }else if(familyStatus.equals("Single")){
270             return percentageForSingleTaxpayer;
271         }else if(familyStatus.equals("Head Of Household")){
272             return percentageForHeadOfHouseholdTaxpayer;
273         }else{
274             System.out.println("Can't find suck taxpayer type!");
275             System.exit(-1);
276             return percentageForMarriedFilingSeparatelyTaxpayer;
277         }
278     }
279
280     private ArrayList<Double> getBorderOfTax(){
281         if (familyStatus.equals("Married Filing Jointly")){
282             return borderOfTaxForMarriedFilingJointlyTaxpayer;
283         }else if(familyStatus.equals("Married Filing Separately")){
284             return borderOfTaxForMarriedFilingSeparatelyTaxpayer;
285         }else if(familyStatus.equals("Single")){
286             return borderOfTaxForSingleTaxpayer;
287         }else if(familyStatus.equals("Head Of Household")){
288             return borderOfTaxForHeadOfHouseholdTaxpayer;
289         }else{
290             System.out.println("Can't find suck taxpayer type!");
291             System.exit(-1);
292             return percentageForMarriedFilingSeparatelyTaxpayer;
293         }
294     }
295
296     private ArrayList<Double> getStaticTax(){
297         if (familyStatus.equals("Married Filing Jointly")){
298             return staticTaxForMarriedFilingJointlyTaxpayer;
299         }else if(familyStatus.equals("Married Filing Separately")){
300             return staticTaxForMarriedFilingSeparatelyTaxpayer;
301         }else if(familyStatus.equals("Single")){
302             return percentageForSingleTaxpayer;
303         }else if(familyStatus.equals("Head Of Household")){
304             return staticTaxForHeadOfHouseholdTaxpayer;
305         }else{
306             System.out.println("Can't find suck taxpayer type!");
307             System.exit(-1);
308             return percentageForMarriedFilingSeparatelyTaxpayer;
309         }
310     }
```

```
311
312     private void createTaxIncrease(){
313         receiptTotalAmountPaid = this.getReceiptTotalAmountPaid();
314         for(int i=0; i<percentageForTaxIncrease.size(); i++){
315             if(checkTaxIncreaseCategory(i)){
316                 taxIncrease = computeTaxIncrease(i);
317                 return;
318             }
319         }
320         System.out.println(
321                 "Something went wrong calculating tax increase!");
322         System.exit(-1);
323     }
324
325     private boolean checkTaxIncreaseCategory(int categoryNumber){
326         return receiptTotalAmountPaid >= borderForTaxIncrease.get(
327                 categoryNumber)*income/100 && receiptTotalAmountPaid
328                 < borderForTaxIncrease.get(categoryNumber+1)*
329                 income/100;
330     }
331
332     private double computeTaxIncrease(int categoryNumber){
333         return percentageForTaxIncrease.get(categoryNumber)*basicTax /
334                 100;
335     }
336
337     private void createTotalTax(){
338         double basicTax = this.getBasicTax();
339         double taxIncrease = this.getTaxIncrease();
340         totalTax = basicTax + taxIncrease;
341     }
342
343     private void createTaxpayerInformationString(){
344         taxpayerInformationString = " name : " + name + " afm : " +
345     afm + " family status : "
346                 + familyStatus + " income : " + income + "/n";
347         for(int i=0; i<receipts.size(); i++){
348             taxpayerInformationString += " receipt code: " +
349         receipts.get(i).getReceiptCode();
350             taxpayerInformationString += " date of issue: " +
351         receipts.get(i).getDateOfIssue();
352             taxpayerInformationString += " category: " +
353         receipts.get(i).getCategory();
354             taxpayerInformationString += " amount paid: " +
355         receipts.get(i).getAmountPaid();
356             taxpayerInformationString += " company name: " +
357         receipts.get(i).getSeller().getCompanyName();
358             taxpayerInformationString += " country: " +
359         receipts.get(i).getSeller().getCountry();
360             taxpayerInformationString += " city: " +
361         receipts.get(i).getSeller().getCity();
362             taxpayerInformationString += " street: " +
363         receipts.get(i).getSeller().getStreet();
364             taxpayerInformationString += " street number: " +
365         receipts.get(i).getSeller().getStreetNumber();
366         }
367     }
368
369     private void createTaxpayerBunchOfData(){
370         createBasicTax();
371         createTaxIncrease();
372         createTotalTax();
```

```
373         TaxpayerBunchOfData = new ArrayList<String>();
374         TaxpayerBunchOfData.add(name);
375         TaxpayerBunchOfData.add(afm+"");
376         TaxpayerBunchOfData.add(familyStatus+"");
377         TaxpayerBunchOfData.add(income+"");
378         TaxpayerBunchOfData.add(basicTax+"");
379         TaxpayerBunchOfData.add(taxIncrease+"");
380         TaxpayerBunchOfData.add(totalTax+"");
381         TaxpayerBunchOfData.add(getTotalReceiptsGathered()+"");
382         TaxpayerBunchOfData.add(getCategoryNumberOfReceipts(
383                 "Entertainment")+"");
384         TaxpayerBunchOfData.add(getCategoryNumberOfReceipts(
385                 "Basic")+"");
386         TaxpayerBunchOfData.add(getCategoryNumberOfReceipts(
387                 "Travel")+"");
388         TaxpayerBunchOfData.add(getCategoryNumberOfReceipts(
389                 "Health")+"");
390         TaxpayerBunchOfData.add(getCategoryNumberOfReceipts(
391                 "Other")+"");
392         for(int i=0; i<receipts.size(); i++){
393             TaxpayerBunchOfData.addAll(receipts.get(i).
394                     createReceiptBunchOfData());
395         }
396     }
397 
398     private void initializeReceiptTotalAmountPaid(){
399         receiptTotalAmountPaid = 0;
400         for(int i=0; i<receipts.size(); i++){
401             receiptTotalAmountPaid += receipts.get(i).getAmountPaid();
402         }
403     }
404 
405     public void setReceipts(ArrayList<Receipt> receipts){
406         this.receipts = receipts;
407     }
408 
409     public int getCategoryNumberOfReceipts(String receiptCategory){
410         int categoryNumberOfReceipts = 0;
411         for(int i=0; i<receipts.size(); i++){
412             if(receipts.get(i).getCategory().equals(receiptCategory)){
413                 categoryNumberOfReceipts += 1;
414             }
415         }
416         return categoryNumberOfReceipts;
417     }
418 
419     public double getCategoryReceiptAmountPaid(
420             String receiptCategory){
421         double categoryReceiptAmountPaid = 0;
422         for(int i=0; i<receipts.size(); i++){
423             if(receipts.get(i).getCategory().equals(receiptCategory)){
424                 categoryReceiptAmountPaid +=
425                         receipts.get(i).getAmountPaid();
426             }
427         }
428         return categoryReceiptAmountPaid;
429     }
430 
431     public Taxpayer getInitializedTaxpayer(ArrayList<String> myList){
432         initializeTaxpayer(myList);
433         return this;
434     }
```

```
435
436     public ArrayList<String> getTaxpayerBunchOfData(){
437         createTaxpayerBunchOfData();
438         return TaxpayerBunchOfData;
439     }
440
441     public String getTaxpayerInformationString(){
442         createTaxpayerInformationString();
443         return taxpayerInformationString;
444     }
445
446     public double getReceiptTotalAmountPaid(){
447         initializeReceiptTotalAmountPaid();
448         return receiptTotalAmountPaid;
449     }
450
451     public double getBasicTax(){
452         return basicTax;
453     }
454
455     public double getTaxIncrease(){
456         return taxIncrease;
457     }
458
459     public double getTotalTax(){
460         return totalTax;
461     }
462
463     public int getTotalReceiptsGathered(){
464         return receipts.size();
465     }
466
467     public String getName(){
468         return name;
469     }
470
471     public int getAfm(){
472         return afm;
473     }
474
475     public ArrayList<Receipt> getReceipts(){
476         return receipts;
477     }
478 }
```