```
##########################################

Student Name = Jitong Ding

##########################################



CSE017 Grading sheet for Jitong Ding

Homework Assignment TaxTable

Total points                          maximum: 100

Completeness: All class/methods included (40)   [  40  ]

Compilation: Program compiles (20)               [  20  ]

Execution: Program executes properly (30)        [  30  ]

Style: Program obeys style rules (10)            [  10  ]

Subtotal                                           100

Late Penalty                                     [      ]

Total Points                                       100


##########################################
##########################################
```

```
/**
CSE 17
Jitong Ding
jid221
Program #1    Deadline: September 21, 2018
Program:  A class for two people who are married
*/

public class Couple{
  /** A private data field named firstSpouse for the firstSpouse of the Couple.
*/
  private Person firstSpouse;
  /** A private data field named secondSpouse for the secondSpouse of the Couple
. */
  private Person secondSpouse;

  /** Construct a new Couple with firstSpouse and  secondSpouse*/
  public Couple(Person aFirstSpouse, Person aSecondSpouse){
    firstSpouse = aFirstSpouse;
    secondSpouse = aSecondSpouse;
  }

  /** An instance method return the firstSpouse */
  public Person getFirstSpouse(){
    return firstSpouse;
  }

  /** An instance method return the secondSpouse */
  public Person getSecondSpouse(){
    return secondSpouse;
  }

  /** An instance method return the total income */
  public int getTotalIncome(){
    int sum = 0;
    return sum = firstSpouse.getIncome() + secondSpouse.getIncome();
  }

   /** An instance method return the string words of the title */
  public String toString(){
    return firstSpouse.getName() + "&" + secondSpouse.getName()
      + ":$" + firstSpouse.getIncome() +"/$" + secondSpouse.getIncome();
  }

   /** An instance method return the saving */
  public double calculateSavings(TaxTable aBase, TaxTable aComparison){
    double differ = 0;
    double saving1 = 0;
    double saving2 = 0;
    saving1 = getTotalIncome()- aComparison.calculateTax(getTotalIncome());
    saving2 = getTotalIncome()- aBase.calculateTax(getTotalIncome());
    return differ = saving1-saving2;
  }
}
```

```
/**
CSE 17
Jitong Ding
jid221
Prpgram #1    Deadline: September 21, 2018
Program:  A class for individual and their income
*/

public class Person{

  /** A private data field named name for the name of the person. */
  private String name;
  /** A private data field named income for the income of the person. */
  private int income;

  /** Construct a new Play with name and income */
  public Person(String aName, int aIncome){
    name = aName;
    income = aIncome;
  }

  /** An instance method return the name */
  public String getName(){
    return name;
  }

  /** An instance method return the income */
  public int getIncome(){
    return income;
  }

  /** An instance method return the new income */
  public void setIncome(int income){
    this.income = income;
  }

}
```

```
/**
CSE 17
Jitong Ding
jid221
Program  Deadline: September 21, 2018
Program Description:  Tax Tables
*/

public class TaxTable{

  /** A private data field named description for the description of the TaxTable
. */
  private String description;
  /** A private data field named incomeLevels for the incomeLevels of the TaxTab
le. */
  private int[] incomeLevels;
  /** A private data field named rates for ratesof the TaxTable. */
  private double[] rates;
  /** A private data field named jointFile for jointFile of the TaxTable. */
  private boolean jointFile;

  /** Construct a new TaxTable description, incomeLevels, rates */
  public TaxTable(String aDescription, int[] aIncomeLevels, double[] aRates){
    description = aDescription;
    incomeLevels = aIncomeLevels;
    rates = aRates;
    jointFile = false;
  }

  /** Construct a new TaxTable with description,incomeLevels,rates and jointFile
 */
  public TaxTable(String theDescription, int[] theIncomeLevels, double[] theRate
s, boolean aJointFile){
    description = theDescription;
    incomeLevels = theIncomeLevels;
    rates = theRates;
    jointFile = aJointFile;
  }

  /** An instance method return the description */
  public String getDescription(){
    return description;
  }

  /** An instance method return the tax */
  public double calculateTax(int aIncome){
    double taxes = 0;

    /** A for loop to compute the tax bracket by bracket */
    for (int i=0; i<incomeLevels.length;++i){
      if(i==incomeLevels.length-1){
        taxes += (aIncome- (incomeLevels[i]-1))*rates[i];
        break;
      }
      else if(aIncome <= (incomeLevels[i+1]-1)){
        taxes += (aIncome - (incomeLevels[i]-1))*rates[i];
        break;
    }
      else{
        if(incomeLevels[i]==0){
          taxes = ((incomeLevels[i+1]-1)-(incomeLevels[i]))*rates[i];
        }
        else{
          taxes += ((incomeLevels[i+1]-1)- (incomeLevels[i]-1))*rates[i];
        }
    }
}
  return taxes;
  }

  /** An instance method return the tax */
  public double calculateTax(Couple pair){
    double theTax = 0;
```



```
    /** if statement to figure out if it is jointFile*/
    if(jointFile){
      theTax = calculateTax(pair.getTotalIncome());
    }
    else{
      theTax = calculateTax(pair.getFirstSpouse().getIncome())
        + calculateTax(pair.getSecondSpouse().getIncome());
    }
    return theTax;
  }

  /** A menthod for printing a table of the four kinds of TaxTables*/
  public static void printTaxTables(TaxTable table1, TaxTable table2){
    System.out.printf("%s  %s\n", table1.description, table2.description);
    System.out.println("------------------------------  ------------------------
---");
    /** A for loop to print the incomelevels and the rate*/
    for (int i = 0; i<table1.incomeLevels.length; ++i){
      if(i<table1.incomeLevels.length-1){
        System.out.printf("$%-20s%2.1f%%",(table1.incomeLevels[i])
                          +" - $"+(table1.incomeLevels[i+1]-1)+":",table1.rates
[i]*100);
        System.out.printf("    $%-20s%2.1f%%\n",(table2.incomeLevels[i])
                          +" - $"+(table2.incomeLevels[i+1]-1)+":",table2.rates
[i]*100);
      }
      else{
        System.out.printf("$%-20s%2.1f%%",(table1.incomeLevels[i])+"+"+":", table
1.rates[i]*100);
        System.out.printf("    $%-20s%2.1f%% \n", (table2.incomeLevels[i])
                          +"+"+":", table2.rates[i]*100);
      }
    }
  }

  /** A method to sort the savings and sort the Couple[]*/
  public static void sortBySavings(Couple[] aPair, TaxTable theBase, TaxTable th
eComparison){
    double[] saving = new double[aPair.length];
    saving[0]= aPair[0].calculateSavings(theBase, theComparison);
    saving[1]= aPair[1].calculateSavings(theBase, theComparison);
    saving[2]= aPair[2].calculateSavings(theBase, theComparison);
    saving[3]= aPair[3].calculateSavings(theBase, theComparison);

    /** Using bubble sort to sort the savings and Couple[]*/
    /**A for loop*/
    for(int i=0; i< saving.length; ++i){
      for(int j=0; j< saving.length-1; ++j){

        if(saving[j]<saving[j+1]){
          double temp = saving[j];
          Couple[] a = new Couple[1];
          a[0] = aPair[j];
          saving[j] = saving[j+1];
          aPair[j] = aPair[j+1];
          saving[j+1] = temp;
          aPair[j+1] = a[0];
        }
      }
    }
  }

   /** The main method */
  public static void main(String[] args){

    /** Creat four TaxTable Objects */
    TaxTable[] tax = new TaxTable[4];
    tax[0] = new TaxTable("2017 Married Filing Separately",
                          new int[] {0, 9326, 37951,76551,116676, 208351, 235351
},
                          new double[] {0.10, 0.15, 0.25, 0.28,0.33, 0.35, 0.396
});
    tax[1] = new TaxTable("2017 Married Filing Jointly",
```

```
                            new int[] {0, 18651, 75901, 153101, 233351, 416701, 47
0701},
                            new double[] {0.10, 0.15, 0.25, 0.28,0.33, 0.35, 0.396
}, true);
    tax[2] = new TaxTable("2018 Married Filing Separately",
                            new int[] {0, 9526, 38701, 82501, 157501, 200001, 3000
01},
                            new double[] {0.10, 0.12, 0.22, 0.24,0.32, 0.35, 0.37}
);
    tax[3] = new TaxTable("2018 Married Filing Jointly",
                            new int[] {0, 19051, 77401, 165001, 315001, 400001, 60
0001},
                            new double[] {0.10, 0.12, 0.22, 0.24,0.32, 0.35, 0.37}
, true);

    /** Print two table of the TaxTable*/
    printTaxTables(tax[0],tax[1]);
    System.out.println();
    /** Print two table of the TaxTable*/
    printTaxTables(tax[2],tax[3]);

    /** Creat four Couple Objects */
    Couple[] marry = new Couple[4];
    marry[0] = new Couple(new Person("Michelle", 50000),new Person("Joe", 25000));
    marry[1] = new Couple(new Person("Bob", 20000),new Person("Theresa", 0));
    marry[2] = new Couple(new Person("Gary", 21000000),new Person("Lisa", 50000))
;
    marry[3] = new Couple(new Person("Henry", 140000),new Person("Ray", 90000));

    System.out.println();
    System.out.println("The Taxes Michelle and Joe owe under each tax table:  ");
    /** A for loop to print the Michelle family's tax based on four different Ta
xTable*/
    for(int i=0; i<tax.length; ++i){
      System.out.printf("%-36s$%8.2f\n",tax[i].description,tax[i].calculateTax(mar
ry[0]));
    }

    /** invoke the sortBySavings methpod*/
    sortBySavings(marry,tax[1],tax[3]);
    System.out.println();
    System.out.println("Savings for 2018 joint filers vs. 2017 joint filers:");
    /** A for loop to print the savings of four families in descending order*/
    for(int i=0; i<marry.length;++i){
      System.out.printf("%-36s$%10.2f \n", marry[i].toString(),marry[i].calculateS
avings(tax[1],tax[3]));

  }
}
}
```

```
##########################################
##########################################


############ Compiled Result ###############


Source Code Compilation:



##########################################


##########################################
############ Execution Result ##############
##########################################


##########################################
Test1 output - testOutput1.txt


2017 Married Filing Separately     2017 Married Filing Jointly
-------------------------------    -------------------------------
$0 - $9325:           10.0%        $0 - $18650:          10.0%
$9326 - $37950:       15.0%        $18651 - $75900:      15.0%
$37951 - $76550:      25.0%        $75901 - $153100:     25.0%
$76551 - $116675:     28.0%        $153101 - $233350:    28.0%
$116676 - $208350:    33.0%        $233351 - $416700:    33.0%
$208351 - $235350:    35.0%        $416701 - $470700:    35.0%
$235351+:             39.6%        $470701+:             39.6%

2018 Married Filing Separately     2018 Married Filing Jointly
-------------------------------    -------------------------------
$0 - $9525:           10.0%        $0 - $19050:          10.0%
$9526 - $38700:       12.0%        $19051 - $77400:      12.0%
$38701 - $82500:      22.0%        $77401 - $165000:     22.0%
$82501 - $157500:     24.0%        $165001 - $315000:    24.0%
$157501 - $200000:    32.0%        $315001 - $400000:    32.0%
$200001 - $300000:    35.0%        $400001 - $600000:    35.0%
$300001+:             37.0%        $600001+:             37.0%

The Taxes Michelle and Joe owe under each tax table:
2017 Married Filing Separately      $11522.50
2017 Married Filing Jointly         $10317.50
2018 Married Filing Separately      $ 9749.00
2018 Married Filing Jointly         $ 8619.00

Savings for 2018 joint filers vs. 2017 joint filers:
Gary & Lisa: $21000000 / $50000     $ 553151.80
Henry & Ray: $140000 / $90000       $   7505.50
Michelle & Joe: $50000 / $25000     $   1698.50
Bob & Theresa: $20000 / $0          $     48.50
```