```
//Mickie Blair
//Java I – CIST 2371
//Final Project Invoice - Invoice Class

package FinalProjectInvoice;

public class Invoice
{
   private int invoiceNumber;                  //field for invoice number
   private double balanceDue;                  //field for balance due
   private int month;                          //field for month
   private int day;                            //field for day
   private int year;                            //field for year

   public Invoice(int number, double balance, int month, int day, int year)
      {
         //force invoice number to be 0 if it's less than 1000
         if (number < 1000)
         {
            invoiceNumber = 0;
         }
         else
         {
            invoiceNumber = number;
         }

         //balance field
         balanceDue = balance;

         //if month is not between 1 and 12 force month to be 0
         if (month < 1 || month > 12)
         {
            this.month = 0;
         }
         else
         {
            this.month = month;
         }

         //if day is not between 1 and 31 force month to be 0
         if (day < 1 ||day > 31)
         {
            this.day = 0;
         }
         else
         {
            this.day = day;
         }
```

```
          //if year is not between 2011 and 2017 force month to be 0
          if (year < 2011 || year > 2017)
          {
             this.year = 0;
          }
          else
          {
             this.year = year;
          }
       }

    public void displayResults()
    {
       System.out.println("Test Results:");
       System.out.println("-----------------------------------");
       System.out.printf("Invoice Number: \t%11d\n", invoiceNumber);
       System.out.printf("Balance Due:\t\t$%10.2f\n", balanceDue);
       System.out.printf("Due Date: \t\t %02d-%02d-%04d\n", month, day, year);
    }
}

//Mickie Blair
//Java I – CIST 2371
//Final Project Invoice - TestInvoice Class

package FinalProjectInvoice;

public class TestInvoice
{

       public static void main(String[] args)
       {
          System.out.println("Invoice Constructor Tests\n");

          //test 1
          System.out.println ("Constructor Test 1 - Invoice Test");
          System.out.println("-----------------------------------");
          System.out.println ("Data to be sent to constructor:\n"
                    + "Invoice Number:\t 100\n"
                    + "Balance Due:\t 212.12\n"
                    + "Month:\t\t 1\n"
                    + "Day:\t\t 15\n"
                    + "Year:\t\t 2015\n");
          //create test1 object
          Invoice test1=new Invoice(100, 212.12, 1, 15, 2015);

          //display test1 results
          test1.displayResults();
```

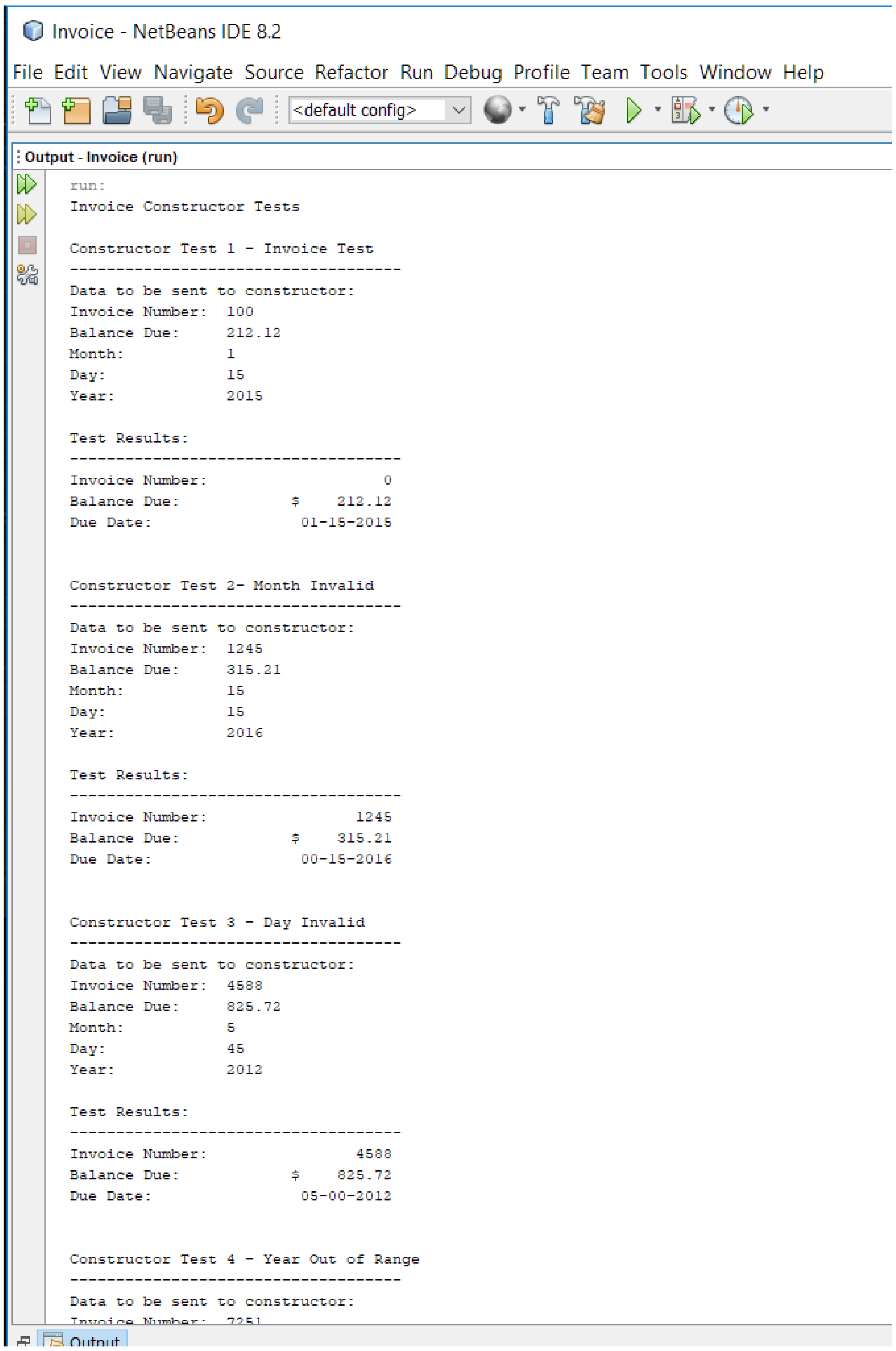
```
//test 2
System.out.println();
System.out.println();
System.out.println ("Constructor Test 2- Month Invalid");
System.out.println("------------------------------------");
System.out.println ("Data to be sent to constructor:\n"
                + "Invoice Number:\t 1245\n"
                + "Balance Due:\t 315.21\n"
                + "Month:\t\t 15\n"
                + "Day:\t\t 15\n"
                + "Year:\t\t 2016\n");
//create test2 object
Invoice test2=new Invoice(1245, 315.21, 15, 15, 2016);

//display test2 results
test2.displayResults();

//test 3
System.out.println();
System.out.println();
System.out.println ("Constructor Test 3 - Day Invalid");
System.out.println("------------------------------------");
System.out.println ("Data to be sent to constructor:\n"
                + "Invoice Number:\t 4588\n"
                + "Balance Due:\t 825.72\n"
                + "Month:\t\t 5\n"
                + "Day:\t\t 45\n"
                + "Year:\t\t 2012\n");
//create test3 object
Invoice test3=new Invoice(4588, 825.72, 5, 45, 2012);

//display test3 results
test3.displayResults();

//test 4
System.out.println();
System.out.println();
System.out.println ("Constructor Test 4 - Year Out of Range");
System.out.println("------------------------------------");
System.out.println ("Data to be sent to constructor:\n"
                + "Invoice Number:\t 7251\n"
                + "Balance Due:\t 129.92\n"
                + "Month:\t\t 7\n"
                + "Day:\t\t 21\n"
                + "Year:\t\t 2010\n");
//create test4 object
Invoice test4=new Invoice(7251, 129.92, 7, 21, 2010);

//display test results
test4.displayResults();
```

```
        //test 5
        System.out.println();
        System.out.println();
        System.out.println ("Constructor Test 5 - All Data Valid");
        System.out.println("------------------------------------");
        System.out.println ("Data to be sent to constructor:\n"
                  + "Invoice Number:\t 3269\n"
                  + "Balance Due:\t 719.33\n"
                  + "Month:\t\t 9\n"
                  + "Day:\t\t 28\n"
                  + "Year:\t\t 2013\n");
        //create test5 object
        Invoice test5=new Invoice(3269, 719.33, 9, 28, 2013);

        //display test results
        test5.displayResults();
    }
}
```

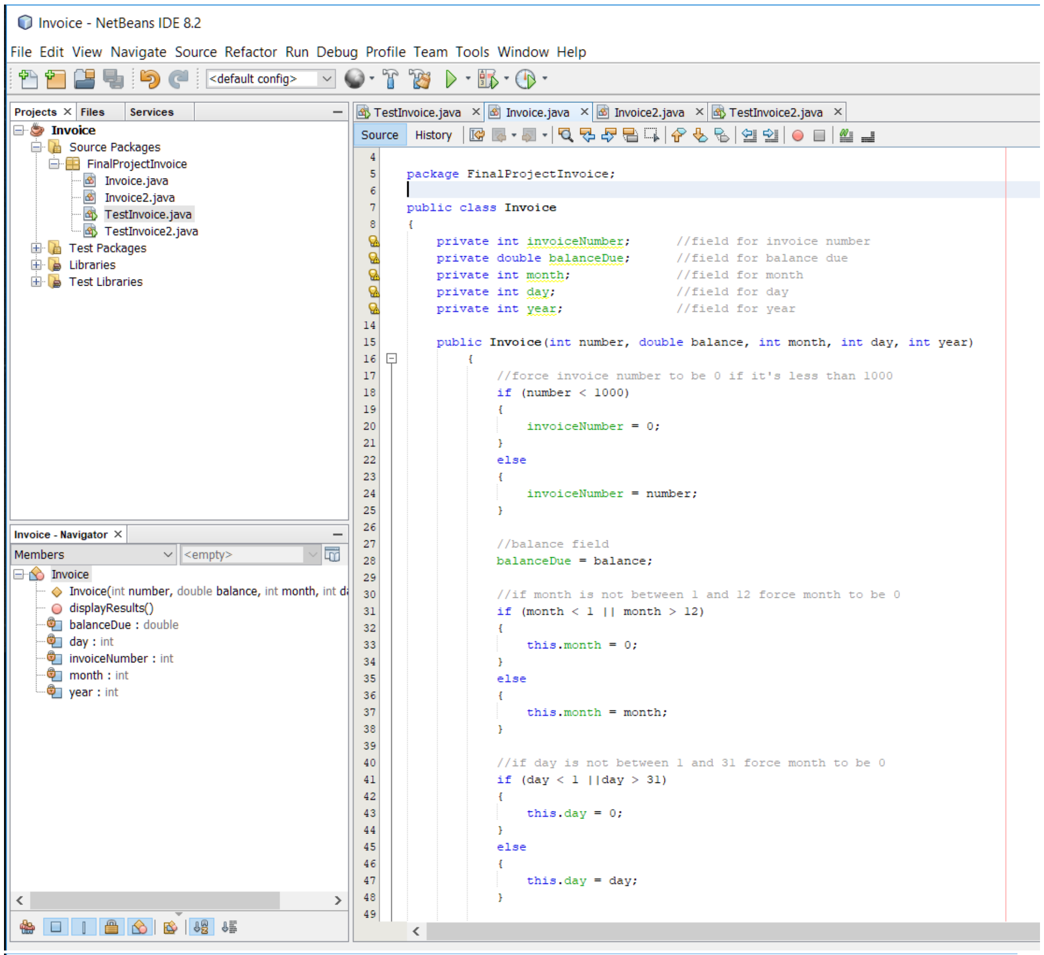
OUTPUT

```
Invoice Constructor Tests

Constructor Test 1 - Invoice Test
------------------------------------
Data to be sent to constructor:
Invoice Number:   100
Balance Due:             212.12
Month:                   1
Day:                     15
Year:                    2015

Test Results:
------------------------------------
Invoice Number:                      0
Balance Due:                 $  212.12
Due Date:                   01-15-2015

Constructor Test 2- Month Invalid
------------------------------------
Data to be sent to constructor:
Invoice Number:   1245
Balance Due:             315.21
Month:                   15
Day:                     15
Year:                     2016

Test Results:
------------------------------------
Invoice Number:                   1245
Balance Due:                 $  315.21
Due Date:                   00-15-2016
```

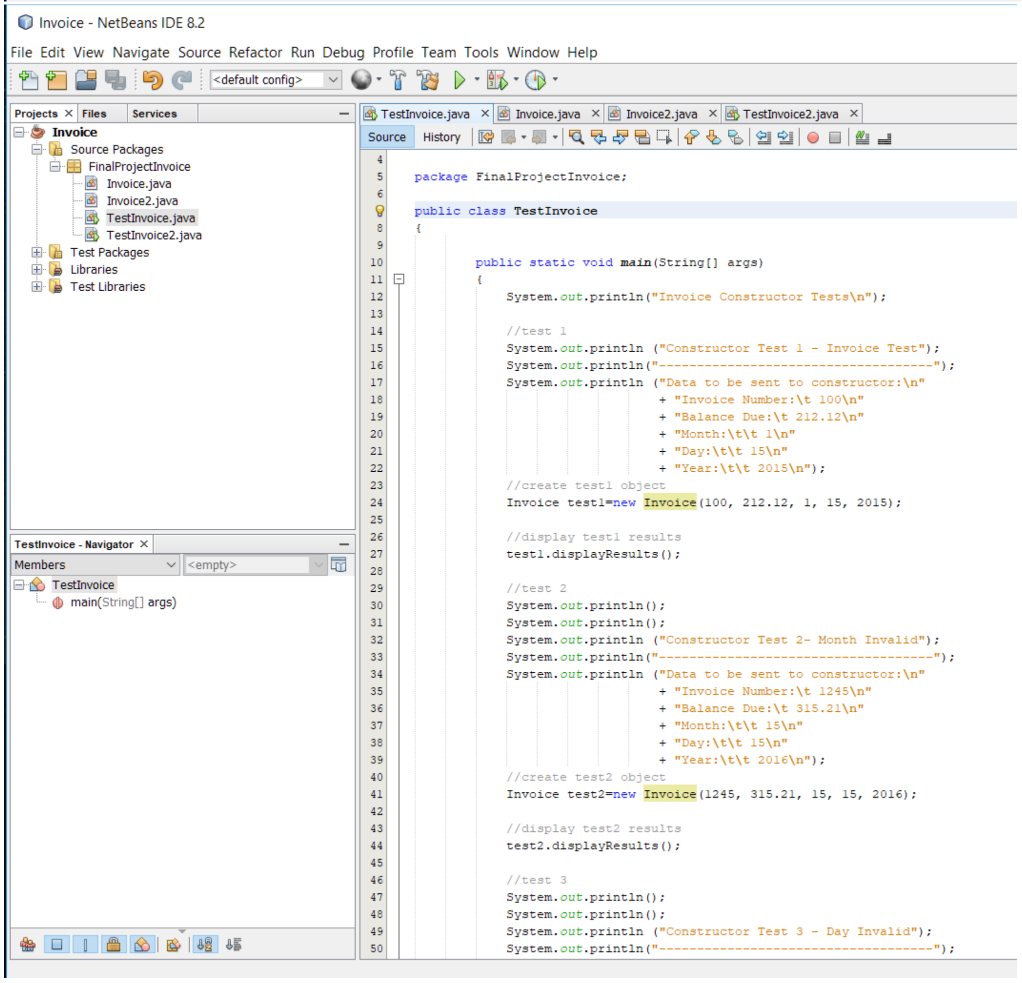
```
Constructor Test 3 - Day Invalid
-----------------------------------
Data to be sent to constructor:
Invoice Number:              4588
Balance Due:                 825.72
Month:                        5
Day:                          45
Year:                        2012

Test Results:
-----------------------------------
Invoice Number:                             4588
Balance Due:                          $    825.72
Due Date:                              05-00-2012


Constructor Test 4 - Year Out of Range
-----------------------------------
Data to be sent to constructor:
Invoice Number:              7251
Balance Due:                 129.92
Month:                       7
Day:                         21
Year:                        2010

Test Results:
-----------------------------------
Invoice Number:                             7251
Balance Due:                          $    129.92
Due Date:                              07-21-0000


Constructor Test 5 - All Data Valid
-----------------------------------
Data to be sent to constructor:
Invoice Number:              3269
Balance Due:                 719.33
Month:                       9
Day:                         28
Year:                        2013

Test Results:
-----------------------------------
Invoice Number:                             3269
Balance Due:                          $    719.33
Due Date:                              09-28-2013
```


Invoice - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Output - Invoice (run)
Output

```
run:
Invoice Constructor Tests

Constructor Test 1 - Invoice Test
------------------------------------
Data to be sent to constructor:
Invoice Number:  100
Balance Due:     212.12
Month:           1
Day:             15
Year:            2015

Test Results:
------------------------------------
Invoice Number:                   0
Balance Due:            $    212.12
Due Date:                01-15-2015


Constructor Test 2- Month Invalid
------------------------------------
Data to be sent to constructor:
Invoice Number:  1245
Balance Due:     315.21
Month:           15
Day:             15
Year:            2016

Test Results:
------------------------------------
Invoice Number:                1245
Balance Due:            $    315.21
Due Date:                00-15-2016


Constructor Test 3 - Day Invalid
------------------------------------
Data to be sent to constructor:
Invoice Number:  4588
Balance Due:     825.72
Month:           5
Day:             45
Year:            2012

Test Results:
------------------------------------
Invoice Number:                4588
Balance Due:            $    825.72
Due Date:                05-00-2012


Constructor Test 4 - Year Out of Range
------------------------------------
Data to be sent to constructor:
Invoice Number:  7251
```



```java
package FinalProjectInvoice;

public class Invoice
{
    private int invoiceNumber;      //field for invoice number
    private double balanceDue;      //field for balance due
    private int month;              //field for month
    private int day;                //field for day
    private int year;               //field for year

    public Invoice(int number, double balance, int month, int day, int year)
        {
            //force invoice number to be 0 if it's less than 1000
            if (number < 1000)
            {
                invoiceNumber = 0;
            }
            else
            {
                invoiceNumber = number;
            }

            //balance field
            balanceDue = balance;

            //if month is not between 1 and 12 force month to be 0
            if (month < 1 || month > 12)
            {
                this.month = 0;
            }
            else
            {
                this.month = month;
            }

            //if day is not between 1 and 31 force month to be 0
            if (day < 1 ||day > 31)
            {
                this.day = 0;
            }
            else
            {
                this.day = day;
            }
```



```java
package FinalProjectInvoice;

public class TestInvoice
{

        public static void main(String[] args)
        {
            System.out.println("Invoice Constructor Tests\n");

            //test 1
            System.out.println ("Constructor Test 1 - Invoice Test");
            System.out.println("-----------------------------------");
            System.out.println ("Data to be sent to constructor:\n"
                                    + "Invoice Number:\t 100\n"
                                    + "Balance Due:\t 212.12\n"
                                    + "Month:\t\t 1\n"
                                    + "Day:\t\t 15\n"
                                    + "Year:\t\t 2015\n");
            //create test1 object
            Invoice test1=new Invoice(100, 212.12, 1, 15, 2015);

            //display test1 results
            test1.displayResults();

            //test 2
            System.out.println();
            System.out.println();
            System.out.println ("Constructor Test 2- Month Invalid");
            System.out.println("-----------------------------------");
            System.out.println ("Data to be sent to constructor:\n"
                                    + "Invoice Number:\t 1245\n"
                                    + "Balance Due:\t 315.21\n"
                                    + "Month:\t\t 15\n"
                                    + "Day:\t\t 15\n"
                                    + "Year:\t\t 2016\n");
            //create test2 object
            Invoice test2=new Invoice(1245, 315.21, 15, 15, 2016);

            //display test2 results
            test2.displayResults();

            //test 3
            System.out.println();
            System.out.println();
            System.out.println ("Constructor Test 3 - Day Invalid");
            System.out.println("-----------------------------------");
```

```
//Mickie Blair
//Java I – CIST 2371
//Final Project Invoice - Invoice2 Class

package FinalProjectInvoice;

public class Invoice2
{

  private int invoiceNumber;                  //field for invoice number
  private double balanceDue;                  //field for balance due
  private int month;                          //field for month
  private int day;                            //field for day
  private int year;                           //field for year

  public Invoice2(int number, double balance, int month, int day, int year)
    {
      //force invoice number to be 0 if it's less than 1000
      if (number < 1000)
      {
        invoiceNumber = 0;
      }
      else
      {
        invoiceNumber = number;
      }

      //balance field
      balanceDue = balance;

      //month and day if statements
      if (month < 1 || month > 12)
        {
          this.month = 0;
          this.day = 0;
        }
      else if (month == 2)
        {
          this.month = month;

          if (day>28)
            {
              this.day = 28;
            }
          else
            {
              this.day = day;
            }
        }
      else if (month == 1 || month == 3 || month == 5 || month == 7
             || month == 8 || month == 10 || month == 12)
          {
             this.month = month;

             if (day>31)
                {
                   this.day = 31;
                }
             else
                {
                   this.day = day;
                }
          }

        else if (month == 4 || month == 6 || month == 9 || month == 11)

          {
             this.month = month;

             if (day>30)
                {
                   this.day = 30;
                }
             else
                {
                   this.day = day;
                }
          }


        //if year is not between 2011 and 2017 force month to be 0
        if (year < 2011 || year > 2017)
        {
           this.year = 0;
        }
        else
        {
           this.year = year;
        }

     }

  public void displayResults()
  {
     System.out.println("Test Results:");
     System.out.println("-------------------------------------");
     System.out.printf("Invoice Number: \t%11d\n", invoiceNumber);
     System.out.printf("Balance Due:\t\t$%10.2f\n", balanceDue);
     System.out.printf("Due Date: \t\t %02d-%02d-%04d\n", month, day, year);
  }
}
```

```
//Mickie Blair
//Java I – CIST 2371
//Final Project Invoice - TestInvoice2 Class

package FinalProjectInvoice;

public class TestInvoice2
{
    public static void main(String[] args)
    {
        System.out.println("Invoice2 Constructor Tests\n");

        //test 1
        System.out.println ("Constructor Test 1 - Invoice Test");
        System.out.println("-----------------------------------");
        System.out.println ("Data to be sent to constructor:\n"
                    + "Invoice Number:\t 100\n"
                    + "Balance Due:\t 212.12\n"
                    + "Month:\t\t 1\n"
                    + "Day:\t\t 15\n"
                    + "Year:\t\t 2015\n");
        //create test1 object
        Invoice2 test1=new Invoice2(100, 212.12, 1, 15, 2015);

        //display test1 results
        test1.displayResults();

        //test 2
        System.out.println();
        System.out.println();
        System.out.println ("Constructor Test 2 - Invalid Month");
        System.out.println("-----------------------------------");
        System.out.println ("Data to be sent to constructor:\n"
                    + "Invoice Number:\t 1245\n"
                    + "Balance Due:\t 315.21\n"
                    + "Month:\t\t 15\n"
                    + "Day:\t\t 15\n"
                    + "Year:\t\t 2016\n");
        //create test2 object
        Invoice2 test2=new Invoice2(1245, 315.21, 15, 15, 2016);

        //display test2 results
        test2.displayResults();

        //test 3
        System.out.println();
        System.out.println();
        System.out.println ("Constructor Test 3a - February Test - Days > 28");
        System.out.println("-----------------------------------");
        System.out.println ("Data to be sent to constructor:\n"
                    + "Invoice Number:\t 4588\n"
```

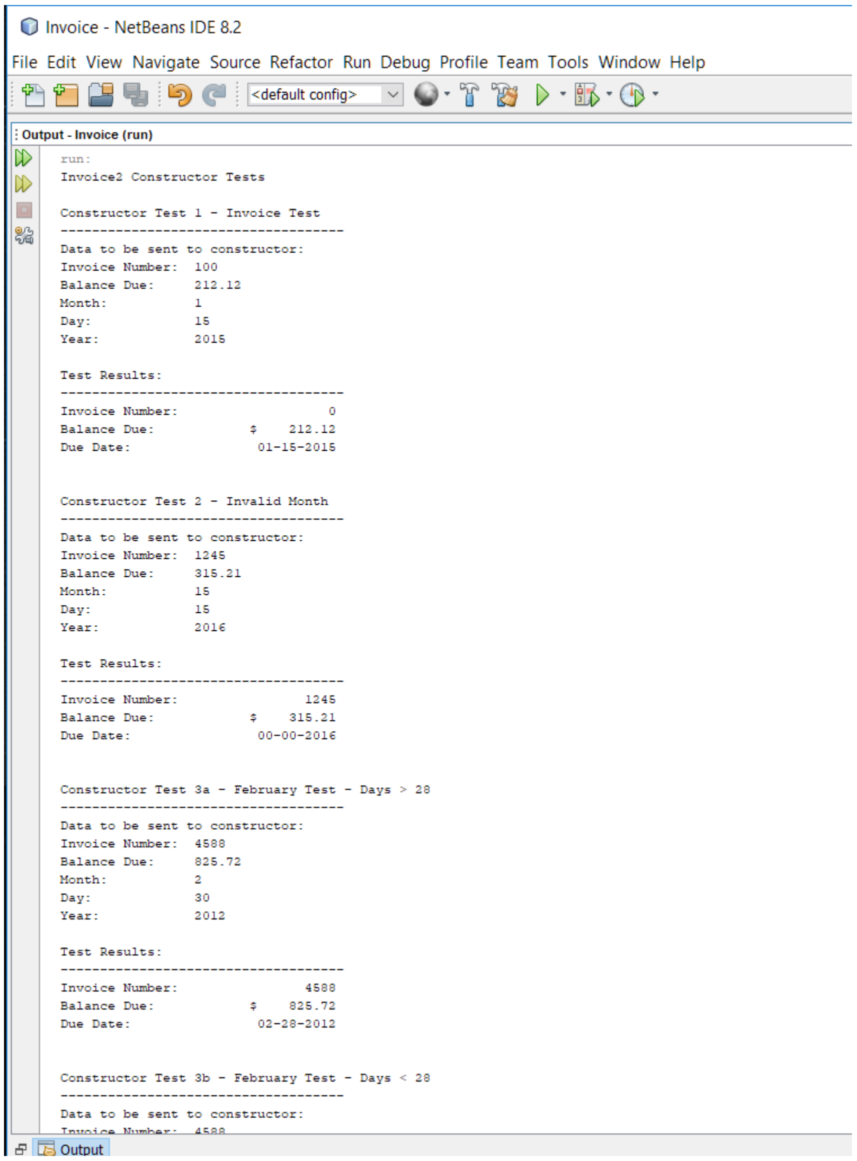
```
			+ "Balance Due:\t 825.72\n"
			+ "Month:\t\t 2\n"
			+ "Day:\t\t 30\n"
			+ "Year:\t\t 2012\n");
//create test3 object
Invoice2 test3a=new Invoice2(4588, 825.72, 2, 30, 2012);

//display test3 results
test3a.displayResults();

//test 3b
System.out.println();
System.out.println();
System.out.println ("Constructor Test 3b - February Test - Days < 28");
System.out.println("------------------------------------");
System.out.println ("Data to be sent to constructor:\n"
			+ "Invoice Number:\t 4588\n"
			+ "Balance Due:\t 825.72\n"
			+ "Month:\t\t 2\n"
			+ "Day:\t\t 21\n"
			+ "Year:\t\t 2012\n");
//create test3 object
Invoice2 test3b=new Invoice2(4588, 825.72, 2, 21, 2012);

//display test3 results
test3b.displayResults();



//test 4a
System.out.println();
System.out.println();
System.out.println ("Constructor Test 4a - 31 day Months - Days > 31");
System.out.println("------------------------------------");
System.out.println ("Data to be sent to constructor:\n"
			+ "Invoice Number:\t 7251\n"
			+ "Balance Due:\t 129.92\n"
			+ "Month:\t\t 7\n"
			+ "Day:\t\t 35\n"
			+ "Year:\t\t 2014\n");
//create test4 object
Invoice2 test4a=new Invoice2(7251, 129.92, 7, 35, 2014);

//display test results
test4a.displayResults();

//test 4b
System.out.println();
System.out.println();
System.out.println ("Constructor Test 4b - 31 day Months - Days < 31");
System.out.println("------------------------------------");
System.out.println ("Data to be sent to constructor:\n"
                    + "Invoice Number:\t 7251\n"
                    + "Balance Due:\t 129.92\n"
                    + "Month:\t\t 7\n"
                    + "Day:\t\t 28\n"
                    + "Year:\t\t 2014\n");
        //create test4 object
        Invoice2 test4b=new Invoice2(7251, 129.92, 7, 28, 2014);

        //display test results
        test4b.displayResults();


        //test 5a
        System.out.println();
        System.out.println();
        System.out.println ("Constructor Test 5a - 30 day Months - Days > 30");
        System.out.println("-----------------------------------");
        System.out.println ("Data to be sent to constructor:\n"
                    + "Invoice Number:\t 3269\n"
                    + "Balance Due:\t 719.33\n"
                    + "Month:\t\t 9\n"
                    + "Day:\t\t 31\n"
                    + "Year:\t\t 2013\n");
        //create test5 object
        Invoice2 test5a=new Invoice2(3269, 719.33, 9, 31, 2013);

        //display test results
        test5a.displayResults();

        //test 5b
        System.out.println();
        System.out.println();
        System.out.println ("Constructor Test 5b - 30 day Months - Days < 30");
        System.out.println("-----------------------------------");
        System.out.println ("Data to be sent to constructor:\n"
                    + "Invoice Number:\t 3269\n"
                    + "Balance Due:\t 719.33\n"
                    + "Month:\t\t 9\n"
                    + "Day:\t\t 27\n"
                    + "Year:\t\t 2013\n");
        //create test5 object
        Invoice2 test5b=new Invoice2(3269, 719.33, 9, 27, 2013);

        //display test results
        test5b.displayResults();

    }

}
```

OUTPUT

```
Invoice2 Constructor Tests

Constructor Test 1 - Invoice Test
-----------------------------------
Data to be sent to constructor:
Invoice Number:         100
Balance Due:            212.12
Month:                  1
Day:                    15
Year:                   2015

Test Results:
-----------------------------------
Invoice Number:                  0
Balance Due:            $   212.12
Due Date:               01-15-2015


Constructor Test 2 - Invalid Month
-----------------------------------
Data to be sent to constructor:
Invoice Number:         1245
Balance Due:            315.21
Month:                  15
Day:                    15
Year:                   2016

Test Results:
-----------------------------------
Invoice Number:               1245
Balance Due:            $   315.21
Due Date:               00-00-2016


Constructor Test 3a - February Test - Days > 28
-----------------------------------
Data to be sent to constructor:
Invoice Number:         4588
Balance Due:            825.72
Month:                  2
Day:                    30
Year:                   2012

Test Results:
-----------------------------------
Invoice Number:               4588
Balance Due:            $   825.72
Due Date:               02-28-2012
```

```
Constructor Test 3b - February Test - Days < 28
-----------------------------------
Data to be sent to constructor:
Invoice Number:          4588
Balance Due:             825.72
Month:                   2
Day:                     21
Year:                    2012

Test Results:
-----------------------------------
Invoice Number:                4588
Balance Due:              $  825.72
Due Date:                02-21-2012


Constructor Test 4a - 31 day Months - Days > 31
-----------------------------------
Data to be sent to constructor:
Invoice Number:          7251
Balance Due:             129.92
Month:                   7
Day:                     35
Year:                    2014

Test Results:
-----------------------------------
Invoice Number:                7251
Balance Due:              $  129.92
Due Date:                07-31-2014


Constructor Test 4b - 31 day Months - Days < 31
-----------------------------------
Data to be sent to constructor:
Invoice Number:          7251
Balance Due:             129.92
Month:                   7
Day:                     28
Year:                    2014

Test Results:
-----------------------------------
Invoice Number:                7251
Balance Due:              $  129.92
Due Date:                07-28-2014


Constructor Test 5a - 30 day Months - Days > 30
-----------------------------------
Data to be sent to constructor:
Invoice Number:          3269
Balance Due:             719.33
Month:                   9
Day:                     31
Year:                    2013
```

```
Test Results:
------------------------------------
Invoice Number:                  3269
Balance Due:                 $   719.33
Due Date:                   09-30-2013


Constructor Test 5b - 30 day Months - Days < 30
------------------------------------
Data to be sent to constructor:
Invoice Number:             3269
Balance Due:                719.33
Month:                      9
Day:                        27
Year:                       2013

Test Results:
------------------------------------
Invoice Number:                  3269
Balance Due:                 $   719.33
Due Date:                   09-27-2013
```



Invoice - NetBeans IDE 8.2

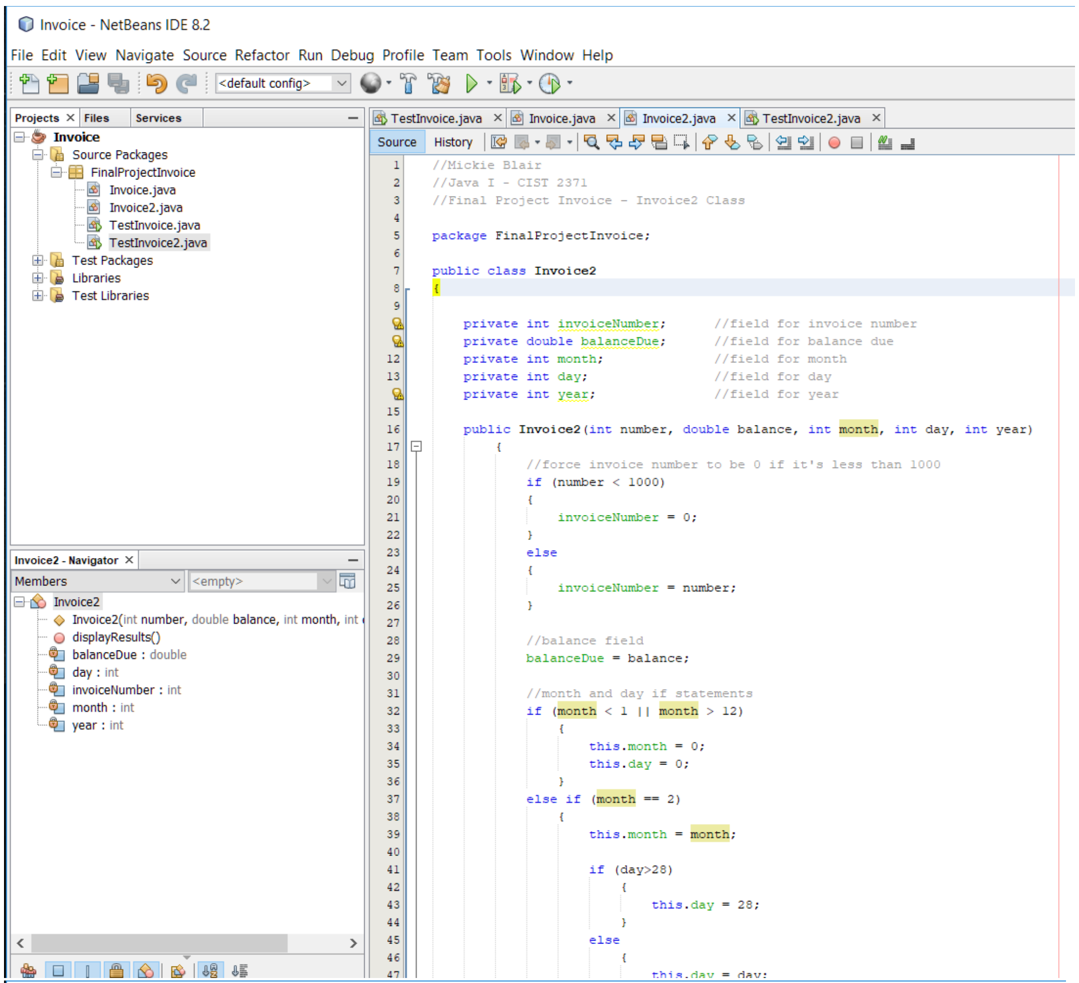
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

<default config>

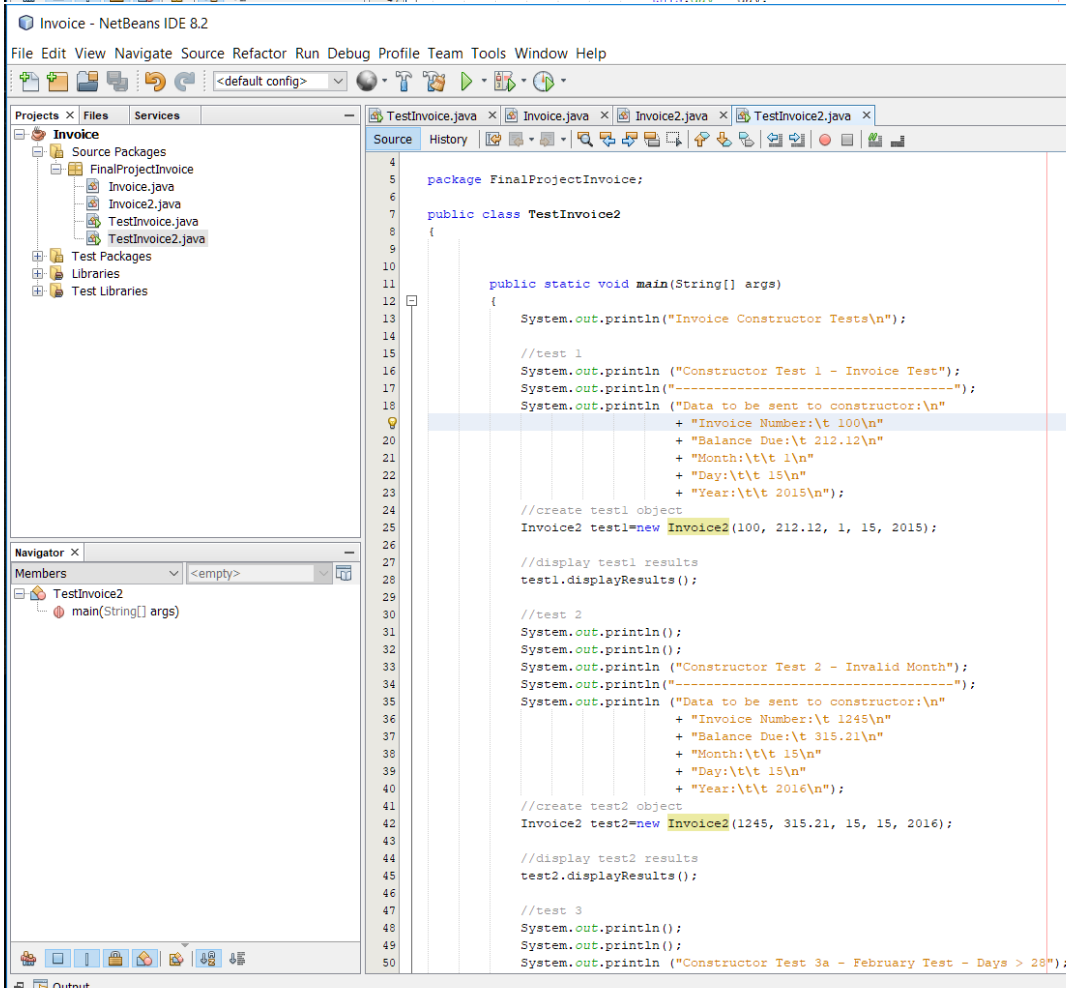
Output - Invoice (run)

```
run:
Invoice2 Constructor Tests

Constructor Test 1 - Invoice Test
------------------------------------
Data to be sent to constructor:
Invoice Number:  100
Balance Due:     212.12
Month:           1
Day:             15
Year:            2015

Test Results:
------------------------------------
Invoice Number:                    0
Balance Due:           $    212.12
Due Date:               01-15-2015


Constructor Test 2 - Invalid Month
------------------------------------
Data to be sent to constructor:
Invoice Number:  1245
Balance Due:     315.21
Month:           15
Day:             15
Year:            2016

Test Results:
------------------------------------
Invoice Number:                 1245
Balance Due:           $    315.21
Due Date:               00-00-2016


Constructor Test 3a - February Test - Days > 28
------------------------------------
Data to be sent to constructor:
Invoice Number:  4588
Balance Due:     825.72
Month:           2
Day:             30
Year:            2012

Test Results:
------------------------------------
Invoice Number:                 4588
Balance Due:           $    825.72
Due Date:               02-28-2012


Constructor Test 3b - February Test - Days < 28
------------------------------------
Data to be sent to constructor:
Invoice Number:  4588
```

Output



```
//Mickie Blair
//Java I - CIST 2371
//Final Project Invoice - Invoice2 Class

package FinalProjectInvoice;

public class Invoice2
{

    private int invoiceNumber;      //field for invoice number
    private double balanceDue;      //field for balance due
    private int month;              //field for month
    private int day;                //field for day
    private int year;               //field for year

    public Invoice2(int number, double balance, int month, int day, int year)
        {
            //force invoice number to be 0 if it's less than 1000
            if (number < 1000)
            {
                invoiceNumber = 0;
            }
            else
            {
                invoiceNumber = number;
            }

            //balance field
            balanceDue = balance;

            //month and day if statements
            if (month < 1 || month > 12)
                {
                    this.month = 0;
                    this.day = 0;
                }
            else if (month == 2)
                {
                    this.month = month;

                    if (day>28)
                        {
                            this.day = 28;
                        }
                    else
                        {
                            this.day = day;
```



```

package FinalProjectInvoice;

public class TestInvoice2
{


        public static void main(String[] args)
        {
            System.out.println("Invoice Constructor Tests\n");

            //test 1
            System.out.println ("Constructor Test 1 - Invoice Test");
            System.out.println("-------------------------------------");
            System.out.println ("Data to be sent to constructor:\n"
                                + "Invoice Number:\t 100\n"
                                + "Balance Due:\t 212.12\n"
                                + "Month:\t\t 1\n"
                                + "Day:\t\t 15\n"
                                + "Year:\t\t 2015\n");
            //create test1 object
            Invoice2 test1=new Invoice2(100, 212.12, 1, 15, 2015);

            //display test1 results
            test1.displayResults();

            //test 2
            System.out.println();
            System.out.println();
            System.out.println ("Constructor Test 2 - Invalid Month");
            System.out.println("-------------------------------------");
            System.out.println ("Data to be sent to constructor:\n"
                                + "Invoice Number:\t 1245\n"
                                + "Balance Due:\t 315.21\n"
                                + "Month:\t\t 15\n"
                                + "Day:\t\t 15\n"
                                + "Year:\t\t 2016\n");
            //create test2 object
            Invoice2 test2=new Invoice2(1245, 315.21, 15, 15, 2016);

            //display test2 results
            test2.displayResults();

            //test 3
            System.out.println();
            System.out.println();
            System.out.println ("Constructor Test 3a - February Test - Days > 28");
```

```
//Mickie Blair
//Java I – CIST 2371
//Final Project - Swimming Pool Class

package SwimmingPool;

public class SwimmingPool
{
    private double length;
    private double width;
    private double depth;
    private double fillRate;
    private double drainRate;
    private final double GAL_PER_FT3 = 7.5;  //gallons of water in a cubic foot
    private double capacity;

    /**
     * Constructor
     * @param length Length of Pool
     * @param width Width of Pool
     * @param depth Depth of Pool
     * @param fillRate Fill rate in gpm
     * @param drainRate Drain rate in gpm
     */
    public SwimmingPool(double length, double width, double depth,
                        double fillRate, double drainRate)
    {
        this.length = length;
        this.width = width;
        this.depth = depth;
        this.fillRate = fillRate;
        this.drainRate = drainRate;
        this.capacity = length * width * depth * GAL_PER_FT3;
    }

    //return pool's water capacity
    public double getPoolCapacity()
        {
            return capacity;
        }

    /**
     *
     * @return Max time to fill
     */
    public double getMaxTimeToFill()
    {
        return (capacity/fillRate)/60;
    }

    /**
     *
     * @return Max Time to Drain
     */
    public double getMaxTimeToDrain()
    {
        return (capacity/drainRate)/60;
}

/**
 * Calculate the gallons of water needed to adjust fill percentage
 * @param current Current Percentage Full of Pool
 * @param target Target Percentage Full of Pool
 * @return Absolute value of water needed to adjust the fill level
 */
public double calcGallonsofWater(double current, double target)
   {
      return Math.abs(((target - current)/100) * capacity);
   }

/**
 * Calculate Time to Fill
 * @param needed Gallons to add to adjust the level
 * @return Hours to Fill
 */
public double calcTimeToFill(double needed)
   {
      return (needed/fillRate)/60;
   }

/**
 * Calculate Time To Drain
 * @param remove Gallons to drain to adjust the level
 * @return
 */
public double calcTimeToDrain(double remove)
   {
      return (remove/drainRate)/60;
   }

/**
 * Calculate Gallons added during filling time
 * @param fillTime hours the user would like to run water
 * @return Gallons added in the time inputted
 */
public double calcGallonsFill(double fillTime)
{
   return (fillRate* 60) * fillTime;
}

 /**
 * Calculate Gallons removed during drain time
 * @param drainTime hours the user would like to run water
 * @return Gallons added in the time inputted
 */
public double calcGallonsDrain(double drainTime)
{
   return (drainRate* 60) * drainTime;
}

/**
 *
 * @param percentFull Percentage Full
 * @return  Gallons of water in the pool
 */
public double getGallonsInPool(double percentFull)
   {
      return (percentFull/100) * capacity;
   }

//to String
public String toString ()
{

   String str = String.format("Pool Information\n\n"
   + "Pool Length:\t\t%8.1f feet\n"
   + "Pool Width:\t\t%8.1f feet\n"
   + "Pool Average Depth:\t%8.1f feet\n"
   + "Rate of Fill:\t\t%8.1f gallons per minute\n"
   + "Drain Rate:\t\t%8.1f gallons per minute\n"
   + "Pool Capacity:\t\t%8.1f gallons", length, width, depth, fillRate,
                                   drainRate, capacity);
  return str;
}
}


//Mickie Blair
//Java I – CIST 2371
//Final Project - Swimming Pool Class Test Program

package SwimmingPool;

import javax.swing.JOptionPane;

public class SwimmingPoolDemo
{
   public static void main(String[] args)
   {
      String input;
      double lengthOfPool;
      double widthOfPool;
      double averageDepth;
      double poolFillRate;
      double poolDrainRate;
      double poolCapacity;
      int menuChoice;

      //Ask the user for pool dimensions, fill rate, and drain rate
      input = JOptionPane.showInputDialog("Length of Pool in feet:");
      lengthOfPool=Double.parseDouble(input);

      input = JOptionPane.showInputDialog("Width of Pool in feet:");
      widthOfPool=Double.parseDouble(input);

      input = JOptionPane.showInputDialog("Average Depth of Pool in feet:");
      averageDepth=Double.parseDouble(input);

      input = JOptionPane.showInputDialog("Fill Rate in gallons per minute:");
      poolFillRate=Double.parseDouble(input);
```

```
    input = JOptionPane.showInputDialog("Drain Rate in gallons per minute:");
    poolDrainRate=Double.parseDouble(input);

    //create a new pool object
    SwimmingPool test1 = new SwimmingPool(lengthOfPool, widthOfPool,
                                averageDepth, poolFillRate,
                                poolDrainRate);

    //menu for determining next steps
    input = JOptionPane.showInputDialog("Program Menu Options\n\n"
       + " 1. Determine the amount of water and time needed "
           + "adjust the level in the pool.\n"
       + " 2. Add water for a specific amount of time.\n"
       + " 3. Drain water for a specific amount of time.\n\n"
       + "Enter Menu Number:   ");

    menuChoice = Integer.parseInt(input);

    //switch for menu
    switch(menuChoice)
    {
       case 1: calcGallonsTime(test1);
           break;
       case 2: calcUsingTimeFill(test1);
           break;
       case 3: calcUsingTimeDrain(test1);
           break;
    }

  System.exit(0);
 }

 /**
  * Determine the amount of water and time needed for pool filling
  * and draining
  * @param test Swimming Pool test object
  *
  */
 public static void calcGallonsTime(SwimmingPool test)
 {
    String input;                   //variable for JOptionPane Input
    double currentPercent;          //percentage of water currently in
    double targetPercent;           //target percentage of water in the pool
    double gallonsForAdjust;         //gallons needed to adjust the level

    //Ask the user the current percentage of water in pool
    input = JOptionPane.showInputDialog("How much water is currently in the pool?\n\n"
                          + "Examples:\n0 for an empty pool\n"
                          + "50 if the pool is 50% full\n"
                          + "100 if the pool is 100% full\n\n"
                          + "Enter current percentage full:  ");

    currentPercent=Double.parseDouble(input);


    //Ask the user how full they would like the pool
    input = JOptionPane.showInputDialog("How much water do you want in the pool?\n\n"
                              + "Example:\n0 for an empty pool\n"
                              + "50 for 50% full\n"
                              + "100 for 100% full\n\n"
                              + "Enter target percentage full:  ");

    targetPercent=Double.parseDouble(input);

    //calculate amount to fill
    gallonsForAdjust = test.calcGallonsofWater(currentPercent, targetPercent);

    //display pool info and results
    System.out.println(test);
    System.out.printf("\nTo adjust the pool from %.1f%% to %.1f%% "
                + " full:\n" ,currentPercent, targetPercent);

    //if statements for fill or drain
    if (currentPercent<targetPercent)
    {
      System.out.printf("\nWater To Add:\t\t%8.1f gallons\n", gallonsForAdjust);
      System.out.printf("Time to Fill:\t\t%8.1f hours \n\n",
                    test.calcTimeToFill(gallonsForAdjust));
    }
    else if (currentPercent>targetPercent)
    {
      System.out.printf("\nWater to Drain:\t\t%8.1f gallons\n", gallonsForAdjust);
      System.out.printf("Time to Drain:\t\t%8.1f hours \n\n",
                    test.calcTimeToDrain(gallonsForAdjust));
    }
  }

  /**
  * Calculate how much water is filled in a specific amount of time
  * @param test Swimming Pool Object
  */
  public static void calcUsingTimeFill(SwimmingPool test)
  {
    String input;                 //variable for JOptionPane Input
    double initialPercent;        //percentage of water currently in pool
    double hours;                 //hours to fill or drain
    double gallonsAdded;          //gallons added
    double initialGallons;        //initial gallons in pool
    double endPercent;            //ending percent full


    //Ask the user the current percentage of water in pool
    input = JOptionPane.showInputDialog("How much water is currently in the pool?\n\n"
                              + "Examples:\n0 for an empty pool\n"
                              + "50 if the pool is 50% full\n"
                              + "100 if the pool is 100% full\n\n"
                              + "Enter current percentage full:  ");

    initialPercent=Double.parseDouble(input);
    //calculate initial gallons in pool
    initialGallons=test.getGallonsInPool(initialPercent);

    //Ask the user how long they would like to fill
    input = JOptionPane.showInputDialog("Enter the hours you plan on leaving\n"
        + "the water on to fill the pool:  ");

    hours=Double.parseDouble(input);

    //display results
    if (hours > test.getMaxTimeToFill())
    {
      System.out.println(test);

      System.out.printf("\nInitially (%.1f%% full), the pool has %.1f gallons of "
                  + "water. \n",initialPercent, initialGallons);

      System.out.println("\nThe time entered is greater than needed.");

      //calculate amount added
      gallonsAdded = test.getPoolCapacity()- initialGallons ;

      System.out.printf("\nThe Pool will be 100%% full in %.1f hours\n",
                   test.getMaxTimeToFill());

      System.out.printf("\nThe amount added was %.1f gallons.\n", gallonsAdded);

    }

    else
    {
      System.out.println(test);

      System.out.printf("\nInitially (%.1f%% full), the pool has %.1f gallons of "
                  + "water. \n",initialPercent, initialGallons);

      //calculate amount added in time period
      gallonsAdded = test.calcGallonsFill(hours);

      //calculate percent full after time period
      endPercent = ((initialGallons + gallonsAdded)/test.getPoolCapacity())*100;

      System.out.printf("\nDuring %.1f hours of filling, %.1f gallons "
          + "will be added.\n", hours, gallonsAdded);

      System.out.printf("\nThe Pool will then be %.1f %% full.\n", endPercent);

    }
  }
```

```
/**
  * Calculate how much water is removed in a specific amount of time
  * @param test Swimming Pool Object
  */
 public static void calcUsingTimeDrain(SwimmingPool test)
 {
   String input;              //variable for JOptionPane Input
   double initialPercent;     //percentage of water currently in pool
   double hours;              //hours to fill or drain
   double gallonsRemoved;     //gallons added
   double initialGallons;     //initial gallons in pool
   double endPercent;         //ending percent full


   //Ask the user the current percentage of water in pool
   input = JOptionPane.showInputDialog("How much water is currently in the pool?\n\n"
                          + "Examples:\n0 for an empty pool\n"
                          + "50 if the pool is 50% full\n"
                          + "100 if the pool is 100% full\n\n"
                          + "Enter current percentage full:  ");

   initialPercent=Double.parseDouble(input);

   //calculate initial gallons in pool
   initialGallons=test.getGallonsInPool(initialPercent);

   //Ask the user how long they would like to drain
   input = JOptionPane.showInputDialog("Enter the hours you plan on draining\n"
         + "the water from the pool:  ");

   hours=Double.parseDouble(input);

   //using max time to fill let user know
   if (hours > test.getMaxTimeToDrain())
   {
     System.out.println(test);

     System.out.printf("\nInitially (%.1f%% full), the pool has %.1f gallons of "
               + "water. \n",initialPercent, initialGallons);

     System.out.println("\nThe time entered is greater than needed.");

     //calculate amount added
     gallonsRemoved = initialGallons ;

     System.out.printf("\nThe Pool will be empty in %.1f hours\n",
                test.getMaxTimeToDrain());

     System.out.printf("\nThe amount drained was %.1f gallons.\n", gallonsRemoved);
  }

   else
   {
     System.out.println(test);

     System.out.printf("\nInitially (%.1f%% full), the pool has %.1f gallons of "
               + "water. \n",initialPercent, initialGallons);
```

```
            //calculate amount added in time period
            gallonsRemoved = test.calcGallonsDrain(hours);

            //calculate percent full after time period
            endPercent = ((initialGallons - gallonsRemoved)/test.getPoolCapacity())*100;

            System.out.printf("\nDuring %.1f hours of draining, %.1f gallons "
                + "will be removed.\n", hours, gallonsRemoved);

            System.out.printf("\nThe Pool will then be only %.1f %% full.\n", endPercent);

        }
    }
}
```

OUTPUT

Program run to fill pool

```
Pool Information

Pool Length:              10.0 feet
Pool Width:               20.0 feet
Pool Average Depth:        5.0 feet
Rate of Fill:              5.0 gallons per minute
Drain Rate:               10.0 gallons per minute
Pool Capacity:          7500.0 gallons

To adjust the pool from 20.0% to 100.0%  full:

Water To Add:           6000.0 gallons
Time to Fill:             20.0 hours
```

Program run to drain pool

```
Pool Information

Pool Length:              15.0 feet
Pool Width:               12.0 feet
Pool Average Depth:        4.0 feet
Rate of Fill:              7.0 gallons per minute
Drain Rate:               10.0 gallons per minute
Pool Capacity:          5400.0 gallons

To adjust the pool from 100.0% to 20.0%  full:

Water to Drain:         4320.0 gallons
Time to Drain:             7.2 hours
```

Program run to fill for a specific amount of time

```
Pool Information

Pool Length:                10.0 feet
Pool Width:                 20.0 feet
Pool Average Depth:          3.5 feet
Rate of Fill:                8.0 gallons per minute
Drain Rate:                 12.0 gallons per minute
Pool Capacity:            5250.0 gallons

Initially (20.0% full), the pool has 1050.0 gallons of water.

During 4.0 hours of filling, 1920.0 gallons will be added.

The Pool will then be 56.6 % full.
```

Program run to fill for a specific amount of time (greater than time to fill)

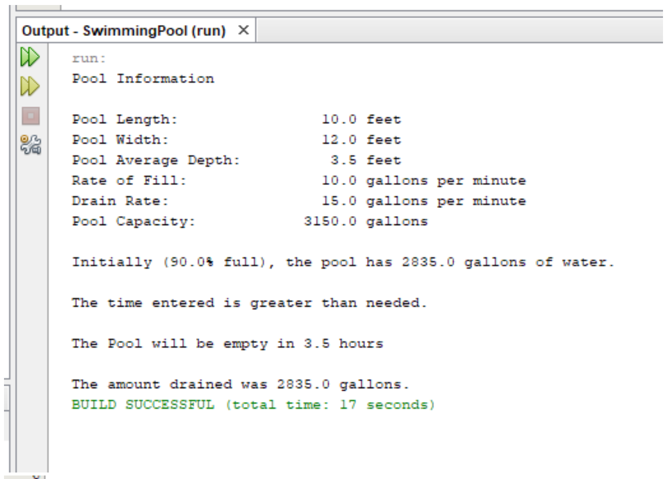
```
Pool Information

Pool Length:              10.0 feet
Pool Width:                8.0 feet
Pool Average Depth:        4.0 feet
Rate of Fill:             10.0 gallons per minute
Drain Rate:               15.0 gallons per minute
Pool Capacity:          2400.0 gallons

Initially (10.0% full), the pool has 240.0 gallons of water.

The time entered is greater than needed.

The Pool will be 100% full in 4.0 hours

The amount added was 2160.0 gallons.
```

Program run to drain for a specific amount of time

```
Pool Information

Pool Length:              10.0 feet
Pool Width:               10.0 feet
Pool Average Depth:       15.0 feet
Rate of Fill:              5.0 gallons per minute
Drain Rate:               10.0 gallons per minute
Pool Capacity:         11250.0 gallons

Initially (90.0% full), the pool has 10125.0 gallons of water.

During 2.0 hours of draining, 1200.0 gallons will be removed.

The Pool will then be only 79.3 % full.
```

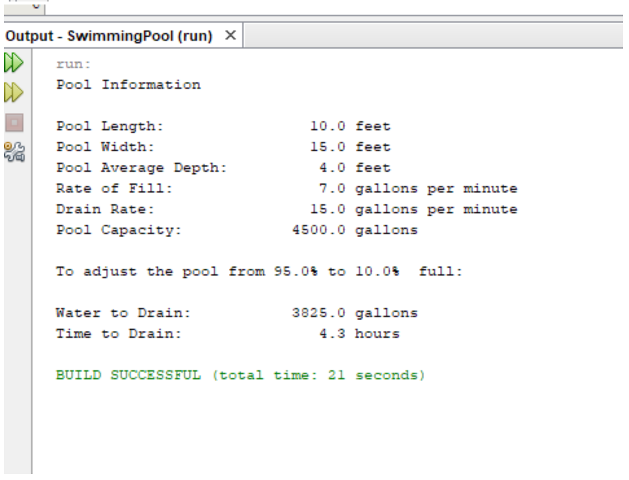
Program run to drain for a specific amount of time (greater than time to drain)

```
Pool Information

Pool Length:                  10.0 feet
Pool Width:                   12.0 feet
Pool Average Depth:            3.5 feet
Rate of Fill:                 10.0 gallons per minute
Drain Rate:                   15.0 gallons per minute
Pool Capacity:              3150.0 gallons

Initially (90.0% full), the pool has 2835.0 gallons of water.

The time entered is greater than needed.

The Pool will be empty in 3.5 hours

The amount drained was 2835.0 gallons.
```



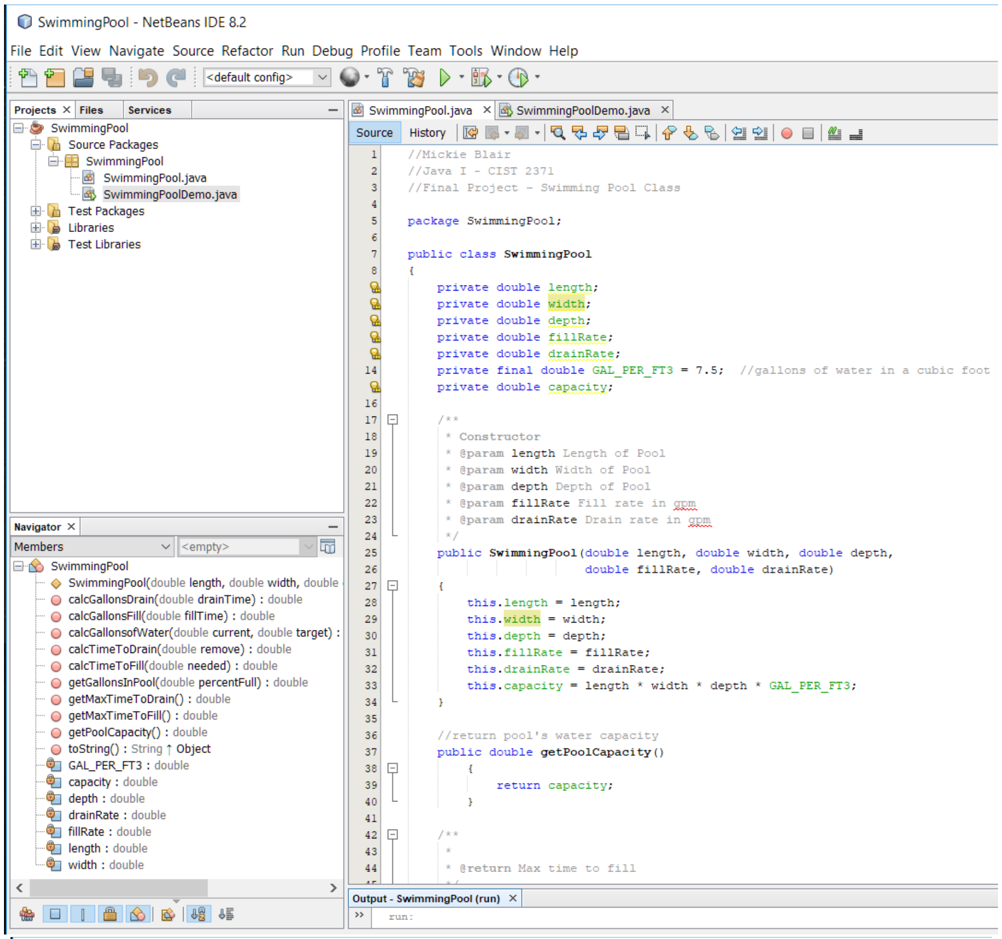
Output - SwimmingPool (run)

```
run:
Pool Information

Pool Length:                  10.0 feet
Pool Width:                   12.0 feet
Pool Average Depth:            3.5 feet
Rate of Fill:                 10.0 gallons per minute
Drain Rate:                   15.0 gallons per minute
Pool Capacity:              3150.0 gallons

Initially (90.0% full), the pool has 2835.0 gallons of water.

The time entered is greater than needed.

The Pool will be empty in 3.5 hours

The amount drained was 2835.0 gallons.
BUILD SUCCESSFUL (total time: 17 seconds)
```



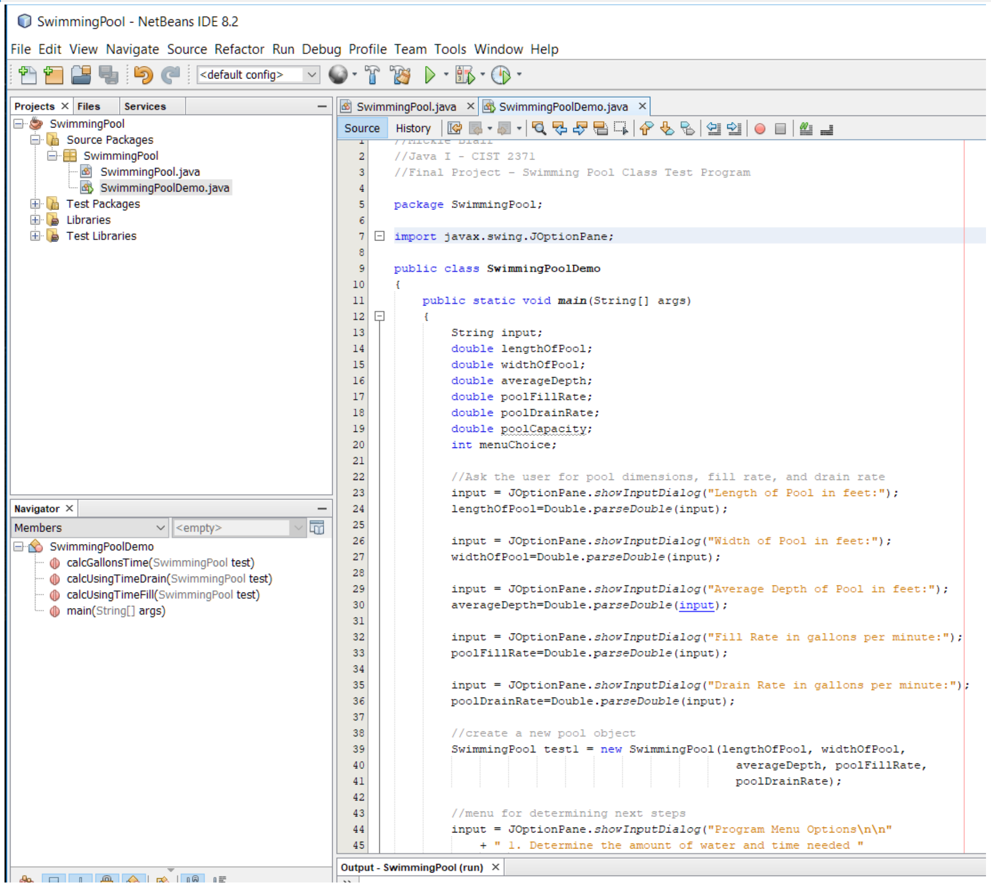
Output - SwimmingPool (run)

```
run:
Pool Information

Pool Length:                  10.0 feet
Pool Width:                   15.0 feet
Pool Average Depth:            4.0 feet
Rate of Fill:                  7.0 gallons per minute
Drain Rate:                   15.0 gallons per minute
Pool Capacity:              4500.0 gallons

To adjust the pool from 95.0% to 10.0%  full:

Water to Drain:             3825.0 gallons
Time to Drain:                 4.3 hours

BUILD SUCCESSFUL (total time: 21 seconds)
```


SwimmingPool - NetBeans IDE 8.2
File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
Projects Files Services
SwimmingPool
Source Packages
SwimmingPool
SwimmingPool.java
SwimmingPoolDemo.java
Test Packages
Libraries
Test Libraries
Navigator
Members
<empty>
SwimmingPool
SwimmingPool(double length, double width, double
calcGallonsDrain(double drainTime) : double
calcGallonsFill(double fillTime) : double
calcGallonsofWater(double current, double target) :
calcTimeToDrain(double remove) : double
calcTimeToFill(double needed) : double
getGallonsInPool(double percentFull) : double
getMaxTimeToDrain() : double
getMaxTimeToFill() : double
getPoolCapacity() : double
toString() : String ↑ Object
GAL_PER_FT3 : double
capacity : double
depth : double
drainRate : double
fillRate : double
length : double
width : double
SwimmingPool.java
SwimmingPoolDemo.java
Source History
Output - SwimmingPool (run)

```java
//Mickie Blair
//Java I - CIST 2371
//Final Project - Swimming Pool Class

package SwimmingPool;

public class SwimmingPool
{
    private double length;
    private double width;
    private double depth;
    private double fillRate;
    private double drainRate;
    private final double GAL_PER_FT3 = 7.5;  //gallons of water in a cubic foot
    private double capacity;

    /**
     * Constructor
     * @param length Length of Pool
     * @param width Width of Pool
     * @param depth Depth of Pool
     * @param fillRate Fill rate in gpm
     * @param drainRate Drain rate in gpm
     */
    public SwimmingPool(double length, double width, double depth,
                        double fillRate, double drainRate)
    {
        this.length = length;
        this.width = width;
        this.depth = depth;
        this.fillRate = fillRate;
        this.drainRate = drainRate;
        this.capacity = length * width * depth * GAL_PER_FT3;
    }

    //return pool's water capacity
    public double getPoolCapacity()
        {
            return capacity;
        }

    /**
     *
     * @return Max time to fill
```

```
run:
```



```java
//Java I - CIST 2371
//Final Project - Swimming Pool Class Test Program

package SwimmingPool;

import javax.swing.JOptionPane;

public class SwimmingPoolDemo
{
    public static void main(String[] args)
    {
        String input;
        double lengthOfPool;
        double widthOfPool;
        double averageDepth;
        double poolFillRate;
        double poolDrainRate;
        double poolCapacity;
        int menuChoice;

        //Ask the user for pool dimensions, fill rate, and drain rate
        input = JOptionPane.showInputDialog("Length of Pool in feet:");
        lengthOfPool=Double.parseDouble(input);

        input = JOptionPane.showInputDialog("Width of Pool in feet:");
        widthOfPool=Double.parseDouble(input);

        input = JOptionPane.showInputDialog("Average Depth of Pool in feet:");
        averageDepth=Double.parseDouble(input);

        input = JOptionPane.showInputDialog("Fill Rate in gallons per minute:");
        poolFillRate=Double.parseDouble(input);

        input = JOptionPane.showInputDialog("Drain Rate in gallons per minute:");
        poolDrainRate=Double.parseDouble(input);

        //create a new pool object
        SwimmingPool test1 = new SwimmingPool(lengthOfPool, widthOfPool,
                                              averageDepth, poolFillRate,
                                              poolDrainRate);

        //menu for determining next steps
        input = JOptionPane.showInputDialog("Program Menu Options\n\n"
            + " 1. Determine the amount of water and time needed "
```

```
//Mickie Blair
//Java I – CIST 2371
//Final Project - Person Class
//superclass

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class Person
{
   private String firstName;       //first name
   private String lastName;        //last name
   private String streetAddress;   //street address
   private int zipCode;            //zip code
   private String phoneNumber;     //phone number

   //set person data
   public void setPersonData()
   {
      firstName = JOptionPane.showInputDialog("Enter the First Name:");

      lastName = JOptionPane.showInputDialog("Enter the Last Name:");

      streetAddress = JOptionPane.showInputDialog("Enter the Street Address:");

      String input= JOptionPane.showInputDialog("Enter the ZipCode:");
      zipCode = Integer.parseInt(input);

      phoneNumber= JOptionPane.showInputDialog("Enter the Phone Number:");
   }

   //display results on a single line
   public void displayPersonData()
   {
      String fullName = firstName + " " + lastName;
      System.out.printf("\n%-20s", fullName);
      System.out.printf("%-25s", streetAddress);
      System.out.printf("%-12d", zipCode);
      System.out.printf("%-15s", phoneNumber);
   }
}



//Mickie Blair
//Java I – CIST 2371
//Final Project - College Employee Class
//extends from person(subclass)

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class CollegeEmployee extends Person
```

```
{
   private String socialSecurityNumber;          //social security number
   private double annualSalary;                   //annual salary
   private String deptName;                       //department name
   private static int empCount = 0;               //count of college Employees

   //set college employee data with a method to override Person Class method
   @Override
   public void setPersonData()
   {
      super.setPersonData();

      socialSecurityNumber = JOptionPane.showInputDialog("Enter the "
                        + "Employees's Social Security Number");

      String input= JOptionPane.showInputDialog("Enter the Employees's Annual"
            + " Salary:");
      annualSalary = Double.parseDouble(input);

      deptName = JOptionPane.showInputDialog("Enter the Employee's Department"
                        + " Name");

      empCount++;
   }

   /**
    *
    * @return Employee Count
    */
   public int getEmpCount()
   {
      return empCount;
   }

   //display results with a method to override Person Class method
   @Override
   public void displayPersonData()
   {
      super.displayPersonData();

      System.out.printf("%-15s", socialSecurityNumber);
      System.out.printf("$ %-15.2f", annualSalary);
      System.out.printf("%-15s", deptName);
   }
}
```

```
//Mickie Blair
//Java I – CIST 2371
//Final Project - Faculty Class
//extends from CollegeEmployee (subclass)

package FinalProjectPeople;

import javax.swing.JOptionPane;
```

```
public class Faculty extends CollegeEmployee
{
  private boolean tenured;       //boolean for tenure state (true = Yes)
  private static int fCount;      //count of faculty

  //set faculty data with a method to override Person Class/College Employee method
  @Override
  public void setPersonData()
  {
    super.setPersonData();
    String input = JOptionPane.showInputDialog("Is the Faculty member tenured?"
                + "(Enter Y or N)");

    input = input.toUpperCase();

    while (!input.equals("Y") && !input.equals("N"))
    {


      input = JOptionPane.showInputDialog("Invalid Response. Try Again.\n"
          + "Is the Faculty member tenured?"
                + "(Enter Y or N)");

      input = input.toUpperCase();
    }

    if (input.equals("Y"))
      {
        tenured = true;
      }

      if (input.equals("N"))
      {
        tenured = false;
      }
    fCount++;
  }

  /**
   *
   * @return Faculty Count
   */
  public int getFCount()
  {
    return fCount;
  }

  //display results with a method to override Person Class/College Employee method
  @Override
  public void displayPersonData()
  {
    super.displayPersonData();

    if (tenured)
        {
          System.out.printf("%-10s", "YES");
```

```
            }
        else
            {
               System.out.printf("%-10s", "NO");
            }
    }
}




//Mickie Blair
//Java I – CIST 2371
//Final Project - Student Class
//extends from person (subclass)

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class Student extends Person
{
    private String major;       //students major
    private double gpa;         //students GPA
    private static int sCount;   //count of students

    //set person data with a method to override Person Class method
    @Override
    public void setPersonData()
    {
        super.setPersonData();

        major = JOptionPane.showInputDialog("Enter the Student's Major:");

        String input= JOptionPane.showInputDialog("Enter the Student's GPA:");
        gpa = Double.parseDouble(input);

        sCount++;
    }

     /**
     *
     * @return Student Count
     */
    public int getSCount()
    {
        return sCount;
    }

     //display results with a method to override Person Class method
    @Override
    public void displayPersonData()
    {
        super.displayPersonData();

        System.out.printf("%-15s", major);
        System.out.printf("%-10.2f", gpa);
    }
}


//Mickie Blair
//Java I – CIST 2371
//Final Project - College List Class

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class CollegeList
{
   public static void main(String []args)
   {
      String menuChoice ="";          //to hold user choice
      final int NUM_EMPLOYEES = 4;    //constant for number of college Employees
      final int NUM_FACULTY = 3;    //constant for number of college Employees
      final int NUM_STUDENTS = 7;    //constant for number of college Employees
      int employeeCounter = 0;           //college employee counter
      int facultyCounter = 0;            //faculty counter
      int studentCounter = 0;            //student counter

      //Declare an array of four regular College employees
      CollegeEmployee[] collegeEmployeeArray = new CollegeEmployee[NUM_EMPLOYEES];

      //Declare an array of three faculty
      Faculty[] facultyArray = new Faculty[NUM_FACULTY];

      //Declare an array of seven students
      Student[] studentArray = new Student[NUM_STUDENTS];

      //loop to ask the user which type of person they would like to enter
      while (!menuChoice.equalsIgnoreCase("Q"))
      {
         String input = JOptionPane.showInputDialog("Data Entry Program\n\n"
            + "College Employee (Enter C)\n"
            + "Faculty (Enter F)\n"
            + "Student (Enter S)\n"
            + "To Quit Data Entry and Print Report(Enter Q)\n\n"
            + "Enter Selection:");

         menuChoice=input.toUpperCase();

         //switch statement for adding
         switch (menuChoice)
         {
            case "C": {
                     //if less than allowed create new object
                     if (employeeCounter < NUM_EMPLOYEES)
                     {
                        CollegeEmployee employee = new CollegeEmployee();

                        employee.setPersonData();

                        collegeEmployeeArray[employeeCounter] = employee;
```

```
                employeeCounter = employee.getEmpCount();
              }

              else
              {
                JOptionPane.showMessageDialog(null, "The number"
                + " of College Employees has reached the "
                + "maximum. Please Enter a different choice.");
              }
            }
      break;

    case "F": {
              //if less than allowed create new object
              if (facultyCounter < NUM_FACULTY)
              {
                Faculty collegeFaculty = new Faculty();
                collegeFaculty.setPersonData();

                facultyArray[facultyCounter] = collegeFaculty;

                facultyCounter = collegeFaculty.getFCount();
              }

              else
              {
                JOptionPane.showMessageDialog(null, "The number"
                + " of Faculty has reached the "
                + "maximum. Please Enter a different choice.");
              }
            }
      break;

    case "S": {
              //if less than allowed create new object
              if (studentCounter < NUM_STUDENTS)
              {
                Student collegeStudent = new Student();
                collegeStudent.setPersonData();

                studentArray[studentCounter] = collegeStudent;

                studentCounter = collegeStudent.getSCount();
              }

              else
              {
                JOptionPane.showMessageDialog(null, "The number"
                + " of Students has reached the "
                + "maximum. Please Enter a different choice.");
              }
            }
      break;

    case "Q":{
              //display report if the user quits
```

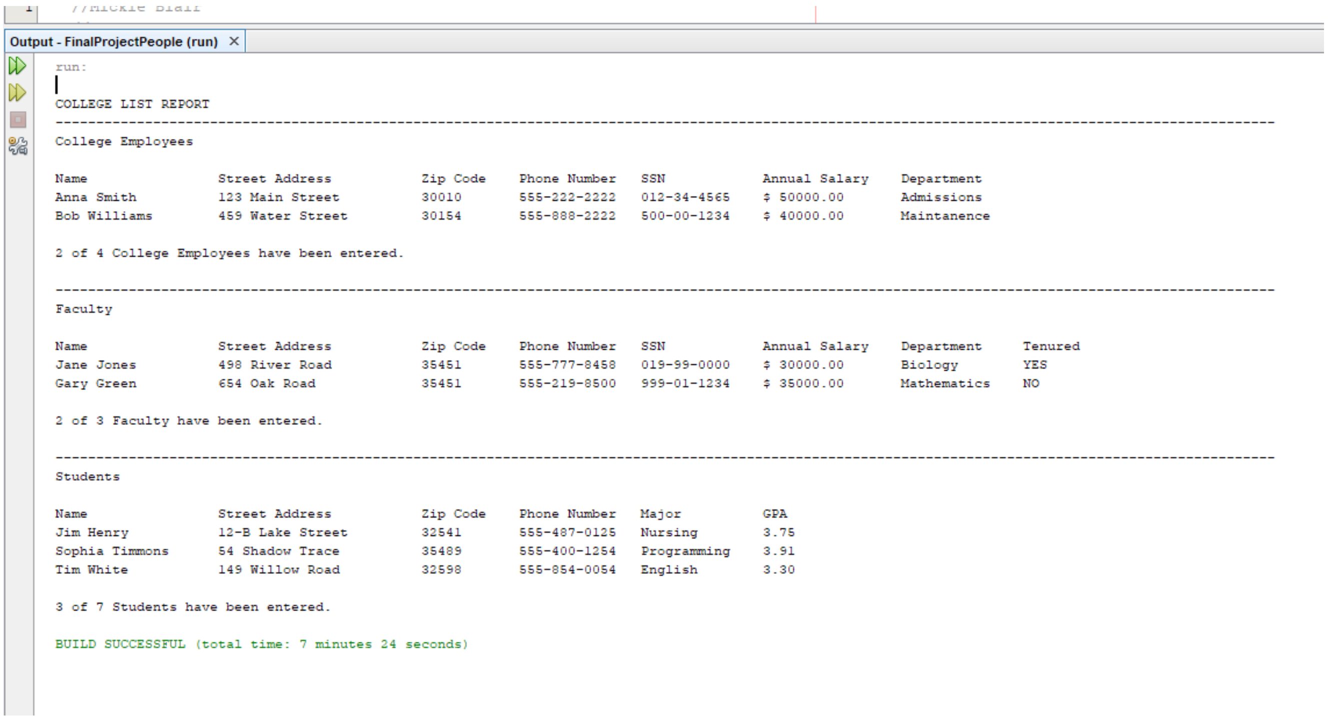
```
                JOptionPane.showMessageDialog(null, "Data Entry "
                + "Complete \n\n"
                + "College List Report will be displayed.");
            }
        break;

    default:{
                //display message if choice is invalid
                JOptionPane.showMessageDialog(null, "The selection"
                + " entered is invalid.\n\n"
                + "Please Enter a valid menu choice.");
            }

}


//display report
if (menuChoice.equals("Q"))
{
    //header for report
    System.out.println("\nCOLLEGE LIST REPORT");

    //display the college employees
    System.out.println("-------------------------------------------"
        + "----------------------------------------------------"
        + "----------------------------------------------------");

    System.out.println("College Employees\n");

    System.out.printf("%-20s%-25s%-12s%-15s%-15s%-17s%-15s",
        "Name", "Street Address", "Zip Code", "Phone Number",
        "SSN", "Annual Salary", "Department");

    //if less than needed
    if (employeeCounter < NUM_EMPLOYEES)
    {

        for ( int index = 0; index < employeeCounter; index++)
        {
            collegeEmployeeArray[index].displayPersonData();
        }

        System.out.printf("\n\n%d of %d College Employees have been "
            + "entered.\n", employeeCounter, NUM_EMPLOYEES);
    }

    //display all
    else
    {
        for ( int index = 0; index < employeeCounter; index++)
        {
            collegeEmployeeArray[index].displayPersonData();
        }
    }

    System.out.println();
```

```
//display the faculty
System.out.println("-------------------------------------------"
      + "----------------------------------------------------"
      + "----------------------------------------------------");

System.out.println("Faculty\n");

System.out.printf("%-20s%-25s%-12s%-15s%-15s%-17s%-15s%-10s",
      "Name", "Street Address", "Zip Code", "Phone Number",
      "SSN", "Annual Salary", "Department", "Tenured");

if (facultyCounter < NUM_FACULTY)
{

   for ( int index = 0; index < facultyCounter; index++)
   {
      facultyArray[index].displayPersonData();
   }

   System.out.printf("\n\n%d of %d Faculty have been "
         + "entered.\n", facultyCounter, NUM_FACULTY);
}

else
{
   for ( int index = 0; index < facultyCounter; index++)
   {
      facultyArray[index].displayPersonData();
   }
}

System.out.println();

//display the students
System.out.println("-------------------------------------------"
      + "----------------------------------------------------"
      + "----------------------------------------------------");

System.out.println("Students\n");

System.out.printf("%-20s%-25s%-12s%-15s%-15s%-10s",
      "Name", "Street Address", "Zip Code", "Phone Number",
      "Major", "GPA");

if (studentCounter < NUM_STUDENTS)
{

   for ( int index = 0; index < studentCounter; index++)
   {
      studentArray[index].displayPersonData();
   }

   System.out.printf("\n\n%d of %d Students have been "
         + "entered.\n", studentCounter, NUM_STUDENTS);
}

else
```

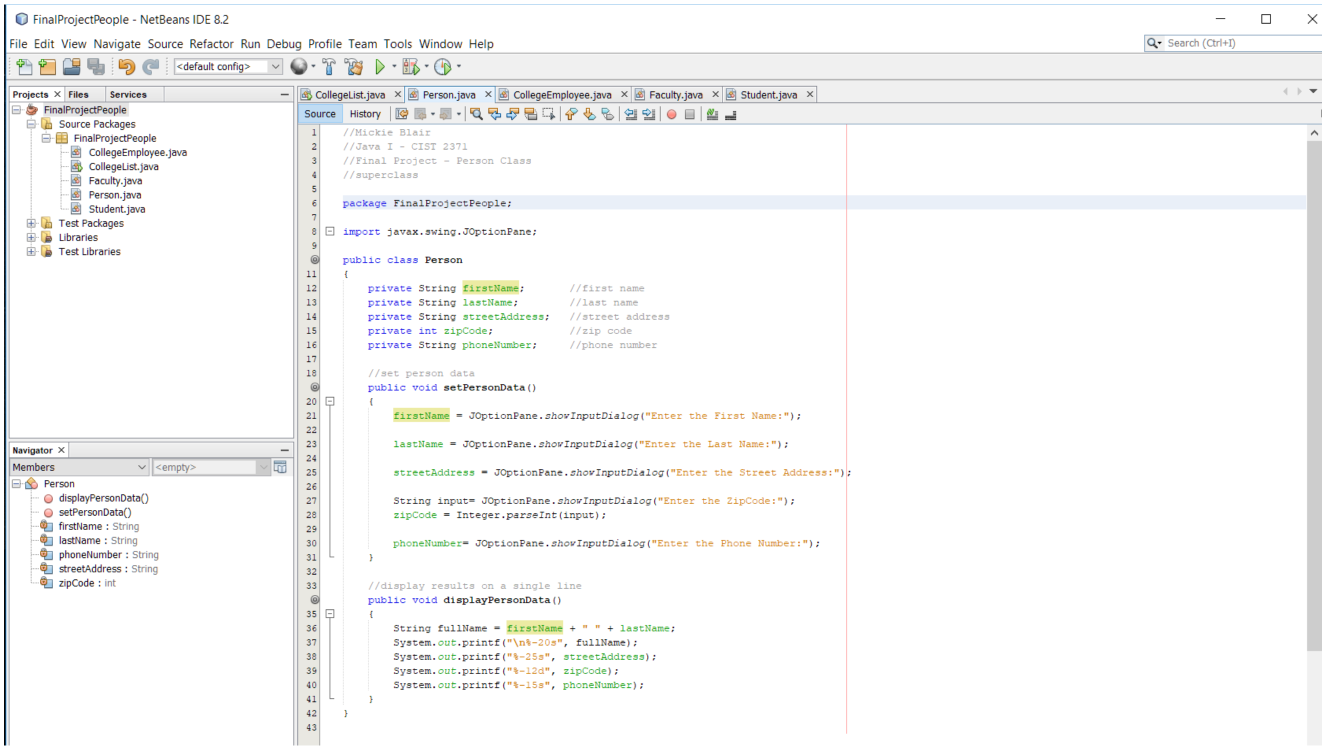
```
            {
               for ( int index = 0; index < facultyCounter; index++)
               {
                  studentArray[index].displayPersonData();
               }
            }

            System.out.println();
         }

      }

   }

}
```

OUTPUT

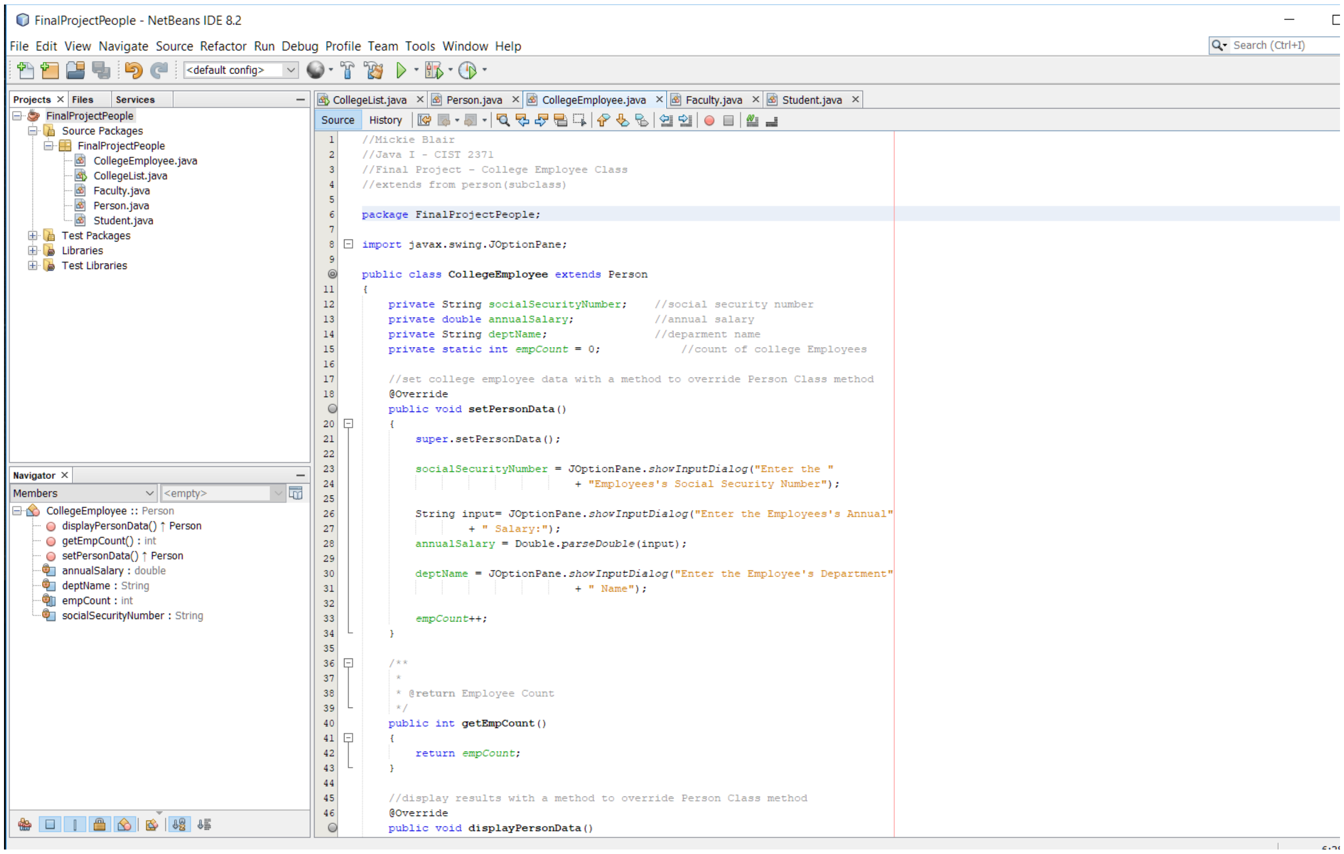
```
COLLEGE LIST REPORT
----------------------------------------------------------------------------------------------------------------------------------------------------
College Employees

Name              Street Address      Zip Code    Phone Number   SSN            Annual Salary    Department
Anna Smith        123 Main Street     30010       555-222-2222   012-34-4565    $ 50000.00       Admissions
Bob Williams      459 Water Street    30154       555-888-2222   500-00-1234    $ 40000.00       Maintenance

2 of 4 College Employees have been entered.

----------------------------------------------------------------------------------------------------------------------------------------------------
Faculty

Name              Street Address      Zip Code    Phone Number   SSN            Annual Salary    Department     Tenured
Jane Jones        498 River Road      35451       555-777-8458   019-99-0000    $ 30000.00       Biology        YES
Gary Green        654 Oak Road        35451       555-219-8500   999-01-1234    $ 35000.00       Mathematics    NO

2 of 3 Faculty have been entered.

----------------------------------------------------------------------------------------------------------------------------------------------------
Students

Name              Street Address      Zip Code    Phone Number   Major          GPA
Jim Henry         12-B Lake Street    32541       555-487-0125   Nursing        3.75
Sophia Timmons    54 Shadow Trace     35489       555-400-1254   Programming    3.91
Tim White         149 Willow Road     32598       555-854-0054   English        3.30

3 of 7 Students have been entered.
```



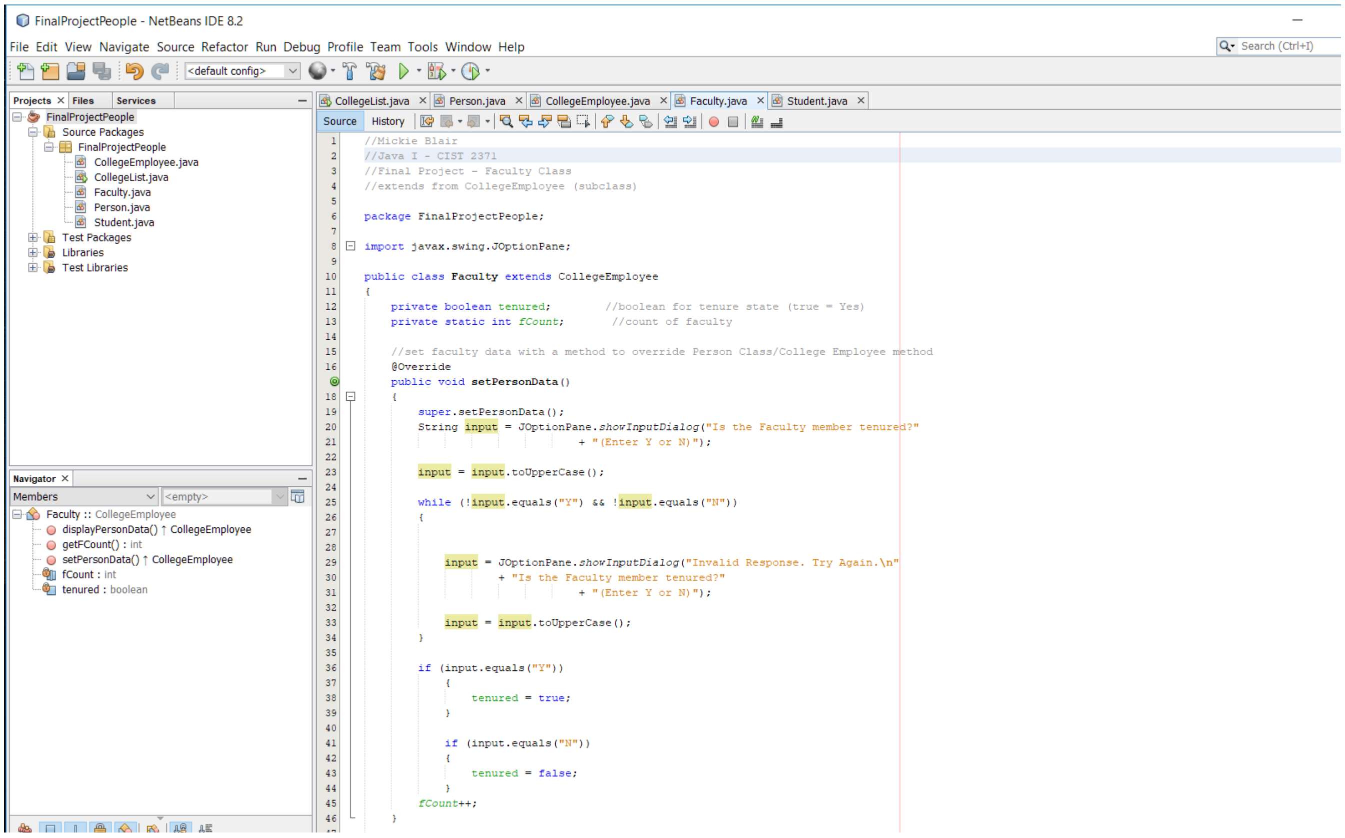
```
run:

COLLEGE LIST REPORT
------------------------------------------------------------------------------------------------------------------------------------------------------
College Employees

Name                Street Address           Zip Code    Phone Number   SSN            Annual Salary    Department
Anna Smith          123 Main Street          30010       555-222-2222   012-34-4565    $ 50000.00       Admissions
Bob Williams        459 Water Street         30154       555-888-2222   500-00-1234    $ 40000.00       Maintanence

2 of 4 College Employees have been entered.

------------------------------------------------------------------------------------------------------------------------------------------------------
Faculty

Name                Street Address           Zip Code    Phone Number   SSN            Annual Salary    Department     Tenured
Jane Jones          498 River Road           35451       555-777-8458   019-99-0000    $ 30000.00       Biology        YES
Gary Green          654 Oak Road             35451       555-219-8500   999-01-1234    $ 35000.00       Mathematics    NO

2 of 3 Faculty have been entered.

------------------------------------------------------------------------------------------------------------------------------------------------------
Students

Name                Street Address           Zip Code    Phone Number   Major          GPA
Jim Henry           12-B Lake Street         32541       555-487-0125   Nursing        3.75
Sophia Timmons      54 Shadow Trace          35489       555-400-1254   Programming    3.91
Tim White           149 Willow Road          32598       555-854-0054   English        3.30

3 of 7 Students have been entered.

BUILD SUCCESSFUL (total time: 7 minutes 24 seconds)
```



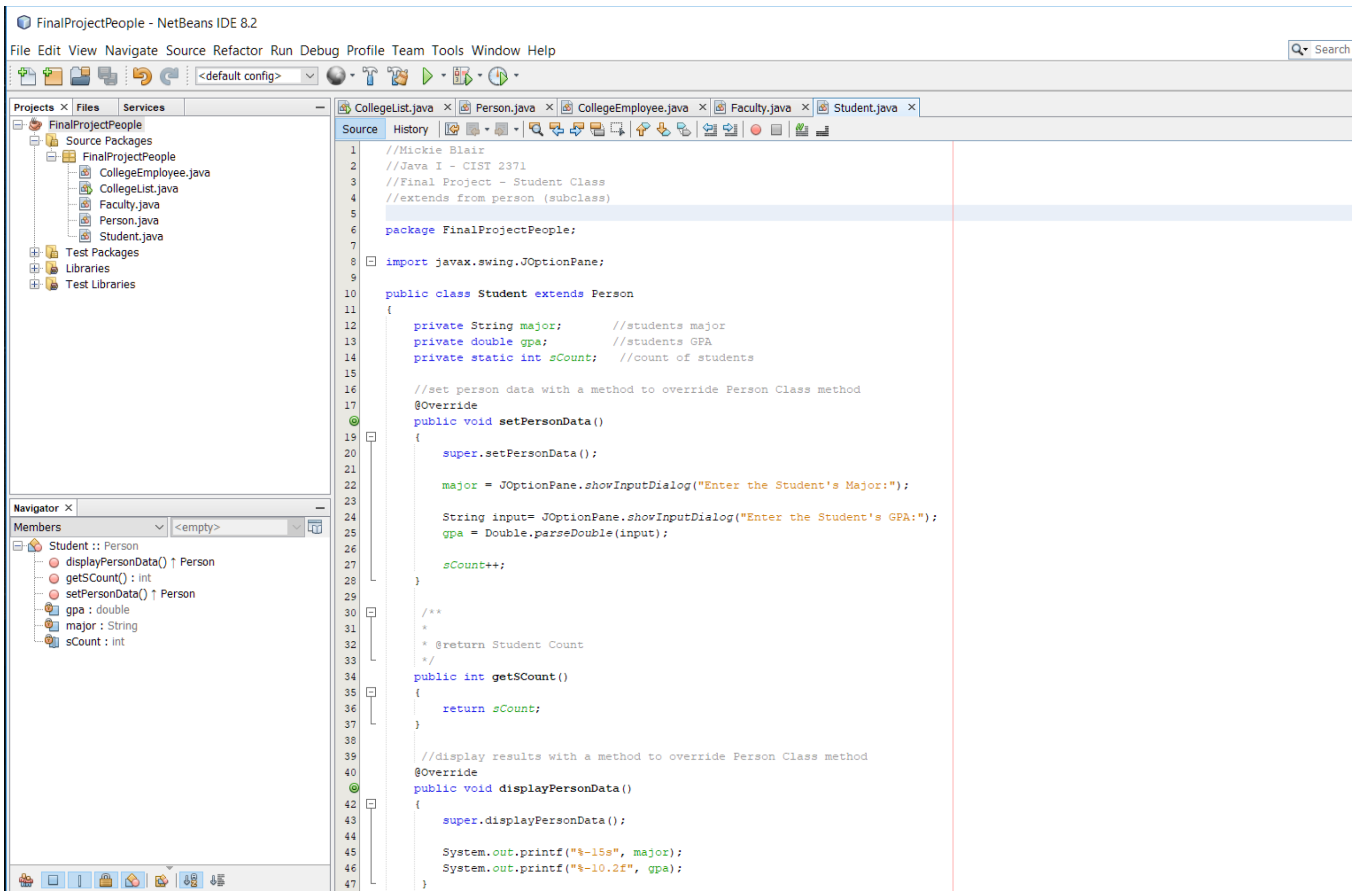
```java
//Mickie Blair
//Java I - CIST 2371
//Final Project - Person Class
//superclass

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class Person
{
    private String firstName;       //first name
    private String lastName;        //last name
    private String streetAddress;   //street address
    private int zipCode;            //zip code
    private String phoneNumber;     //phone number

    //set person data
    public void setPersonData()
    {
        firstName = JOptionPane.showInputDialog("Enter the First Name:");

        lastName = JOptionPane.showInputDialog("Enter the Last Name:");

        streetAddress = JOptionPane.showInputDialog("Enter the Street Address:");

        String input= JOptionPane.showInputDialog("Enter the ZipCode:");
        zipCode = Integer.parseInt(input);

        phoneNumber= JOptionPane.showInputDialog("Enter the Phone Number:");
    }

    //display results on a single line
    public void displayPersonData()
    {
        String fullName = firstName + " " + lastName;
        System.out.printf("\n%-20s", fullName);
        System.out.printf("%-25s", streetAddress);
        System.out.printf("%-12d", zipCode);
        System.out.printf("%-15s", phoneNumber);
    }
}
```



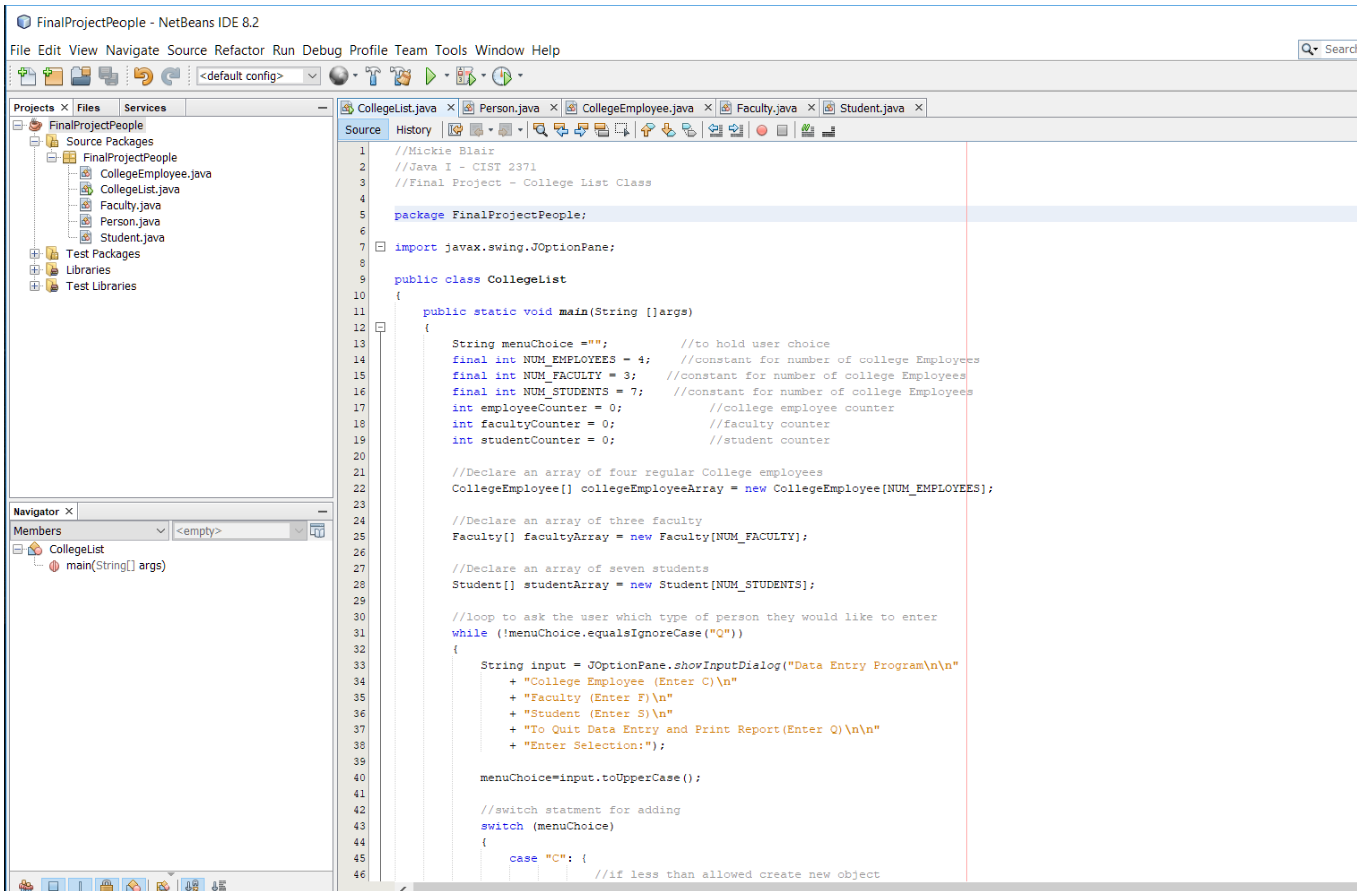
```java
//Mickie Blair
//Java I - CIST 2371
//Final Project - College Employee Class
//extends from person(subclass)

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class CollegeEmployee extends Person
{
    private String socialSecurityNumber;    //social security number
    private double annualSalary;            //annual salary
    private String deptName;                //deparment name
    private static int empCount = 0;            //count of college Employees

    //set college employee data with a method to override Person Class method
    @Override
    public void setPersonData()
    {
        super.setPersonData();

        socialSecurityNumber = JOptionPane.showInputDialog("Enter the "
                                + "Employees's Social Security Number");

        String input= JOptionPane.showInputDialog("Enter the Employees's Annual"
                + " Salary:");
        annualSalary = Double.parseDouble(input);

        deptName = JOptionPane.showInputDialog("Enter the Employee's Department"
                                + " Name");

        empCount++;
    }

    /**
     *
     * @return Employee Count
     */
    public int getEmpCount()
    {
        return empCount;
    }

    //display results with a method to override Person Class method
    @Override
    public void displayPersonData()
```



```java
//Mickie Blair
//Java I - CIST 2371
//Final Project - Faculty Class
//extends from CollegeEmployee (subclass)

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class Faculty extends CollegeEmployee
{
    private boolean tenured;        //boolean for tenure state (true = Yes)
    private static int fCount;       //count of faculty

    //set faculty data with a method to override Person Class/College Employee method
    @Override
    public void setPersonData()
    {
        super.setPersonData();
        String input = JOptionPane.showInputDialog("Is the Faculty member tenured?"
                                + "(Enter Y or N)");

        input = input.toUpperCase();

        while (!input.equals("Y") && !input.equals("N"))
        {


            input = JOptionPane.showInputDialog("Invalid Response. Try Again.\n"
                    + "Is the Faculty member tenured?"
                                + "(Enter Y or N)");

            input = input.toUpperCase();
        }

        if (input.equals("Y"))
            {
                tenured = true;
            }

            if (input.equals("N"))
            {
                tenured = false;
            }
        fCount++;
    }
```



```java
//Mickie Blair
//Java I - CIST 2371
//Final Project - Student Class
//extends from person (subclass)

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class Student extends Person
{
    private String major;       //students major
    private double gpa;         //students GPA
    private static int sCount;   //count of students

    //set person data with a method to override Person Class method
    @Override
    public void setPersonData()
    {
        super.setPersonData();

        major = JOptionPane.showInputDialog("Enter the Student's Major:");

        String input= JOptionPane.showInputDialog("Enter the Student's GPA:");
        gpa = Double.parseDouble(input);

        sCount++;
    }

     /**
     *
     * @return Student Count
     */
    public int getSCount()
    {
        return sCount;
    }

     //display results with a method to override Person Class method
    @Override
    public void displayPersonData()
    {
        super.displayPersonData();

        System.out.printf("%-15s", major);
        System.out.printf("%-10.2f", gpa);
     }
```



```java
//Mickie Blair
//Java I - CIST 2371
//Final Project - College List Class

package FinalProjectPeople;

import javax.swing.JOptionPane;

public class CollegeList
{
    public static void main(String []args)
    {
        String menuChoice ="";          //to hold user choice
        final int NUM_EMPLOYEES = 4;    //constant for number of college Employees
        final int NUM_FACULTY = 3;    //constant for number of college Employees
        final int NUM_STUDENTS = 7;    //constant for number of college Employees
        int employeeCounter = 0;            //college employee counter
        int facultyCounter = 0;             //faculty counter
        int studentCounter = 0;             //student counter

        //Declare an array of four regular College employees
        CollegeEmployee[] collegeEmployeeArray = new CollegeEmployee[NUM_EMPLOYEES];

        //Declare an array of three faculty
        Faculty[] facultyArray = new Faculty[NUM_FACULTY];

        //Declare an array of seven students
        Student[] studentArray = new Student[NUM_STUDENTS];

        //loop to ask the user which type of person they would like to enter
        while (!menuChoice.equalsIgnoreCase("Q"))
        {
            String input = JOptionPane.showInputDialog("Data Entry Program\n\n"
                + "College Employee (Enter C)\n"
                + "Faculty (Enter F)\n"
                + "Student (Enter S)\n"
                + "To Quit Data Entry and Print Report(Enter Q)\n\n"
                + "Enter Selection:");

            menuChoice=input.toUpperCase();

            //switch statment for adding
            switch (menuChoice)
            {
                case "C": {
                            //if less than allowed create new object
```