```
//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

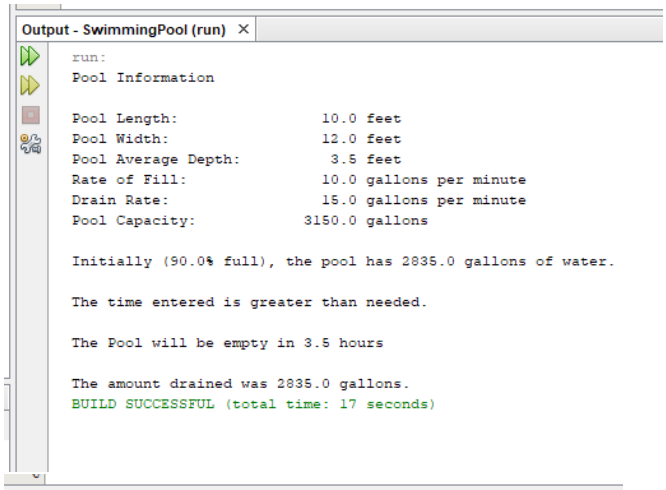
Pool Length:	10.0 feet
Pool Width:	12.0 feet
Pool Average Depth:	3.5 feet
Rate of Fill:	10.0 gallons per minute
Drain Rate:	15.0 gallons per minute
Pool Capacity:	3150.0 gallons

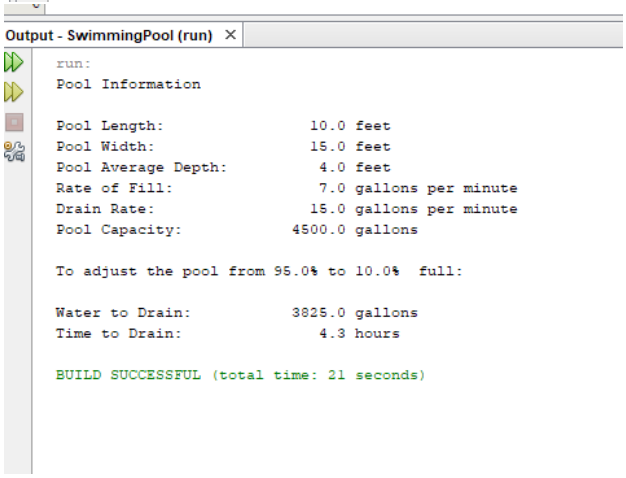
Initially (90.0% full), the pool has 2835.0 gallons of water.

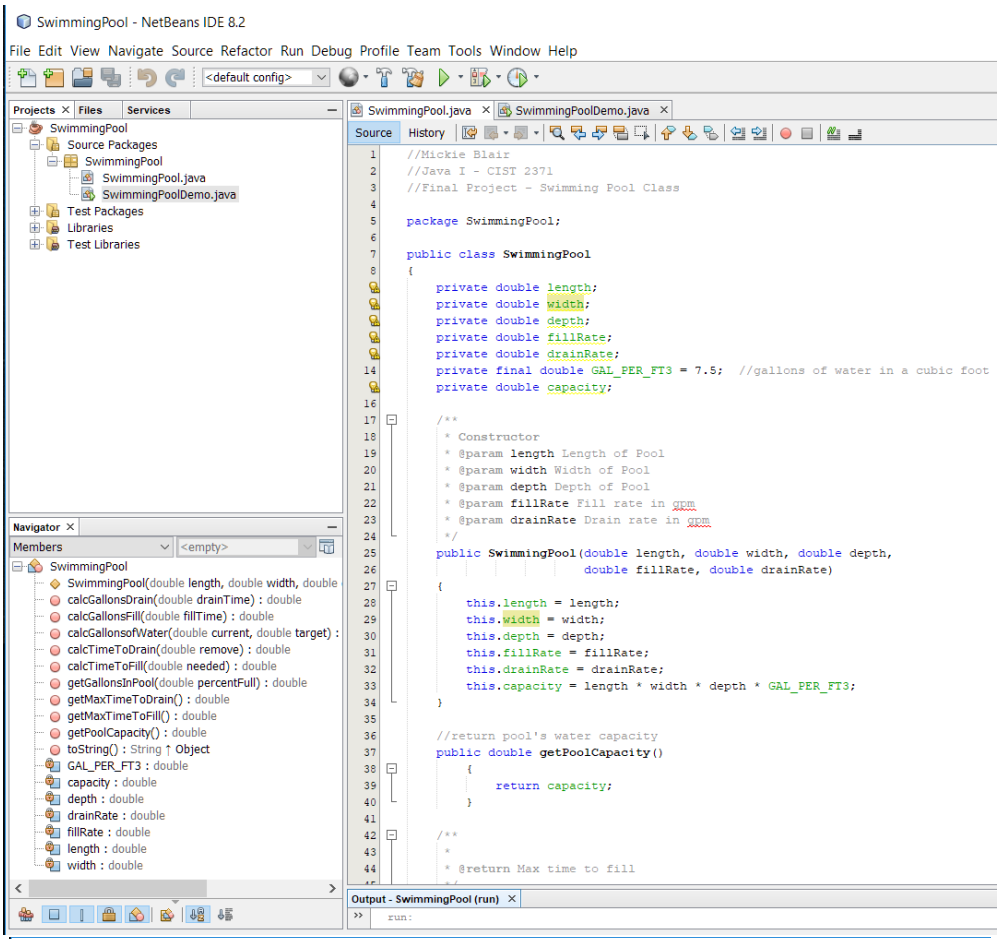
The time entered is greater than needed.

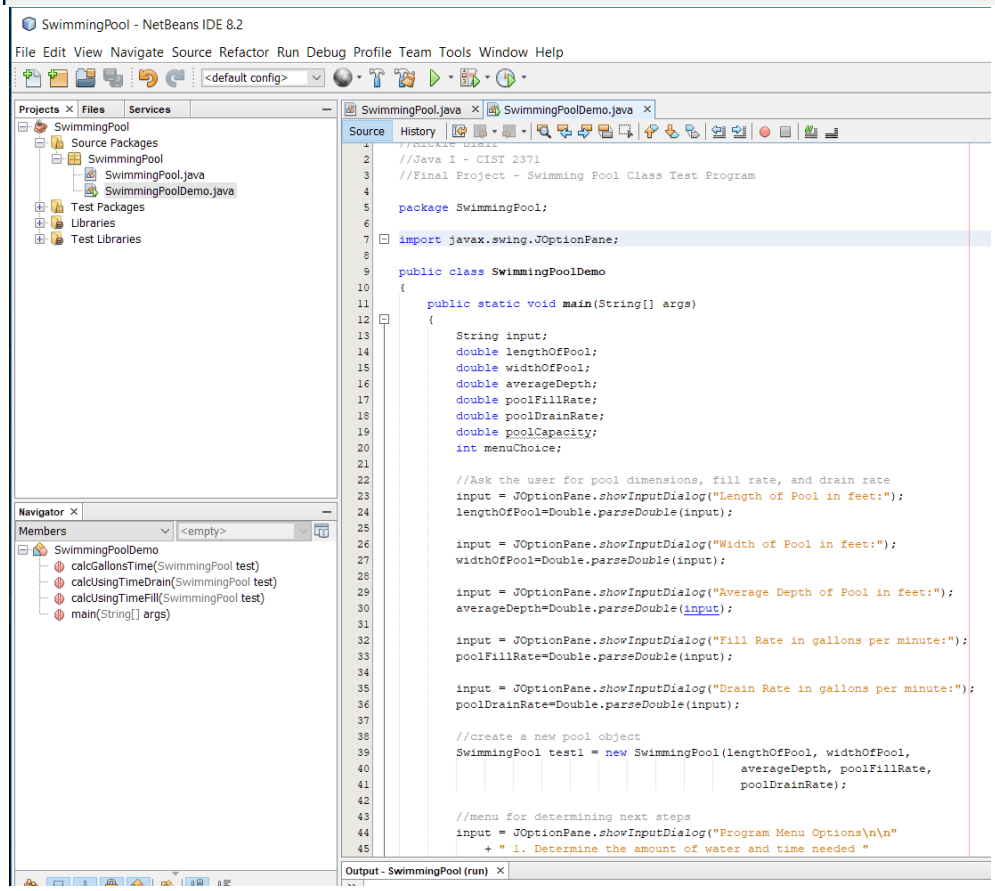
The Pool will be empty in 3.5 hours

The amount drained was 2835.0 gallons.






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
1 //Mickie Blair
2 //Java I - CIST 2371
3 //Final Project - Swimming Pool Class
4
5 package SwimmingPool;
6
7 public class SwimmingPool
8 {
9 private double length;
10 private double width;
11 private double depth;
12 private double fillRate;
13 private double drainRate;
14 private final double GAL_PER_FT3 = 7.5; //gallons of water in a cubic foot
15 private double capacity;
16
17 /**
18 * Constructor
19 * @param length Length of Pool
20 * @param width Width of Pool
21 * @param depth Depth of Pool
22 * @param fillRate Fill rate in gpm
23 * @param drainRate Drain rate in gpm
24 */
25 public SwimmingPool(double length, double width, double depth,
26 double fillRate, double drainRate)
27 {
28 this.length = length;
29 this.width = width;
30 this.depth = depth;
31 this.fillRate = fillRate;
32 this.drainRate = drainRate;
33 this.capacity = length * width * depth * GAL_PER_FT3;
34 }
35
36 //return pool's water capacity
37 public double getPoolCapacity()
38 {
39 return capacity;
40 }
41
42 /**
43 *
44 * @return Max time to fill
Output - SwimmingPool (run)
run:


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>
SwimmingPoolDemo
calcGallonsTime(SwimmingPool test)
calcUsingTimeDrain(SwimmingPool test)
calcUsingTimeFill(SwimmingPool test)
main(String[] args)
SwimmingPool.java
SwimmingPoolDemo.java
Source History
2 //Java I - CIST 2371
3 //Final Project - Swimming Pool Class Test Program
4
5 package SwimmingPool;
6
7 import javax.swing.JOptionPane;
8
9 public class SwimmingPoolDemo
10 {
11 public static void main(String[] args)
12 {
13 String input;
14 double lengthOfPool;
15 double widthOfPool;
16 double averageDepth;
17 double poolFillRate;
18 double poolDrainRate;
19 double poolCapacity;
20 int menuChoice;
21
22 //Ask the user for pool dimensions, fill rate, and drain rate
23 input = JOptionPane.showInputDialog("Length of Pool in feet:");
24 lengthOfPool=Double.parseDouble(input);
25
26 input = JOptionPane.showInputDialog("Width of Pool in feet:");
27 widthOfPool=Double.parseDouble(input);
28
29 input = JOptionPane.showInputDialog("Average Depth of Pool in feet:");
30 averageDepth=Double.parseDouble(input);
31
32 input = JOptionPane.showInputDialog("Fill Rate in gallons per minute:");
33 poolFillRate=Double.parseDouble(input);
34
35 input = JOptionPane.showInputDialog("Drain Rate in gallons per minute:");
36 poolDrainRate=Double.parseDouble(input);
37
38 //create a new pool object
39 SwimmingPool test1 = new SwimmingPool(lengthOfPool, widthOfPool,
40 averageDepth, poolFillRate,
41 poolDrainRate);
42
43 //menu for determining next steps
44 input = JOptionPane.showInputDialog("Program Menu Options\n\n"
45 + " 1. Determine the amount of water and time needed "
Output - SwimmingPool (run)