```
import java.util.*;
import java.math.*;
public class NoteTester
{
    public static void main(String[] args)
    {
      //Declare variables to hold user input
      int noteValue;
      String noteLength;
      int tempLength;

      //Create scanner object to hold user input
      Scanner userInput = new Scanner(System.in);

      //Retrieve starting value from user
      System.out.println("Please enter a value for your note\n"+
                         "between -48 and 39:");
      noteValue = userInput.nextInt();

      //While loop ensures a valid number
      while(noteValue < -48 || noteValue > 39)
      {
         System.out.println("Please enter a valid value for your
note\n"+
                            "between -48 and 39:");
         noteValue = userInput.nextInt();
      }


      //Retrieve starting length from user
      System.out.println("Please enter the number between 1-5\n"
+
                         "you would like to represent\n" +
                         "the length of your note.\n"+
                         "1. Whole\n" +
                         "2. Half\n" +
                         "3. Quarter\n" +
                         "4. Eighth\n" +
                         "5. Sixteenth");
      tempLength = userInput.nextInt();

      //While loop ensures a valid length
      while(tempLength > 5 || tempLength < 1)
      {
```

```
            System.out.println("Please enter the number between 1-
5\n" +
                               "you would like to represent\n" +
                               "the length of your note.\n"+
                               "1. Whole\n" +
                               "2. Half\n" +
                               "3. Quarter\n" +
                               "4. Eighth\n" +
                               "5. Sixteenth");
            tempLength = userInput.nextInt();
         }
         //Create demoNote object
         Note demoNote = new Note(noteValue, tempLength);


         //Print user the information about their note
         System.out.print("Your value was " + demoNote.getValue() +
", your length was a(n) " + demoNote.getLengthString()
                          + " note, your note's \nletter value was
" + demoNote.getLetter() + " which is a " +
                          demoNote.getSharp() + " note. \nThe
frequency of your" +
                           " note is " + demoNote.getFreq() + "
hz." );
         //This portion of the program creates a list of Note
objects
         //and sorts them
         Note other = new Note(-35, 1);
         Note other1 = new Note(4, 1);
         Note other2 = new Note(16, 4);
         Note other3 = new Note(-34, 5);
         Note[] notes = {demoNote,other,other1,other2,other3};
         Arrays.sort(notes);
         System.out.println("\nHere is a list of notes sorted by
length, then by frequency: ");
         for(int i = 0; i < notes.length; i++)
         {
            System.out.println(notes[i].toString());
         }


   }
}
```

```
class Note implements Comparable<Note>
{
   /*Declare variables privately so they cannot be accessed from
outside the class. Set the value default
   to middle C and the length default to quarter.*/
   private int value = 0;
   private int length = 3;
   private String letter;
   private String sharp;
   private double freq;

   //Comparable method
   public int compareTo(Note other)
   {
      if(getLength() > other.getLength())
      {
         return 1;
      }
      else if(getLength() < other.getLength())
      {
         return -1;
      }
      else if(getFreq() < other.getFreq())
      {
        return 1;
      }
      else if(getFreq() > other.getFreq())
      {
        return -1;
      }
      else
      {
         return 0;
      }
   }
   /**
    * Constructor
    * @param value the note's starting value
    * @param length the note's starting length
    */
   public Note(int value, int length)
   {
      this.value = value;
```

```
        this.length = length;
    }

    //Setters

    /**
     * Assigns this.value to variable value
     */
    void setValue(int value)
    {
        this.value = value;
    }

    /**
     * Assigns this.length to variable length
     */
    void setLength(int x)
    {
        this.length = length;
    }

   //Getters

    /**
     * @return The value of the note
     */
    public double getValue()
    {
        return value;
    }

    /**
     * @return The length of the note
     */
    public int getLength()
    {
        return length;
    }

    /**
     * @return The letter of the chromatic scale the value is
assigned to
     */
    public String getLetter()
```

```
{
   switch(value % 12)
   {
      case 0 :
         letter = "A";
         break;
      case 1 :
         letter = "A#";
         break;
      case 2 :
         letter = "B";
         break;
      case 3 :
         letter = "C";
         break;
      case 4 :
         letter = "C#";
         break;
      case 5 :
         letter = "D";
         break;
      case 6 :
         letter = "D#";
         break;
      case 7 :
        letter = "E";
         break;
      case 8 :
         letter = "F";
         break;
      case 9 :
         letter = "F#";
         break;
      case 10 :
         letter = "G";
         break;
      case 11 :
         letter = "G#";
         break;
      case -1 :
         letter = "A#";
         break;
      case -2 :
         letter = "B";
```

```
            break;
         case -3 :
            letter = "C";
            break;
         case -4 :
            letter = "C#";
            break;
         case -5 :
            letter = "D";
            break;
         case -6 :
            letter = "D#";
            break;
         case -7 :
            letter = "E";
            break;
         case -8 :
            letter = "F";
            break;
         case -9 :
            letter = "F#";
            break;
         case -10 :
            letter = "G";
            break;
         case -11 :
            letter = "G#";
            break;
         default :
            letter = "Invalid";

      }
      return letter;
   }

   /**
    * @return Whether or not the note is sharp
    */
   public String getSharp()
   {
      switch(value % 12)
      {
         case 0 :
            sharp = "natural";
```

```
      break;
    case 1 :
      sharp = "sharp";
      break;
    case 2 :
      sharp = "natural";
      break;
    case 3 :
      sharp = "natural";
      break;
    case 4 :
      sharp = "sharp";
      break;
    case 5 :
      sharp = "natural";
      break;
    case 6 :
      sharp = "sharp";
      break;
    case 7 :
      sharp = "natural";
      break;
    case 8 :
      sharp = "natural";
      break;
    case 9 :
      sharp = "sharp";
      break;
    case 10 :
      sharp = "natural";
      break;
    case 11 :
      sharp = "sharp";
      break;
    case -1 :
      sharp = "sharp";
      break;
    case -2 :
      sharp = "natural";
      break;
   case -3 :
      sharp = "natural";
      break;
    case -4 :
```

```
            sharp = "sharp";
          break;
        case -5 :
            sharp = "natural";
            break;
        case -6 :
            sharp = "sharp";
            break;
        case -7 :
            sharp = "natural";
            break;
        case -8 :
            sharp = "natural";
            break;
        case -9 :
          sharp = "sharp";
            break;
        case -10 :
            sharp = "natural";
            break;
        case -11 :
            sharp = "sharp";
            break;

        default :
            letter = "Invalid";
     }
    return sharp;
}

/**
  * @return The frequency of the note
  */
 public double getFreq()
 {
    freq = (440 * Math.pow(2, value / 12.0));
    return freq;
 }
 public String getLengthString()
 {
    if(getLength() == 1)
    {
       return "whole";
    }
```

```
      else if(getLength() == 2)
      {
         return "half";
      }
      else if(getLength() == 3)
      {
         return "quarter";
      }
      else if(getLength() == 4)
      {
         return "eighth";
      }
      else if(getLength() == 5)
      {
         return "sixteenth";
      }
      else
      {
         return "Invalid";
      }
   }

   //ToString to print notes
   public String toString()
   {
      return "(Length :: " + getLengthString() + " Frequency ::
" + getFreq() + ")";
   }
}
/*=====TESTING==========

Correct Notes

1.
Input
Value = 0
Length = 1

Output

Letter = A
SharporNat = Natural
Freq = 440 hz
Length = Whole
```

2.
Input
Value = 15
Length = 3

Output

Letter = C
SharporNat = Natural
Freq = 1046.50
Length = Quarter

3.
Input
Value = -25
Length = 5

Output
Letter = G#
SharporNat = Sharp
Freq = 103.826
Length = Sixteenth

Test Notes (What came out of my program)

1.
Letter = A
SharporNat = Natural
Freq = 440
Length = Whole

2.
Letter = C
SharporNat = Natural= 1046.50
Length = Quarter

3.
Letter = G#
SharporNat = Sharp
Freq = 103.826
Length = Sixteenth

All test results were the same as the correct results done by an

```
external calculator.
==========================*/

import java.util.*;
/*
*
* @author Alex Durso
*
*/

public class QuadraticTester
{
   public static void main(String[] args)
   {
      //Create new scanner to retrieve starting values from user
      Scanner userInput = new Scanner(System.in);

      //Allow user to input values for a, b and, c
      System.out.println("Welcome to the quadratic equation
solver. Your \n" +
                         "equation will be modeled in the form
of ax^2 + bx" +
                         " + c.");
      System.out.println("Please enter a value for a: ");
      double a = userInput.nextDouble();
      System.out.println("Please enter a value for b: ");
      double b = userInput.nextDouble();
      System.out.println("Please enter a value for c: ");
      double c = userInput.nextDouble();
      Quadratic quad = new Quadratic(a, b, c);

      //If statement to determine if values form a valid
quadratic
      //equation
      if(a == 0)
      {
         System.out.println("Your equation is not quadratic");
         System.exit(0);
      }

      //Else statement to determine if equation has real or
      //complex roots
      else if(quad.realRoots() == false)
      {
```

```
         System.out.println("Your equation had a negative
discriminant.\n" +
                            "Therefore your equation did not have
any real roots. \n" +
                             "Your equation must have complex
roots" );
      }

      //If quadratic is valid and has real roots tell user their
      //roots
      else
      {
         double root1 = quad.firstRoot();
         double root2 = quad.secondRoot();
         System.out.println("Your equation was " + a + "^2 + " +
b + "x + " + c +
                             ", \nyour first root was " + root1 +
" and your second\n" +
                            "root was " + root2);
      }

      //Ask user if they would like to calculate a discriminant
      System.out.println("Would you like to compute the value\n"
+
                         "of the first derivative of the
quadratic\n" +
                         "at a specific point?" +
                         "\nPlease enter 1 for yes and 2 for
no");
      double input = userInput.nextDouble();

      //If statement for if user would like to calculate
      //derivaive
      if(input == 1)
      {
         //Gather user value for x
         System.out.println("Please enter a value for x: ");
         double x = userInput.nextDouble();
         //Set X to x
         quad.setValue(x);
         //Set deriv to calculated derivative
         double deriv = quad.derivative();
         System.out.println("The derivative of " + a + "x " + "
+ b"
```

```
                              + "x + " + c + "\nat point " + x + "
was : "
                              + deriv + ".");
      }
      //Statement for if user does not want to calculate
derivative
      else
      {
         System.out.println("Goodbye");
      }
      //This portion of the program creates a list of
quadratics, then sorts
      //them by how fast they open
      Quadratic other = new Quadratic(1,2,3);
      Quadratic other1 = new Quadratic(5,6,7);
      Quadratic other2 = new Quadratic(3,4,6);
      Quadratic other3 = new Quadratic(1,8,12);
      Quadratic[] quadratics =
{quad,other,other1,other2,other3};
      Arrays.sort(quadratics);
      System.out.println("\nHere is a list of quadratics sorted
by how quickly they open: ");
      for(int i = 0; i < quadratics.length; i++)
      {
         System.out.println(quadratics[i].toString());
      }


   }
}
class Quadratic implements Comparable<Quadratic>
{
   //Declare variables
   private double A;
   private double B;
   private double C;
   private double X;
   private double discriminant;

   //Comparable method
   public int compareTo(Quadratic other)
   {
      if(getA() < other.getA())
      {
```

```
      return 1;
   }
   if(getA() > other.getA())
   {
      return -1;
   }
   else
   {
      return 0;
   }
}

/**
 * Constructor
 * @param A the value of a in the quadratic equation
 * @param B the value of b in the quadratic equation
 * @param C the value of c in the quadratic equation
 */
public Quadratic(double a, double b, double c)
{
   A = a;
   B = b;
   C = c;
   calcDiscriminant();
}

//Setters

/**
 * Assigns x to variable X
*/
void setValue(double x)
{
   this.X = x;
}

void setA(double a)
{
  A = a;
}
void setB(double b)
{
  B = b;
}
```

```
void setC(double c)
{
  C = c;
}


/**
 * Calculates discriminant and sets it equal to
 * variable discriminant
 */
private void calcDiscriminant()
{
   discriminant = B*B - 4.0*A*C;
}

//Getters

/**
 * @return The first root of the quadratic
 */

public double getA()
{
   return A;
}
 public double getB()
{
   return B;
}
public double getC()
{
   return C;
}
public double firstRoot()
{
   return (-B + Math.sqrt(discriminant))/(2.0*A);
}

/**
 * @return The second root of the quadratic
 */
public double secondRoot()
{
   return (-B - Math.sqrt(discriminant))/(2.0*A);
```

```
    }

    /**
     * @return Whether or not the quadratic has real roots
     */
    public boolean realRoots()
    {
       if(discriminant < 0) return false;
       else return true;
    }

    /**
     * @return The calculation of the derivative
     */
    public double derivative()
    {
       return ((2.0*A*X) + B);
    }
    //ToString to print quadratics
    public String toString()
    {
       return "(Quadratic:: " + getA() + "x^2 + " + getB() + "x +
" + getC() + ")";
    }
}

/*=====TESTING==========

Correct Equations (Done with calculator)

2x^1 + 4x + 2
Roots = (-1, -1)
Discriminant = Positive
Derivative at point x = 2 : f'(2) = 12

0x^1 + 4x + 2

Not quadratic - N/A

1x^1 + 2x + 2
Roots = Complex
Discriminant = Negative
```

```
Derivative at point x = 6 : f'(6) = 14
x^2 - 2x + 1
Roots = (1, 1)
Discriminant = Zero
Derivative at point x = 6 : f'(6) = 10

15x^2 - 36x + 16
Roots = (1.81, .58)
Discriminant = Positive
Derivative at point x = 15 f'(15) = 414

Output from my code:

2x^2 + 4x + 2
Roots = (-1, -1)
Discriminant = Positive
Derivative at point x = 2 : f'(2) = 12

0x^2 + 4x + 2

Not quadratic - N/A

1x^2 + 2x + 2
Roots = Complex
Discriminant = Negative
Derivative at point x = 6 : f'(6) = 14

x^2 - 2x + 1
Roots = (1, 1)
Discriminant = Zero
Derivative at point x = 6 : f'(6) = 10

15x^2 - 36x + 16
Roots = (1.81, .58)
Discriminant = Positive
Derivative at point x = 15 f'(15) = 414

*/
public interface TV
{
```

```
    public void changeChan();
    public void changeVol();
}
//TV should be an interface because TVs have different hardware.
//Some have speakers or the ability to connect to the internet.
//For TVs like this they should be able to inherit the speaker
//class as well as the internetCapability and TV interface

public interface Polygons
{
    public double findAreaofPolygons()
    {
    }
}

//Polygons should be an interface because it doesn't need a
//method body and is easily changeable by user.
```