

Name: Dan Cassidy

Class: CSCI-C 490, Mobile Application Development

Assignment: Homework 3 Part 1

Date: 2015-07-10

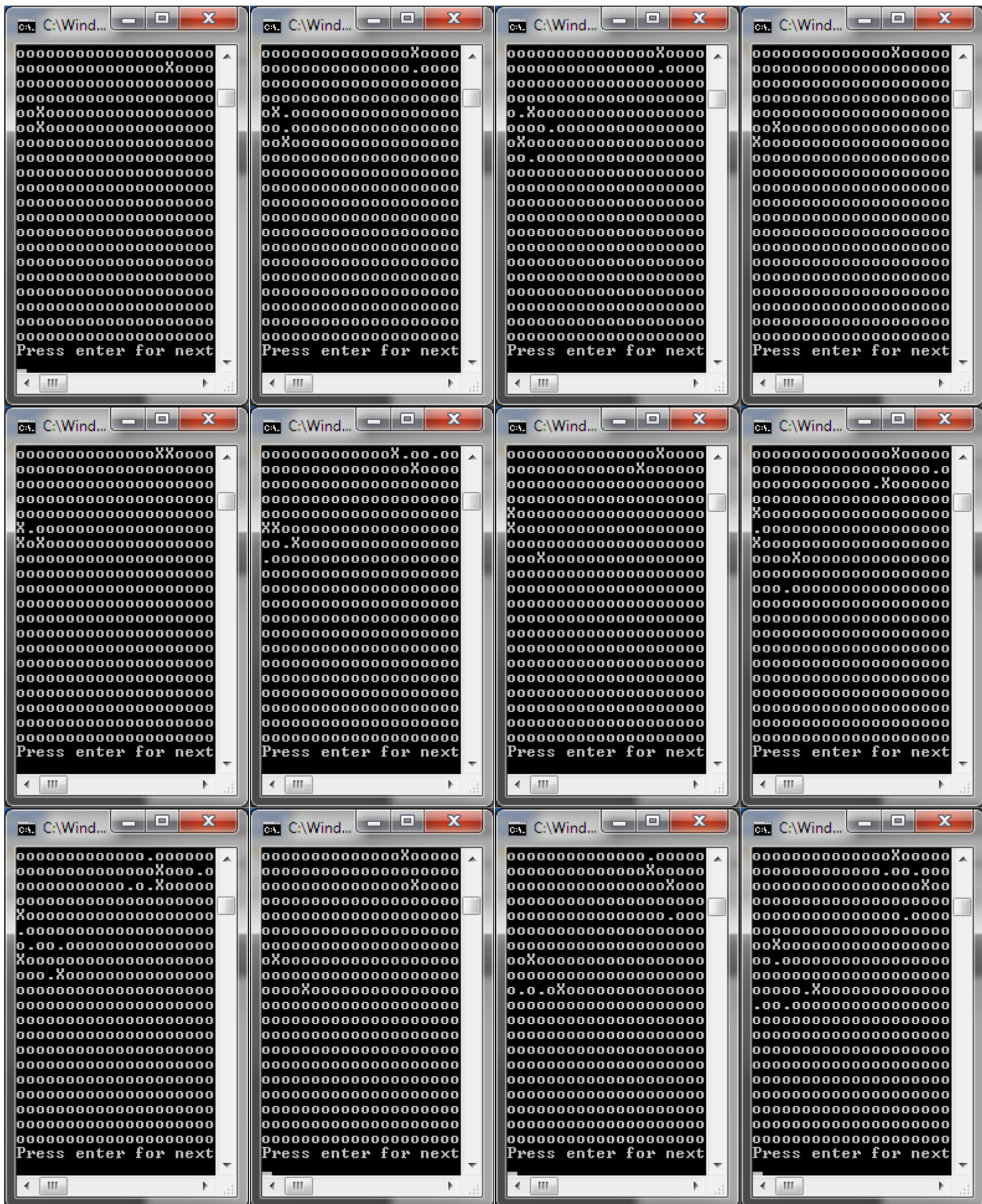
```
1  /**
2   * Now define Anteater Class
3   */
4  class Anteater extends Organism
5  {
6      public static final int ANTEATER_BREED = 8;
7      public static final int ANTEATER_STARVE = 3;
8
9      // Number of ticks since eating.
10     private int starveTicks = 0;
11
12     /**
13      * Default constructor.
14      */
15     public Anteater()
16     {
17         super();
18     }
19
20     /**
21      * 3-parameter constructor.
22      *
23      * @param world The World object this Anteater lives in.
24      * @param x The x coordinate of the spot in <b>world</b> this Anteater inhabits.
25      * @param y The y coordinate of the spot in <b>world</b> this Anteater inhabits.
26      */
27     public Anteater(World world, int x, int y)
28     {
29         super(world, x, y);
30     }
31
32     // Basic code reused from Ant.breed() method with changes to create an Anteater object instead
33     // of an Ant object.
34
35     /**
36      * Adjusts the breed counter for the Anteater object and creates a new one when the appropriate
37      * condition is met.
38      */
39     public void breed()
40     {
41         breedTicks++;
42         if (breedTicks == ANTEATER_BREED)
43         {
44             breedTicks = 0;
45             // Try to create a new Anteater object. Because world reference is passed in and
46             // Anteater object adds itself to that world, Anteater reference doesn't need to be
47             // explicitly saved.
48             if ((y > 0) && (world.getAt(x, y - 1) == null))
49             {
50                 new Anteater(world, x, y - 1);
51             }
52             else if ((y < World.WORLDSIZE - 1) && (world.getAt(x, y + 1) == null))
53             {
54                 new Anteater(world, x, y + 1);
55             }
56             else if ((x > 0) && (world.getAt(x - 1, y) == null))
57             {
58                 new Anteater(world, x - 1, y);
59             }
60             else if ((x < World.WORLDSIZE - 1) && (world.getAt(x + 1, y) == null))
```

```
61         {
62             new Anteater(world, x + 1, y);
63         }
64     }
65 }
66
67 // Basic code reused from Ant.move() method, with optimizations for move checking and additions
68 // to handle starvation.
69
70 /**
71  * Moves an Anteater object around and handles the starvation counter.
72  */
73 public void move()
74 {
75     starveTicks++;
76     int direction = (int) (Math.random() * 4);
77
78     // up
79     if (direction == 0)
80     {
81         if ((y > 0) && !(world.getAt(x, y - 1) instanceof Anteater))
82         {
83             // Reset starvation counter if Anteater "ate".
84             if (world.getAt(x, y - 1) instanceof Ant)
85                 starveTicks = 0;
86
87             // Move to new spot.
88             world.setAt(x, y - 1, world.getAt(x, y));
89             world.setAt(x, y, null);
90             y--;
91         }
92     }
93     // down
94     else if (direction == 1)
95     {
96         if ((y < World.WORLDSIZE - 1) && !(world.getAt(x, y + 1) instanceof Anteater))
97         {
98             // Reset starvation counter if Anteater "ate".
99             if (world.getAt(x, y + 1) instanceof Ant)
100                 starveTicks = 0;
101
102             // Move to new spot.
103             world.setAt(x, y + 1, world.getAt(x, y));
104             world.setAt(x, y, null);
105             y++;
106         }
107     }
108     // left
109     else if (direction == 2)
110     {
111         if ((x > 0) && !(world.getAt(x - 1, y) instanceof Anteater))
112         {
113             // Reset starvation counter if Anteater "ate".
114             if (world.getAt(x - 1, y) instanceof Ant)
115                 starveTicks = 0;
116
117             // Move to new spot.
118             world.setAt(x - 1, y, world.getAt(x, y));
119             world.setAt(x, y, null);
120             x--;
```

```
121     }
122   }
123   // right
124   else
125   {
126     if ((x < World.WORLDSIZE - 1) && !(world.getAt(x + 1, y) instanceof Anteater))
127     {
128       // Reset starvation counter if Anteater "ate".
129       if (world.getAt(x + 1, y) instanceof Ant)
130         starveTicks = 0;
131
132       // Move to new spot.
133       world.setAt(x + 1, y, world.getAt(x, y));
134       world.setAt(x, y, null);
135       x++;
136     }
137   }
138 }
139
140 /**
141  * Checks to see if the anteater is starving.
142  *
143  * @return boolean, indicating whether the anteater is starving (true) or not (false).
144  */
145 public boolean starve()
146 {
147     return (starveTicks == ANTEATER_STARVE ? true : false);
148 }
149
150 /**
151  * Returns "X" as the printable character for an Anteater object.
152  */
153 public String getPrintableChar()
154 {
155     return "X";
156 }
157 } // Anteater
158
```

Read from left to right, top to bottom.











Name: Dan Cassidy

Class: CSCI-C 490, Mobile Application Development

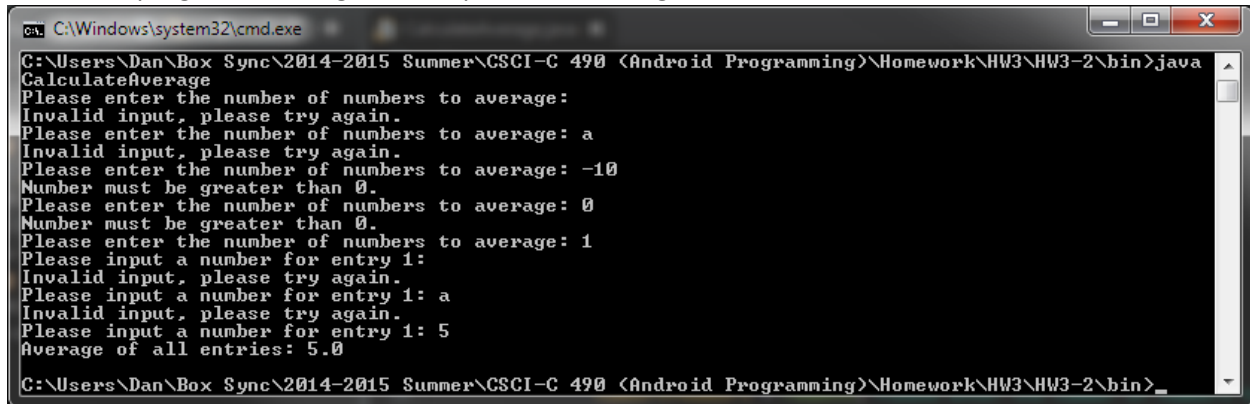
Assignment: Homework 3 Part 2

Date: 2015-07-10

```
1  /*-----*/
2  * Author:      Dan Cassidy
3  * Date:        2015-07-10
4  * Assignment:  HW3-2
5  * Source File: CalculateAverage.java
6  * Language:    Java
7  * Course:      CSCI-C 490, Android Programming, MoWe 08:00
8  -----*/
9  import java.util.Scanner;
10
11 /**
12  * A small class to calculate the average of a given number of integers.
13  *
14  * @author Dan Cassidy
15  */
16 public class CalculateAverage
17 {
18
19     public static void main(String[] args)
20     {
21         int numberOfNumbers = 0;
22         boolean valid = false;
23         Scanner consoleInput = new Scanner(System.in);
24
25         // Loop while input is not valid.
26         while (!valid)
27         {
28             try
29             {
30                 numberOfNumbers = readInt("Please enter the number of numbers to average: ");
31                 if (numberOfNumbers <= 0)
32                     throw new Exception("Number must be greater than 0.");
33                 valid = true;
34             }
35             catch (Exception ex)
36             {
37                 System.out.println(ex.getMessage());
38             }
39         }
40
41         // Declare an array of the specified size and then ask for input for all elements.
42         int[] numbers = new int[numberOfNumbers];
43         for (int counter = 0; counter < numbers.length; counter++)
44             numbers[counter] = readInt("Please input a number for entry " + (counter + 1) + ": ");
45
46         System.out.println("Average of all entries: " + average(numbers));
47     }
48
49     /**
50     * Computes the average (arithmetic mean) of an array of numbers. If <b>numbers</b> is null or
51     * an empty array, 0 is returned.
52     *
53     * @param numbers An array of integers, from which their average will be computed.
54     * @return double, representing the average of the elements contained in <b>numbers</b>.
55     */
56     public static double average(int[] numbers)
57     {
58         // No need to throw an error, just return 0 if the argument is no good.
59         if (numbers == null || numbers.length == 0)
60             return 0;
```

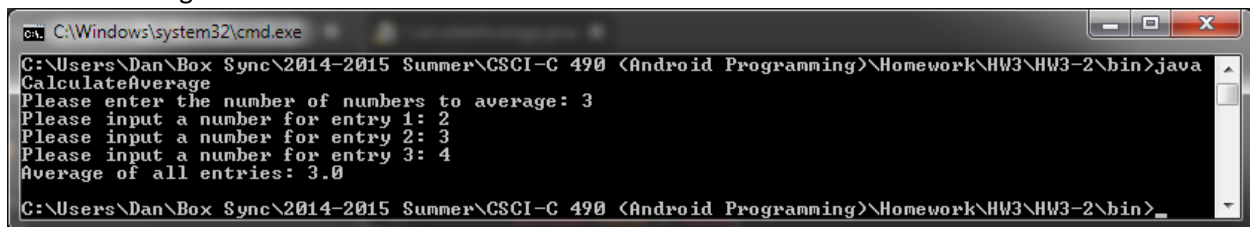
```
61
62     // Compute and return the average.
63     double sum = 0;
64     for (int number : numbers)
65         sum += number;
66     return sum / numbers.length;
67 }
68
69 /**
70  * Reads an integer from the console.
71  *
72  * @param prompt A String object containing the prompt text for a user entering a number.
73  * @return int, holding the integer read from the console.
74  */
75 public static int readInt(String prompt)
76 {
77     int number = 0;
78     boolean valid = false;
79     Scanner consoleInput = new Scanner(System.in);
80
81     // Loop while input is not valid.
82     while (!valid)
83     {
84         try
85         {
86             System.out.print(prompt);
87             number = Integer.parseInt(consoleInput.nextLine());
88             valid = true;
89         }
90         catch (NumberFormatException ex)
91         {
92             System.out.println("Invalid input, please try again.");
93         }
94     }
95
96     return number;
97 }
98 }
99
```

Shows the program handling invalid input, then the edge case of 1 number.



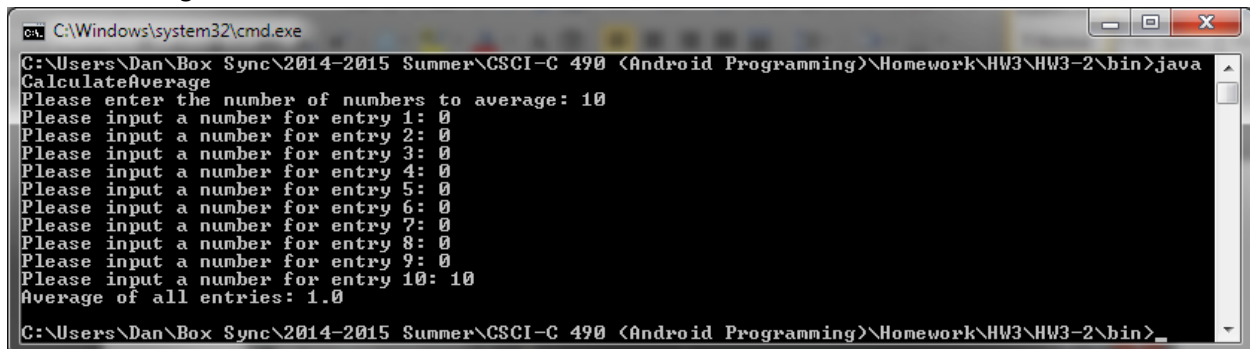
```
C:\Windows\system32\cmd.exe
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW3\HW3-2\bin>java
CalculateAverage
Please enter the number of numbers to average:
Invalid input, please try again.
Please enter the number of numbers to average: a
Invalid input, please try again.
Please enter the number of numbers to average: -10
Number must be greater than 0.
Please enter the number of numbers to average: 0
Number must be greater than 0.
Please enter the number of numbers to average: 1
Please input a number for entry 1:
Invalid input, please try again.
Please input a number for entry 1: a
Invalid input, please try again.
Please input a number for entry 1: 5
Average of all entries: 5.0
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW3\HW3-2\bin>_
```

Shows handling 3 numbers.



```
C:\Windows\system32\cmd.exe
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW3\HW3-2\bin>java
CalculateAverage
Please enter the number of numbers to average: 3
Please input a number for entry 1: 2
Please input a number for entry 2: 3
Please input a number for entry 3: 4
Average of all entries: 3.0
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW3\HW3-2\bin>_
```

Shows handling 10 numbers.



```
C:\Windows\system32\cmd.exe
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW3\HW3-2\bin>java
CalculateAverage
Please enter the number of numbers to average: 10
Please input a number for entry 1: 0
Please input a number for entry 2: 0
Please input a number for entry 3: 0
Please input a number for entry 4: 0
Please input a number for entry 5: 0
Please input a number for entry 6: 0
Please input a number for entry 7: 0
Please input a number for entry 8: 0
Please input a number for entry 9: 0
Please input a number for entry 10: 10
Average of all entries: 1.0
C:\Users\Dan\Box Sync\2014-2015 Summer\CSCI-C 490 (Android Programming)\Homework\HW3\HW3-2\bin>_
```

Name: Dan Cassidy

Class: CSCI-C 490, Mobile Application Development

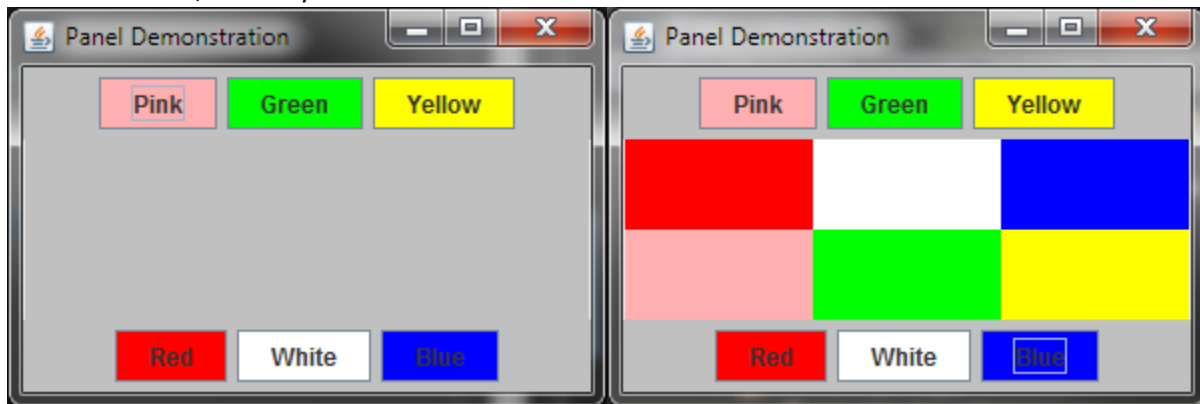
Assignment: Homework 3 Part 3

Date: 2015-07-13

```
1  /**
2   * Default constructor. Handles the setup of all the GUI elements.
3   */
4  public ModifiedPanelDemo()
5  {
6      super("Panel Demonstration");
7      setSize(WIDTH, HEIGHT);
8      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
9      setLayout(new BorderLayout());
10
11     // Set up the top row of buttons.
12     JPanel buttonPanel1 = new JPanel();
13     buttonPanel1.setBackground(Color.LIGHT_GRAY);
14     buttonPanel1.setLayout(new FlowLayout());
15
16     JButton pinkButton = new JButton("Pink");
17     pinkButton.setBackground(Color.PINK);
18     pinkButton.addActionListener(this);
19     buttonPanel1.add(pinkButton);
20
21     JButton greenButton = new JButton("Green");
22     greenButton.setBackground(Color.GREEN);
23     greenButton.addActionListener(this);
24     buttonPanel1.add(greenButton);
25
26     JButton yellowButton = new JButton("Yellow");
27     yellowButton.setBackground(Color.YELLOW);
28     yellowButton.addActionListener(this);
29     buttonPanel1.add(yellowButton);
30
31     add(buttonPanel1, BorderLayout.NORTH);
32
33     // Set up the main color panel.
34     JPanel biggerPanel = new JPanel();
35     biggerPanel.setLayout(new GridLayout(2, 3));
36
37     // Begin new row in main color panel.
38     redPanel = new JPanel();
39     redPanel.setBackground(Color.LIGHT_GRAY);
40     biggerPanel.add(redPanel);
41
42     whitePanel = new JPanel();
43     whitePanel.setBackground(Color.LIGHT_GRAY);
44     biggerPanel.add(whitePanel);
45
46     bluePanel = new JPanel();
47     bluePanel.setBackground(Color.LIGHT_GRAY);
48     biggerPanel.add(bluePanel);
49
50     // Begin new row in main color panel.
51     pinkPanel = new JPanel();
52     pinkPanel.setBackground(Color.LIGHT_GRAY);
53     biggerPanel.add(pinkPanel);
54
55     greenPanel = new JPanel();
56     greenPanel.setBackground(Color.LIGHT_GRAY);
57     biggerPanel.add(greenPanel);
58
59     yellowPanel = new JPanel();
60     yellowPanel.setBackground(Color.LIGHT_GRAY);
```

```
61     biggerPanel.add(yellowPanel);
62
63     add(biggerPanel, BorderLayout.CENTER);
64
65     // Set up the bottom row of buttons.
66     JPanel buttonPanel2 = new JPanel();
67     buttonPanel2.setBackground(Color.LIGHT_GRAY);
68     buttonPanel2.setLayout(new FlowLayout());
69
70     JButton redButton = new JButton("Red");
71     redButton.setBackground(Color.RED);
72     redButton.addActionListener(this);
73     buttonPanel2.add(redButton);
74
75     JButton whiteButton = new JButton("White");
76     whiteButton.setBackground(Color.WHITE);
77     whiteButton.addActionListener(this);
78     buttonPanel2.add(whiteButton);
79
80     JButton blueButton = new JButton("Blue");
81     blueButton.setBackground(Color.BLUE);
82     blueButton.addActionListener(this);
83     buttonPanel2.add(blueButton);
84
85     add(buttonPanel2, BorderLayout.SOUTH);
86 }
87
88 /**
89  * Handles events generated by the buttons.
90  *
91  * @param e Specifies the generated event.
92  */
93 public void actionPerformed(ActionEvent e)
94 {
95     String buttonString = e.getActionCommand();
96
97     if (buttonString.equals("Pink"))
98         pinkPanel.setBackground(Color.PINK);
99     else if (buttonString.equals("Green"))
100         greenPanel.setBackground(Color.GREEN);
101     else if (buttonString.equals("Yellow"))
102         yellowPanel.setBackground(Color.YELLOW);
103     else if (buttonString.equals("Red"))
104         redPanel.setBackground(Color.RED);
105     else if (buttonString.equals("White"))
106         whitePanel.setBackground(Color.WHITE);
107     else if (buttonString.equals("Blue"))
108         bluePanel.setBackground(Color.BLUE);
109     else
110         System.out.println("Unexpected error.");
111 }
112
```

Default window, and fully colored window.



Demonstrating that each of the buttons affects only their own panels.





Name: Dan Cassidy

Class: CSCI-C 490, Mobile Application Development

Assignment: Homework 3 Part 4

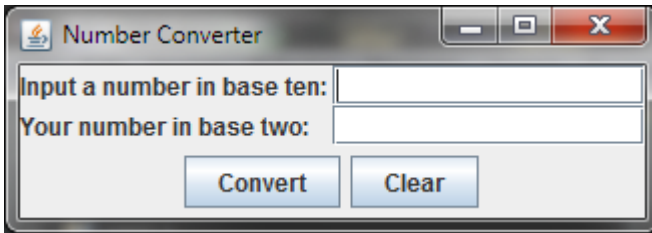
Date: 2015-07-13

```
1  /*-----*/
2  * Author:      Dan Cassidy
3  * Date:        2015-07-13
4  * Assignment:  HW3-4
5  * Source File: NumberConverter.java
6  * Language:    Java
7  * Course:      CSCI-C 490, Android Programming, MoWe 08:00
8  -----*/
9  import java.awt.event.ActionEvent;
10 import java.awt.event.ActionListener;
11 import java.awt.BorderLayout;
12 import java.awt.FlowLayout;
13 import java.awt.GridLayout;
14
15 import javax.swing.JButton;
16 import javax.swing.JFrame;
17 import javax.swing.JLabel;
18 import javax.swing.JPanel;
19 import javax.swing.JTextField;
20
21 /**
22  * Small GUI-based program to convert a base ten number into a base two number.
23  *
24  * @author Dan Cassidy
25  */
26 public class NumberConverter extends JFrame implements ActionListener
27 {
28     private JTextField textBaseTen;
29     private JTextField textBaseTwo;
30
31     /**
32      * Entry point for the class.
33      *
34      * @param args Command line arguments. <i>Ignored</i>.
35      */
36     public static void main(String[] args)
37     {
38         NumberConverter gui = new NumberConverter();
39         gui.setVisible(true);
40     }
41
42     /**
43      * Default constructor. Handles the setup of all the GUI elements.
44      */
45     public NumberConverter()
46     {
47         super("Number Converter");
48         setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
49         setLayout(new BorderLayout());
50
51         // Set up the labels and input text boxes.
52         JPanel inputPanel = new JPanel();
53         inputPanel.setLayout(new GridLayout(2, 2));
54
55         JLabel labelBaseTen = new JLabel("Input a number in base ten: ");
56         inputPanel.add(labelBaseTen);
57
58         textBaseTen = new JTextField();
59         inputPanel.add(textBaseTen);
60
```

```
61     JLabel labelBaseTwo = new JLabel("Your number in base two: ");
62     inputPanel.add(labelBaseTwo);
63
64     textBaseTwo = new JTextField();
65     inputPanel.add(textBaseTwo);
66
67     add(inputPanel, BorderLayout.NORTH);
68
69     // Set up the buttons.
70     JPanel buttonsPanel = new JPanel();
71     buttonsPanel.setLayout(new FlowLayout());
72
73     JButton buttonConvert = new JButton("Convert");
74     buttonConvert.addActionListener(this);
75     buttonsPanel.add(buttonConvert);
76
77     JButton buttonClear = new JButton("Clear");
78     buttonClear.addActionListener(this);
79     buttonsPanel.add(buttonClear);
80
81     add(buttonsPanel, BorderLayout.SOUTH);
82
83     // Make the UI arrange itself.
84     pack();
85 }
86
87 /**
88  * Handles events generated by the buttons.
89  *
90  * @param e Specifies the generated event.
91  */
92 @Override
93 public void actionPerformed(ActionEvent e)
94 {
95     String buttonString = e.getActionCommand();
96
97     // Convert button was clicked.
98     if (buttonString.equals("Convert"))
99     {
100         try
101         {
102             textBaseTwo.setText(convertToBaseTwo(textBaseTen.getText()));
103         }
104         catch (IllegalArgumentException ex)
105         {
106             textBaseTen.setText("Error: Invalid number.");
107         }
108         catch (Exception ex)
109         {
110             textBaseTen.setText(ex.getMessage());
111             ex.printStackTrace();
112         }
113         textBaseTen.requestFocus();
114     }
115     // Clear button was clicked.
116     else if (buttonString.equals("Clear"))
117     {
118         textBaseTen.setText("");
119         textBaseTwo.setText("");
120         textBaseTen.requestFocus();
121     }
122 }
```

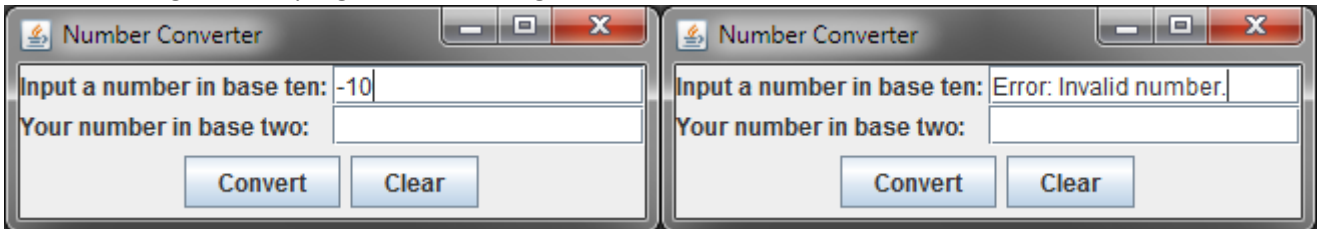
```
121         }
122         // Error.
123         else
124             System.out.println("Unexpected error.");
125     }
126
127     /**
128     * Converts a base ten number into a base two number.
129     *
130     * @param inputNumber Contains the number to be converted from base ten to base two.
131     * @return String, holding the number to be displayed.
132     * @throws NumberFormatException if <b>inputNumber</b> cannot be parsed into an integer.
133     * @throws IllegalArgumentException if <b>inputNumber</b> is negative.
134     */
135     private String convertToBaseTwo(String inputNumber)
136     {
137         String result = "";
138
139         // Try to parse the input string, then check if the number is good. If parsing fails or the
140         // number is bad, exceptions are thrown.
141         int number = Integer.parseInt(inputNumber);
142         if (number < 0)
143             throw new IllegalArgumentException();
144
145         // Handle the number.
146         if (number == 0)
147             result = "0";
148         else
149             while (number != 0)
150             {
151                 result = (number % 2) + result;
152                 number /= 2;
153             }
154
155         return result;
156     }
157
158 }
159
```

Default window.



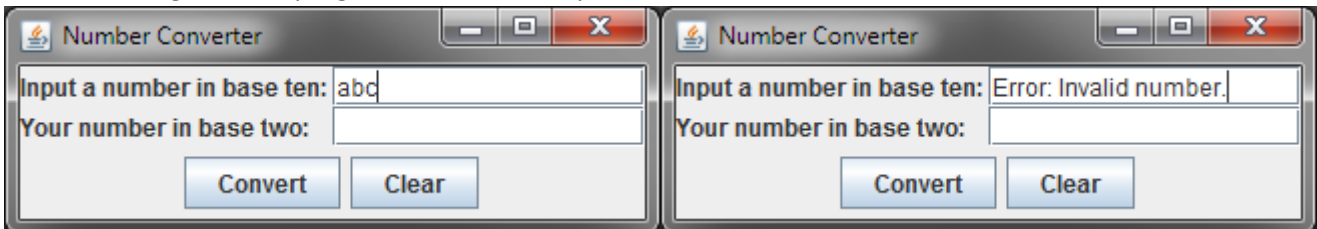
The screenshot shows a window titled "Number Converter" with a standard Windows-style title bar (minimize, maximize, close buttons). Inside the window, there are two text input fields. The first is labeled "Input a number in base ten:" and is empty. The second is labeled "Your number in base two:" and is also empty. Below the input fields are two buttons: "Convert" and "Clear".

Demonstrating how the program handles negative values.



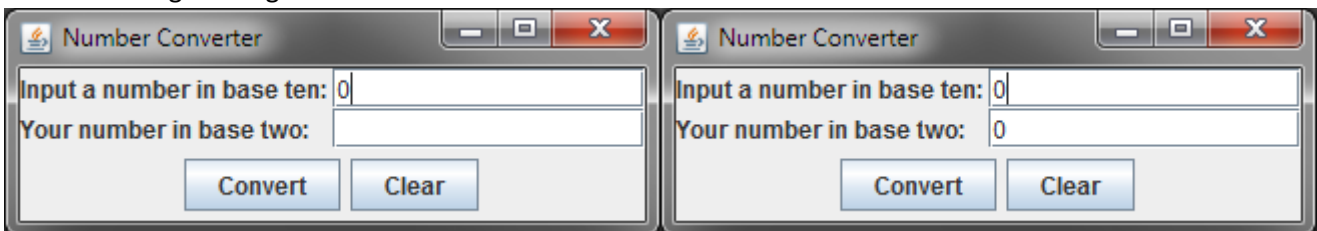
Two side-by-side screenshots of the "Number Converter" window. The left window shows the "Input a number in base ten:" field containing the value "-10". The right window shows the same field containing the text "Error: Invalid number.", indicating that the program does not accept negative values.

Demonstrating how the program handles bad input.



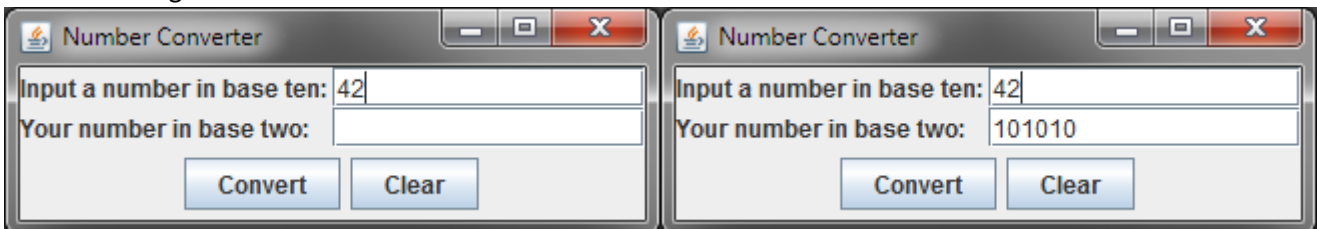
Two side-by-side screenshots of the "Number Converter" window. The left window shows the "Input a number in base ten:" field containing the text "abc". The right window shows the same field containing the text "Error: Invalid number.", indicating that the program does not accept non-numeric input.

Demonstrating the edge case of '0'.



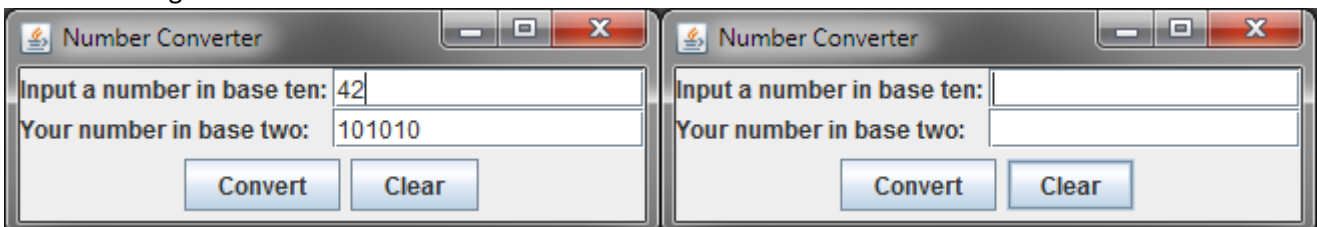
Two side-by-side screenshots of the "Number Converter" window. The left window shows the "Input a number in base ten:" field containing the value "0". The right window shows the same field containing "0", and the "Your number in base two:" field containing the value "0", demonstrating the correct conversion of the edge case.

Demonstrating a normal case.



Two side-by-side screenshots of the "Number Converter" window. The left window shows the "Input a number in base ten:" field containing the value "42". The right window shows the same field containing "42", and the "Your number in base two:" field containing the value "101010", demonstrating a successful conversion of a normal number.

Demonstrating the clear action.



Two side-by-side screenshots of the "Number Converter" window. The left window shows the "Input a number in base ten:" field containing "42" and the "Your number in base two:" field containing "101010". The right window shows the same fields after the "Clear" button was pressed; both fields are now empty.