# IT 114 - ADVANCED PROGRAMMING FOR INFORMATION TECHNOLOGY

## Programming Assignment 1: Vote for Student Body

Author: Jennifer Soh

GitHub Repository: https://github.com/00savethepandas/Java-Assignment-2

## 1. Formulating the Problem

### 1.1 Assignment Description

Design and implement a Java program that determines the winner of a race for Student Body President. The program GUI should have three labeled text fields, each representing a candidate. The user should be able to cast a vote for a particular candidate by clicking a button. A read-only text area should display the accumulated total votes for each candidate.

### 1.2 Verbalization

*What is the goal?*

To create a program that collects votes for several Student Body candidates and displays the results with a GUI. The GUI should consist of labels for each candidate, a button to vote for the candidate, and an area that displays each candidate's total votes.

*What are the givens?*

The givens are the names and images of each candidate. Another given would be that all candidate total votes will start at zero before any votes are placed.

*What are the unknowns?*

The unknowns are the total number of votes each candidate will accumulate.

### 1.3 Information Elicitation

*Goal*

Provide a GUI for the user to place a vote and have the program accumulate and display those votes on the GUI.

*Givens*

The givens are the names and images of the candidates and that the number of votes begins at zero before any votes are placed using the GUI.

*Unknowns*

The total number of votes for each candidate.

*Conditions*

## 2. Planning the Solution

### 2.1 Solution Strategy

Upon starting the program, three images of each candidate will be displayed on the GUI. These images will act as vote buttons with the candidates name displayed under each candidate's image. Below the name tags will be a one-sentence instruction on how to vote for a candidate. Below the instructions will be a panel that displays the accumulated results from each vote placed when the images of the candidates are clicked.

### 2.2 Goal Decomposition

*Sub-goal 1*
Create buttons for each candidate

*Sub-goal 2*
Respond to user button clicks by accumulating votes

*Sub-goal 3*
Display votes.

### 2.3 Resources

*Relevant formulas*
Votes = Votes + 1; (Votes++)

*Formula Derivation*
The user places the vote. The vote is added to the current number of votes.

### 2.4 Data Organization and Description

Input (givens):

| Name | Description | Origin | Used in Sub-goal # |
|---|---|---|---|
| *candidate*Button | Event Object for Event | User | 1 |
| vote*Candidate*Listener | Responds to event | User | 2 |

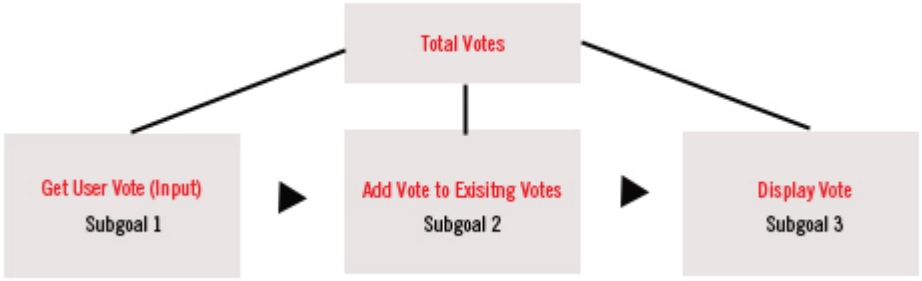
Output (unknowns):

| Name | Description | Origin | Used in Sub-goal # |
|---|---|---|---|
| Vote*CandidateClass*> drawstring | Output the Votes | Screen | 3 |

## 3. Designing the Solution

### 3.1 Structure Chart

*First Level Decomposition*

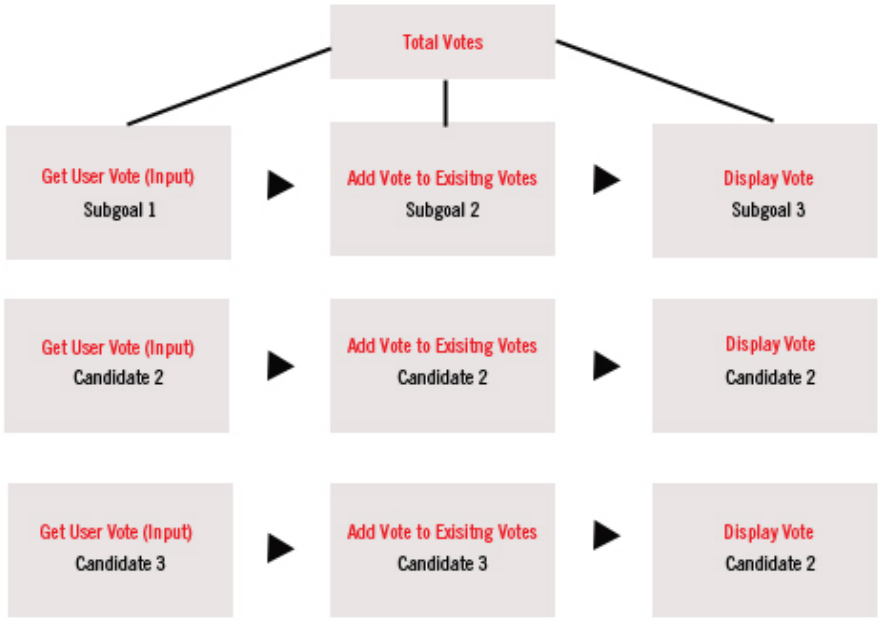



The first level decomposition shows the broad outline of the whole program. The steps are to get input from the User on the GUI (button click) for the candidate. The one is added to the accumulated number of votes and then posted to the GUI for view.

*Goal Refinement*

**Sub-goal 1**
Get account vote from the user
**Sub-goal 2**
Add the vote to existing votes
**Sub-goal 3**
Display votes.

*Second Level Decomposition*




The second level shows the repetition of the process for all three candidates. The program will collect and add to the pool of votes independently from eachother for each candidate.

### 3.2 Module and Data Specifications

**Class: voteStudentBody**
**Name:** voteStudentBody extends JFrame and holds everything (subclasses, variables, constructors, panels etc).
**Global Variables:** v*Candidate*: instantiates vote*Candidate*Class.
(vMickey, vMitch, vJohn)
**Subclasses:**
**voteStudentBody():** Constructor that contains:
vote*Candidate*Class: These classes hold the methods used to draw the total vote output on the GUI. Extends JPanel, repaints vote numbers every time the buttons are clicked. (vMickey, vMitch, and vJohn).
vote*Candidate*Listener: listeners that implement ActionListener, overrides action event and posts the total votes from the vote*Candidate*Class.
**Variables:**

- **canPanel:** creates the candidate panel to hold image buttons.
- **mickey, mitchell, john**: ImageIcons that point to their image resources in img directory.
- **mickeyButton, mitchButton, johnButton**: Creates buttons out of the images.

- **namePanel:** GridLayout Panel holds labels for the image buttons.
- **mLabel, mitchLabel, johnLabel:** Labels for each candidate button displayed.
- **Counters**: GridLayout to hold the label information for each of the candidates.
- **showMickey, showMitchell, showJohn:** actual string labels for the output.
- **Results:** JPanel FlowLayout that holds the results from the vote*Candidate*Classes.
- **Instruct:** JLabel instructions for how to vote.
- **Panel4**: BorderLayout panel to contain canPanel, namePanel, and results.

**Class: vote*Candidate*Class**
**Name:** voteMickClass, voteMitchClass, voteJohnClass: classes increment a variable each time the event object is clicked, and repaints it to the GUI.
**Variables:** mickVotes, mitchVotes, johnVotes: these variables hold the incremented votes each time their event objects are clicked.
**Method:** total*Candidate*Votes(): methods hold the statements that increment the variables.
**Logic:** total*Candidate*Votes() is called each time the button is clicked. The variables are incremented outside of these methods, and the output is repainted every time a these variables are incremented.

**Class: vote*Candidate*Listener**
**Name**: voteMickListener, voteMitchListener, voteJohnListener: classes implements ActionListener, overrides actionPerformed with global variable instances of vote*Candidate*Listener
**Input**: User input
**Output**: None
**Logic**: listen for when the button is clicked. When it is clicked, perform the methods defined in the vote*Candidate*Class of the instance variable.

**Data:**

| Name | Type | Structure |
|---|---|---|
| *candidate*Votes | Real number | Integer parsed into String |

### 3.3 Algorithm

*Logic*
1.0 Create and render the image buttons
2.0 Listen for User input (clicks)
3.0 Respond to user clicks:
    3.1: action listener invokes voteCandidateClasses
    3.2: voteCandidateClasses increment the variable holding total votes.
    3.4: voteCandidateClasses repaints the number of votes.

*Algorithm Description*

This program was pretty complex in terms of rendering a GUI, listening for user input, and then rendering the total number of accumulated input. There isn't much of an algorithm except that each candidate class has a variable that is incremented with ++ to keep adding one to the total number of votes.

## 4. Translation

### 4.1 Source Code

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class voteStudentBody extends JFrame {
   voteMickClass vMickey = new voteMickClass();
   voteMitchClass vMitch = new voteMitchClass();
   voteJohnClass vJohn = new voteJohnClass();
   public voteStudentBody(){
   /* ************************************************************

   1. canPanel: Candidate Button Image Panel. North Panel

   ************************************************************ */
      // Create candidate panel, set it to GridLayout
      JPanel canPanel = new JPanel();
      canPanel.setLayout(new GridLayout(1,3));

      // Create Image Icons
      ImageIcon mickey = new ImageIcon("img/mickey.jpg");
      ImageIcon mitchell = new ImageIcon("img/mitchell.jpg");
      ImageIcon john = new ImageIcon("img/john.jpg");

      // Turn image icons into buttons
      JButton mickeyButton = new JButton(mickey);
      JButton mitchButton = new JButton(mitchell);
      JButton johnButton = new JButton(john);

     // Add Buttons to panels:
      canPanel.add(mickeyButton);
      canPanel.add(mitchButton);
      canPanel.add(johnButton);

      // Attach action listeners to buttons:
      mickeyButton.addActionListener(new voteMickListener());
      mitchButton.addActionListener(new voteMitchListener());
      johnButton.addActionListener(new voteJohnListener());

   /* ************************************************************

   2. namePanel: Holds Candidate Names. Center Panel

   ************************************************************ */
      JPanel namePanel = new JPanel();
      namePanel.setLayout(new GridLayout(1,3));
```

```
        // Candidate Names: Labels for each image:
        JLabel mLabel = new JLabel("Mickey McCormick");
        JLabel mitchLabel = new JLabel("Mitchell Chen");
        JLabel johnLabel = new JLabel("John Fisher");

        // Set Alignment and font size for Mickey's Label:
        mLabel.setHorizontalAlignment(JLabel.CENTER);
        mLabel.setFont(mLabel.getFont().deriveFont(18.0f));

        // Set alignment and font for Mitch's label:
        mitchLabel.setHorizontalAlignment(JLabel.CENTER);
        mitchLabel.setFont(mitchLabel.getFont().deriveFont(18.0f));

        // Set alignment and font for John's lable:
        johnLabel.setHorizontalAlignment(JLabel.CENTER);
        johnLabel.setFont(johnLabel.getFont().deriveFont(18.0f));

        // Add all candidate labels to the namePanel
        namePanel.add(mLabel);
        namePanel.add(mitchLabel);
        namePanel.add(johnLabel);

    /* *************************************************************

    3. Results Panel: counters

    ************************************************************* */

        JPanel counters = new JPanel(new GridLayout(2,3));
        JLabel showMickey = new JLabel("Mickey's Total Votes: "); //
create
        showMickey.setFont(new Font("Arial",Font.BOLD,18));
        JLabel showMitchell = new JLabel("Mitchell's Total Votes: "); //
create
        showMitchell.setFont(new Font("Arial",Font.BOLD,18));
        JLabel showJohn = new JLabel("John's Total Votes: "); // create
        showJohn.setFont(new Font("Arial",Font.BOLD,18));
        counters.add(showMickey);
        counters.add(showMitchell);
        counters.add(showJohn);
        counters.add(vMickey);
        counters.add(vMitch);
        counters.add(vJohn);
        counters.setPreferredSize(new Dimension(800,100));

    /* *************************************************************

    4. South Panel: Holds Results and Instructions

    ************************************************************* */

        JPanel results = new JPanel(new FlowLayout());
        JLabel instruct = new JLabel("CLICK ON THE CANDIDATE TO PLACE
YOUR VOTE"); // create
        instruct.setFont(new Font("Arial",Font.BOLD,18));
        instruct.setForeground(Color.RED);
        results.add(instruct);
```

```
      results.add(counters);
      results.setPreferredSize(new Dimension(800,150));



   /* ************************************************************

   5. Panel 4: Pull together other panels. South Panel

   ************************************************************ */

      JPanel panel4 = new JPanel(new BorderLayout());
      panel4.add(canPanel, BorderLayout.NORTH);
      panel4.add(namePanel, BorderLayout.CENTER);
      panel4.add(results, BorderLayout.SOUTH);
      add(panel4);
   }

   // Start program and render everything.

   public static void main(String[] args){
      voteStudentBody frame = new voteStudentBody();
      frame.setTitle("Vote for Student Body");
      frame.setSize(900,500);
      frame.setLocationRelativeTo(null);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setVisible(true);
   }
  /* ************************************************************

   6. Listener classes (voteCandidateListener)

   ************************************************************ */
   class voteMickListener implements ActionListener {
      @Override
      public void actionPerformed(ActionEvent e){
         vMickey.totalMickVotes();
      }
   }

   class voteMitchListener implements ActionListener {
      @Override
      public void actionPerformed(ActionEvent e){
         vMitch.totalMitchVotes();
      }
   }
   class voteJohnListener implements ActionListener {
      @Override
      public void actionPerformed(ActionEvent e){
         vJohn.totalJohnVotes();
      }
   }
   /* ************************************************************

   7. Functions to perform (voteCandidateClass):

   ************************************************************ */
```

```
    class voteMickClass extends JPanel {
       private int mickVotes = 0;
       public void totalMickVotes(){
          mickVotes++;
          repaint();
       }
       @Override
       public void paintComponent(Graphics g){
          super.paintComponent(g);
          g.setFont(new Font("Arial", Font.PLAIN, 25));
          g.drawString(String.valueOf(mickVotes),80,20);
       }
    }

    class voteMitchClass extends JPanel {
       private int mitchVotes = 0;
       public void totalMitchVotes(){
          mitchVotes++;
          repaint();
       }
       @Override
      public void paintComponent(Graphics g){
          super.paintComponent(g);
          g.setFont(new Font("Arial", Font.PLAIN, 25));
          g.drawString(String.valueOf(mitchVotes),85,20);
       }
    }

       class voteJohnClass extends JPanel {
       private int johnVotes = 0;
       public void totalJohnVotes(){
          johnVotes++;
          repaint();
       }
       @Override
       public void paintComponent(Graphics g){
         super.paintComponent(g);
          g.setFont(new Font("Arial", Font.PLAIN, 25));
          g.drawString(String.valueOf(johnVotes),80,20);
       }
    }
}
```

**4.2 <u>Program and Module Description</u>**

Class: vote*Candidate*Class

Specifically: voteMickClass, voteMitchClass, voteJohnClass. These classes increment a variable each time the event object is clicked, and repaints it to the GUI.

Class: vote*Candidate*Listener

Specifically: voteMickListener, voteMitchListener, voteJohnListener. These classes implement ActionListener, overrides actionPerformed with global variable instances of vote*Candidate*Listener.

Main

The main function is fairly simple. It creates an instance of the voteStudentBody and stores if in the variable *frame*. It then sets the title of the frame to “Vote Student Body”. The size is set to 900px by 500px. The location is set to the default null so it appears in the center. The default close operation is set to closing the program when the window is closed. The visibility of the frame is set to true to make the frame visible.

## 5. Solution Testing

Test the program with following data domain:

I tested the program by clicking on each candidate button in the same window and watching the output increment by one in response to each click.

## 6. Testing: Output


## 7. Project Notes

I definitely feel that this program could be written better since I noticed a lot of redundant code. I think if I learned more about abstract classes and how to implement them, I might be able to cut down on a lot of it and make it more simple and streamlined. I think that the issues that I would come across, with my limited knowledge, is the extension of classes and how that would affect the ActionListeners among other things.

```
// ***************************************************************
//   voteStudentBody.java
//   Author: Jennifer Soh  ID: JS542
//   Compiler Used: JGrasp
//   Create a GUI that provides buttons to vote for Student Body.
//   Increment respective variables when a vote it placed.
//   Display the total votes for each candidate.
// ***************************************************************

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class voteStudentBody extends JFrame {
   voteMickClass vMickey = new voteMickClass();
   voteMitchClass vMitch = new voteMitchClass();
   voteJohnClass vJohn = new voteJohnClass();
   public voteStudentBody(){
   /* ************************************************************

   1. canPanel: Candidate Button Image Panel. North Panel

   ************************************************************ */
      // Create candidate panel, set it to GridLayout
      JPanel canPanel = new JPanel();
      canPanel.setLayout(new GridLayout(1,3));

      // Create Image Icons
      ImageIcon mickey = new ImageIcon("img/mickey.jpg");
      ImageIcon mitchell = new ImageIcon("img/mitchell.jpg");
      ImageIcon john = new ImageIcon("img/john.jpg");

      // Turn image icons into buttons
      JButton mickeyButton = new JButton(mickey);
      JButton mitchButton = new JButton(mitchell);
      JButton johnButton = new JButton(john);

      // Add Buttons to panels:
      canPanel.add(mickeyButton);
      canPanel.add(mitchButton);
      canPanel.add(johnButton);

      // Attach action listeners to buttons:
      mickeyButton.addActionListener(new voteMickListener());
      mitchButton.addActionListener(new voteMitchListener());
      johnButton.addActionListener(new voteJohnListener());

   /* ************************************************************

   2. namePanel: Holds Candidate Names. Center Panel

   ************************************************************ */
      JPanel namePanel = new JPanel();
      namePanel.setLayout(new GridLayout(1,3));

      // Candidate Names: Labels for each image:
      JLabel mLabel = new JLabel("Mickey McCormick");
      JLabel mitchLabel = new JLabel("Mitchell Chen");
      JLabel johnLabel = new JLabel("John Fisher");

      // Set Alignment and font size for Mickey's Label:
      mLabel.setHorizontalAlignment(JLabel.CENTER);
      mLabel.setFont(mLabel.getFont().deriveFont(18.0f));

      // Set alignment and font for Mitch's label:
      mitchLabel.setHorizontalAlignment(JLabel.CENTER);
      mitchLabel.setFont(mitchLabel.getFont().deriveFont(18.0f));
```

```
   // Set alignment and font for John's lable:
   johnLabel.setHorizontalAlignment(JLabel.CENTER);
   johnLabel.setFont(johnLabel.getFont().deriveFont(18.0f));

   // Add all candidate labels to the namePanel
   namePanel.add(mLabel);
   namePanel.add(mitchLabel);
   namePanel.add(johnLabel);

/* ************************************************************

3. Results Panel: counters

************************************************************ */

   JPanel counters = new JPanel(new GridLayout(2,3));
   JLabel showMickey = new JLabel("Mickey's Total Votes: "); // create
   showMickey.setFont(new Font("Arial",Font.BOLD,18));
   JLabel showMitchell = new JLabel("Mitchell's Total Votes: "); // create
   showMitchell.setFont(new Font("Arial",Font.BOLD,18));
   JLabel showJohn = new JLabel("John's Total Votes: "); // create
   showJohn.setFont(new Font("Arial",Font.BOLD,18));
   counters.add(showMickey);
   counters.add(showMitchell);
   counters.add(showJohn);
   counters.add(vMickey);
   counters.add(vMitch);
   counters.add(vJohn);
   counters.setPreferredSize(new Dimension(800,100));

/* ************************************************************

4. South Panel: Holds Results and Instructions

************************************************************ */

   JPanel results = new JPanel(new FlowLayout());
   JLabel instruct = new JLabel("CLICK ON THE CANDIDATE TO PLACE YOUR VOTE"); // create
   instruct.setFont(new Font("Arial",Font.BOLD,18));
   instruct.setForeground(Color.RED);
   results.add(instruct);
   results.add(counters);
   results.setPreferredSize(new Dimension(800,150));



/* ************************************************************

5. Panel 4: Pull together other panels. South Panel

************************************************************ */

   JPanel panel4 = new JPanel(new BorderLayout());
   panel4.add(canPanel, BorderLayout.NORTH);
   panel4.add(namePanel, BorderLayout.CENTER);
   panel4.add(results, BorderLayout.SOUTH);
   add(panel4);
}

// Start program and render everything.

public static void main(String[] args){
   voteStudentBody frame = new voteStudentBody();
   frame.setTitle("Vote for Student Body");
   frame.setSize(900,500);
   frame.setLocationRelativeTo(null);
   frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   frame.setVisible(true);
```

```
}
/* ************************************************************

6. Listener classes (voteCandidateListener)

************************************************************ */
class voteMickListener implements ActionListener {
   @Override
   public void actionPerformed(ActionEvent e){
      vMickey.totalMickVotes();
   }
}

class voteMitchListener implements ActionListener {
   @Override
   public void actionPerformed(ActionEvent e){
      vMitch.totalMitchVotes();
   }
}
class voteJohnListener implements ActionListener {
   @Override
   public void actionPerformed(ActionEvent e){
      vJohn.totalJohnVotes();
   }
}
/* ************************************************************

7. Functions to perform (voteCandidateClass):

************************************************************ */

class voteMickClass extends JPanel {
   private int mickVotes = 0;
   public void totalMickVotes(){
      mickVotes++;
      repaint();
   }
   @Override
   public void paintComponent(Graphics g){
      super.paintComponent(g);
      g.setFont(new Font("Arial", Font.PLAIN, 25));
      g.drawString(String.valueOf(mickVotes),80,20);
   }
}

class voteMitchClass extends JPanel {
   private int mitchVotes = 0;
   public void totalMitchVotes(){
      mitchVotes++;
      repaint();
   }
   @Override
   public void paintComponent(Graphics g){
      super.paintComponent(g);
      g.setFont(new Font("Arial", Font.PLAIN, 25));
      g.drawString(String.valueOf(mitchVotes),85,20);
   }
}

   class voteJohnClass extends JPanel {
   private int johnVotes = 0;
   public void totalJohnVotes(){
      johnVotes++;
      repaint();
   }
   @Override
   public void paintComponent(Graphics g){
      super.paintComponent(g);
```

```
            g.setFont(new Font("Arial", Font.PLAIN, 25));
            g.drawString(String.valueOf(johnVotes),80,20);
        }
    }
}
```