MainMenu Class

```
import java.awt.AWTEvent;
import java.awt.EventQueue;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JPopupMenu;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import java.awt.Toolkit;
import javax.swing.JLabel;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import java.awt.Font;
import javax.swing.SwingConstants;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.awt.event.AWTEventListener;
import java.awt.event.ActionEvent;
import javax.swing.JTable;
import javax.swing.JTextField;
import javax.swing.JScrollPane;
import javax.swing.JComboBox;
import java.awt.event.MouseEvent;
import javax.swing.ImageIcon;
import java.awt.event.KeyAdapter;
import java.awt.event.KeyEvent;
import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;
import java.awt.Color;
import java.awt.SystemColor;
import java.awt.Component;
import javax.swing.JTextArea;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;

public class MainMenu extends Window {

	private JPanel contentPane;

	//decorations
	private JLabel lblLogo;
	private JLabel toothImg1;
```

```
private JLabel toothImg2;

//table parts
private JScrollPane scrollPanePatient;
private DefaultTableModel modelTable;
private JTable tablePatient;

//buttons
private JButton btnAdd;
private JButton btnSave;
private JButton btnQuit;
private JButton btnReset;
private JPopupMenu rightClickMenu;

//sort parts
private JComboBox<String> comboBoxSort;
private JComboBox<String> comboBoxSortOrder;
private JTextArea sortBackground;

//search parts
private JTextField searchLastName;
private JComboBox searchColor;
private JComboBox searchBonding;
private JComboBox searchProcedure;
private JComboBox searchSex;
private JTextField searchFirstName;
private JComboBox searchMaterial;
private JTextArea searchBackground;

//data
private ArrayList<Patient> patientList;
private boolean saved;


/**
 * Launch the application.
 */
public static void main(String[] args) {
	EventQueue.invokeLater(new Runnable() {
		public void run() {
			Database.createImageFolders();
			Database.loadData();
			Database.printList();
			try {
```

```
                                System.out.println(Database.provideList().size());
                                MainMenu frame = new MainMenu();
                                frame.setVisible(true);
                        } catch (Exception e) {
                                e.printStackTrace();
                        }
                }
        });
}

/**
 * Create the frame.
 */
public MainMenu() {
        saved = false;
        patientList = Database.provideList();
        setGUI();
}

//initializes all the GUI elements and features
public void setGUI()
{
        setBackground(SystemColor.activeCaption);
        setTitle("Restoration Information Manager");

        setBounds(100, 100, 1218, 578);
        contentPane = new JPanel();
        contentPane.setBackground(new Color(240, 255, 240));
        contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

        setContentPane(contentPane);
        contentPane.setLayout(null);

        JLabel lblTitle = new JLabel("Restoration Information \r\nManager");
        lblTitle.setForeground(new Color(0, 128, 0));
        lblTitle.setHorizontalAlignment(SwingConstants.CENTER);
        lblTitle.setFont(new Font("Perpetua", Font.PLAIN, 36));
        lblTitle.setBounds(20, 11, 907, 70);
        contentPane.add(lblTitle);


        setTable();
        setPictures();
        setButtons();
```

```
		setSortFeatures();
		setSearchFeatures();
		setPopup();

	}

	//sets up the data table
	private void setTable()
	{
		modelTable = new DefaultTableModel(null, Patient.CATEGORIES);

		scrollPanePatient = new JScrollPane();
		scrollPanePatient.setBounds(20, 72, 907, 417);
		contentPane.add(scrollPanePatient);

		tablePatient = new JTable(modelTable);
		tablePatient.setSurrendersFocusOnKeystroke(true);
		tablePatient.getTableHeader().setReorderingAllowed(false);
		tablePatient.getTableHeader().setResizingAllowed(false);
		tablePatient.getTableHeader().setBackground(new Color(245, 245, 220));
		tablePatient.setToolTipText("Select Patient and Right Click to Modify/View");

		scrollPanePatient.setViewportView(tablePatient);
		scrollPanePatient.getViewport().setBackground(new Color(245, 255, 245));

		if (patientList.size() > 0)
			loadTable();
	}

	//sets up all the picture elements
	private void setPictures()
	{
		lblLogo = new JLabel("");
		lblLogo.setIcon(new
ImageIcon(Toolkit.getDefaultToolkit().getImage(getClass().getResource("logo.png"))));
		lblLogo.setBounds(937, 23, 299, 113);
		contentPane.add(lblLogo);

		toothImg1 = new JLabel("");
		toothImg1.setAlignmentY(Component.BOTTOM_ALIGNMENT);
		toothImg1.setIcon(new
ImageIcon(Toolkit.getDefaultToolkit().getImage(getClass().getResource("tooth.png"))));
		toothImg1.setBounds(20, 11, 178, 125);
		contentPane.add(toothImg1);
```

```
		toothImg2 = new JLabel("");
		toothImg2.setAlignmentY(Component.BOTTOM_ALIGNMENT);
		toothImg2.setIcon(new
ImageIcon(Toolkit.getDefaultToolkit().getImage(getClass().getResource("tooth.png"))));
		toothImg2.setBounds(816, 11, 158, 125);
		contentPane.add(toothImg2);
	}

	//sets up all buttons
	private void setButtons()
	{
		btnAdd = new JButton("Add");
		btnAdd.setToolTipText("Add a Patient");
		btnAdd.setBackground(new Color(245, 245, 220));
		btnAdd.setFont(new Font("Tahoma", Font.PLAIN, 15));
		btnAdd.setBounds(24, 500, 82, 28);
		btnAdd.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent arg0) {
				AddEditPatientWindow addWindow = new AddEditPatientWindow(new
Patient());
				addWindow.setVisible(true);
				dispose();
			}
		});
		contentPane.add(btnAdd);

		btnReset = new JButton("RESET");
		btnReset.setToolTipText("Reset all data");
		btnReset.setBackground(new Color(245, 245, 220));
		btnReset.setFont(new Font("Tahoma", Font.PLAIN, 15));
		btnReset.setForeground(new Color(220, 20, 60));
		btnReset.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				reset();
			}
		});
		btnReset.setBounds(742, 500, 82, 28);
		contentPane.add(btnReset);

		btnSave = new JButton("Save");
		btnSave.setToolTipText("Save changes");
		btnSave.setBackground(new Color(245, 245, 220));
		btnSave.setFont(new Font("Tahoma", Font.PLAIN, 15));
```

```
		btnSave.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				saved = true;
				Database.saveData();
				System.out.println("saved");
				Database.printList();
			}
		});
		btnSave.setBounds(116, 500, 82, 28);
		contentPane.add(btnSave);

		btnQuit = new JButton("Quit");
		btnQuit.setToolTipText("Quit program");
		btnQuit.setBackground(new Color(245, 245, 220));
		btnQuit.setFont(new Font("Tahoma", Font.PLAIN, 15));
		btnQuit.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				quit();
			}
		});
		btnQuit.setBounds(847, 500, 82, 28);
		contentPane.add(btnQuit);
	}

	//sets up the sorting features
	private void setSortFeatures()
	{
		JLabel lblSort = new JLabel("Sort Options");
		lblSort.setHorizontalAlignment(SwingConstants.CENTER);
		lblSort.setFont(new Font("Tahoma", Font.PLAIN, 17));
		lblSort.setBounds(947, 147, 219, 21);
		contentPane.add(lblSort);

		comboBoxSort = new JComboBox(Patient.CATEGORIES);
		comboBoxSort.setBackground(new Color(245, 255, 245));
		comboBoxSort.addItemListener(new ItemListener() {
			public void itemStateChanged(ItemEvent e) {
				sortTable();
			}
		});
		comboBoxSort.setFont(new Font("Tahoma", Font.PLAIN, 15));
		comboBoxSort.setBounds(944, 179, 148, 22);
		contentPane.add(comboBoxSort);
```

```
		comboBoxSortOrder = new JComboBox(new String[]{"A->Z", "Z->A"});
		comboBoxSortOrder.setBackground(new Color(245, 255, 245));
		comboBoxSortOrder.setFont(new Font("Tahoma", Font.PLAIN, 15));
		comboBoxSortOrder.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				sortTable();
			}
		});
		comboBoxSortOrder.setBounds(1102, 179, 64, 22);
		contentPane.add(comboBoxSortOrder);

		sortBackground = new JTextArea();
		sortBackground.setBackground(new Color(245, 245, 220));
		sortBackground.setBounds(937, 142, 245, 70);
		sortBackground.setEnabled(false);
		contentPane.add(sortBackground);
	}

	//sets up the search features
	private void setSearchFeatures()
	{
		JLabel lblSearch = new JLabel("Search Filters");
		lblSearch.setHorizontalAlignment(SwingConstants.CENTER);
		lblSearch.setFont(new Font("Tahoma", Font.PLAIN, 17));
		lblSearch.setBounds(947, 226, 219, 21);
		contentPane.add(lblSearch);

		searchLastName = new JTextField();
		searchLastName.setText("Enter here...");
		searchLastName.addFocusListener(new FocusAdapter() {
			@Override
			public void focusGained(FocusEvent e)
			{
				searchLastName.setText("");
			}
			@Override
			public void focusLost(FocusEvent e)
			{
				if (searchLastName.getText().equals(""))
					searchLastName.setText("Enter here...");
			}
		});
		searchLastName.addKeyListener(new KeyAdapter() {
			@Override
```

```
		public void keyReleased(KeyEvent e) {
			searchTable();
		}
});
searchLastName.setBounds(1030, 255, 136, 21);
contentPane.add(searchLastName);
searchLastName.setColumns(10);

JLabel lblLastName = new JLabel("Last Name:");
lblLastName.setFont(new Font("Tahoma", Font.PLAIN, 13));
lblLastName.setBounds(949, 258, 85, 14);
contentPane.add(lblLastName);

JLabel lblFirstName = new JLabel("First Name:");
lblFirstName.setFont(new Font("Tahoma", Font.PLAIN, 13));
lblFirstName.setBounds(949, 294, 85, 14);
contentPane.add(lblFirstName);

searchFirstName = new JTextField();
searchFirstName.addFocusListener(new FocusAdapter() {
		@Override
		public void focusGained(FocusEvent e)
		{
			searchFirstName.setText("");
		}
		@Override
		public void focusLost(FocusEvent e)
		{
			if (searchFirstName.getText().equals(""))
				searchFirstName.setText("Enter here...");
		}
});
searchFirstName.setText("Enter here...");
searchFirstName.addKeyListener(new KeyAdapter() {
		@Override
		public void keyReleased(KeyEvent e) {
			searchTable();
		}
});
searchFirstName.setColumns(10);
searchFirstName.setBounds(1030, 291, 136, 21);
contentPane.add(searchFirstName);

JLabel lblSex = new JLabel("Sex:");
```

```
lblSex.setFont(new Font("Tahoma", Font.PLAIN, 13));
lblSex.setBounds(949, 333, 56, 14);
contentPane.add(lblSex);

searchSex = new JComboBox(Patient.SEX);
searchSex.setBackground(new Color(245, 255, 245));
searchSex.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e)
	{
		searchTable();
	}
});
searchSex.setBounds(1030, 330, 136, 22);
contentPane.add(searchSex);

JLabel lblProcedure = new JLabel("Procedure:");
lblProcedure.setFont(new Font("Tahoma", Font.PLAIN, 13));
lblProcedure.setBounds(949, 367, 71, 14);
contentPane.add(lblProcedure);

searchProcedure = new JComboBox(Patient.PROCEDURES);
searchProcedure.setBackground(new Color(245, 255, 245));
searchProcedure.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e)
	{
		searchTable();
	}
});
searchProcedure.setBounds(1030, 364, 136, 22);
contentPane.add(searchProcedure);

JLabel lblMaterial = new JLabel("Material:");
lblMaterial.setFont(new Font("Tahoma", Font.PLAIN, 13));
lblMaterial.setBounds(947, 410, 102, 14);
contentPane.add(lblMaterial);

searchMaterial = new JComboBox(Patient.MATERIALS);
searchMaterial.setBackground(new Color(245, 255, 245));
searchMaterial.addActionListener(new ActionListener() {
	public void actionPerformed(ActionEvent e)
	{
		searchTable();
	}
});
```

```
		searchMaterial.setBounds(1030, 407, 136, 22);
		contentPane.add(searchMaterial);

		JLabel lblBonding = new JLabel("Bonding:");
		lblBonding.setFont(new Font("Tahoma", Font.PLAIN, 13));
		lblBonding.setBounds(949, 446, 56, 14);
		contentPane.add(lblBonding);

		searchBonding = new JComboBox(Patient.BONDINGS);
		searchBonding.setBackground(new Color(245, 255, 245));
		searchBonding.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				searchTable();
			}
		});
		searchBonding.setBounds(1030, 443, 136, 22);
		contentPane.add(searchBonding);

		JLabel lblColor = new JLabel("Color:");
		lblColor.setFont(new Font("Tahoma", Font.PLAIN, 13));
		lblColor.setBounds(947, 486, 89, 14);
		contentPane.add(lblColor);

		searchColor = new JComboBox(Patient.COLORS);
		searchColor.setBackground(new Color(245, 255, 245));
		searchColor.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				searchTable();
			}
		});
		searchColor.setBounds(1030, 483, 136, 22);
		contentPane.add(searchColor);

		searchBackground = new JTextArea();
		searchBackground.setBackground(new Color(245, 245, 220));
		searchBackground.setBounds(937, 224, 245, 292);
		searchBackground.setEnabled(false);
		contentPane.add(searchBackground);
}

//sets up the right click popup menu
private void setPopup()
```

```
{
        rightClickMenu = new JPopupMenu();

        JMenuItem editMenuItem = new JMenuItem("Edit");
        JMenuItem viewMenuItem = new JMenuItem("View");
        JMenuItem deleteMenuItem = new JMenuItem("Delete");
        deleteMenuItem.setForeground(Color.RED);

        rightClickMenu.add(editMenuItem);
        rightClickMenu.add(viewMenuItem);
        rightClickMenu.add(deleteMenuItem);


        viewMenuItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        PatientInfoWindow viewWindow = new
PatientInfoWindow(patientList.get(tablePatient.getSelectedRow()),tablePatient.getSelectedRow());
        viewWindow.setVisible(true);
        tablePatient.clearSelection();
        dispose();
      }
    });

        editMenuItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        AddEditPatientWindow editWindow = new
AddEditPatientWindow(patientList.get(tablePatient.getSelectedRow()));
        editWindow.setVisible(true);
        tablePatient.clearSelection();
        dispose();
      }
    });

        deleteMenuItem.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        this.delete();
      }

      private void delete() {
        Database.deleteData(patientList.get(tablePatient.getSelectedRow()));
                modelTable.removeRow(tablePatient.getSelectedRow());
                tablePatient.clearSelection();
        }
    });
```

```
		Toolkit.getDefaultToolkit().addAWTEventListener(new AWTEventListener() {
			public void eventDispatched(AWTEvent awte) {
				if (tablePatient.getSelectedRow() != -1)
					this.showPopup(awte);
			}

			private void showPopup(AWTEvent awte) {
				MouseEvent me = (MouseEvent) awte;
				if (me.isPopupTrigger())
					rightClickMenu.show(me.getComponent(), me.getX(),
me.getY());

			}
		}, 16L);
	}

	//asks the user to save changes and then quit
	public void quit()
	{
		if (!saved)
		{
			int dialogResult = JOptionPane.showConfirmDialog(null,
					" Save unsaved changes?",
					"Quit Confirmation", JOptionPane.ERROR_MESSAGE);
			if (dialogResult == 0) {
				System.out.println("saved");
				Database.saveData();
				Database.printList();
				Database.deleteTempPics();
				dispose();
			}
			else if (dialogResult == 1)
			{
				Database.printList();
				Database.deleteTempPics();
				dispose();
			}
		}
		else
		{
			Database.printList();
			Database.deleteTempPics();
			dispose();
```

```
		}
	}

	//resets all the data and table upon asking the user
	private void reset() {
		int dialogResult = JOptionPane.showConfirmDialog(null,
				"  Are you sure you want to reset this \nprogram? (All data will be
erased.)",
				"Reset Confirmation", JOptionPane.ERROR_MESSAGE);
		if (dialogResult == 0) {
			Database.reset();
			resetTable();
			patientList = new ArrayList<Patient>();
			System.out.println("reset");

		}
	}

	//load the table within a list of patients
	private void loadTable() {
		String[][] table = new String[patientList.size()][Patient.CATEGORIES.length];
		for (int i = 0; i < patientList.size(); i++) {
			Patient pat = patientList.get(i);
			String dateOfBirth = pat.getMonth() + "/" + pat.getDay() + "/" + pat.getYear();
			table[i] = new String[]{ pat.getLastName(), pat.getFirstName(), dateOfBirth,
pat.getSex(),

pat.getProcedure(),pat.getMaterial(),pat.getBonding(),pat.getColor() };

			modelTable = new DefaultTableModel(table, Patient.CATEGORIES) {

				@Override
				public boolean isCellEditable(int row, int column) {
					//all cells false
					return false;
				}
			};
		}
		tablePatient.setModel(modelTable);
	}

	//reset the table (clear it)
	private void resetTable()
	{
```

```
		for (int i = 0; i < patientList.size(); i++)
		{
			modelTable.removeRow(0);
		}
	}

	//sort the table in order based on the select sort method
	private void sortTable()
	{
		resetTable();

		int sortMethod = comboBoxSort.getSelectedIndex();
		ArrayList<Patient> sortedList = new ArrayList<Patient>();
		for (Patient pat : patientList)
		{
			insert(sortedList,pat,sortMethod);
		}

		patientList = sortedList;

		loadTable();
	}

	//insert patient into list based on sort method given (which information to sort by)
	private void insert(ArrayList<Patient> list, Patient pat, int sortMethod)
	{
		int i = 0;

		//determines if data is put into alphabetical or reverse-alphabetical order
		int orderMultipler = (comboBoxSortOrder.getSelectedIndex() == 0) ? 1 : -1;

		String patInfo = pat.getPatientInfo()[sortMethod];

		if (sortMethod != 2) //sort by everything but date of birth
		{
			while (i < list.size() &&
(orderMultipler*(patInfo.compareTo((list.get(i).getPatientInfo())[sortMethod]))) > 0)
				i++;

			list.add(i,pat);
		}
		else  //sort by date of birth
		{
			patInfo = dateConvert(patInfo, pat);
```

```
                        while (i < list.size() &&
(orderMultipler*patInfo.compareTo(dateConvert(list.get(i).getPatientInfo()[sortMethod],list.get(i))) > 0))
                                i++;

                        list.add(i,pat);
                }
        }

        //helper method for the sorting algorithm for date string conversion
        private String dateConvert(String patInfo, Patient pat)
        {
                if (pat.getMonth() < 10)
                        return "0" + patInfo;
                return patInfo;
        }

        //have the table show all patients with information matching the search query
        private void searchTable()
        {
                resetTable();

                patientList = Database.provideList();

                ArrayList<Patient> wantedList = new ArrayList<Patient>();

                for (Patient pat : patientList)
                {
                        if  (fitsFilter(pat))
                        wantedList.add(pat);
                }

                patientList = wantedList;

                loadTable();
        }

        //checks if a patient fits all the filters
        private boolean fitsFilter(Patient pat)
        {
                String firstName = searchFirstName.getText();
                String lastName = searchLastName.getText();
```

```
                return ((firstName.equals("") || firstName.equals("Enter here...")||
((pat.getFirstName().toLowerCase()).indexOf(firstName.toLowerCase()) == 0)) &&
                                (lastName.equals("") || lastName.equals("Enter here...") ||
((pat.getLastName().toLowerCase()).indexOf(lastName.toLowerCase()) == 0)) &&
                                (searchSex.getSelectedIndex() == 0 ||
((String)searchSex.getSelectedItem()).equals(pat.getSex())) &&
                                (searchProcedure.getSelectedIndex() == 0 ||
((String)searchProcedure.getSelectedItem()).equals(pat.getProcedure())) &&
                                (searchMaterial.getSelectedIndex() == 0 ||
((String)searchMaterial.getSelectedItem()).equals(pat.getMaterial())) &&
                                (searchBonding.getSelectedIndex() == 0 ||
((String)searchBonding.getSelectedItem()).equals(pat.getBonding())) &&
                                (searchColor.getSelectedIndex() == 0 ||
((String)searchColor.getSelectedItem()).equals(pat.getColor())));
        }
}
```

Database Class

```
import java.awt.Image;
import java.awt.image.BufferedImage;
import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.PriorityQueue;
import java.util.Scanner;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.filechooser.FileNameExtensionFilter;

public class Database
{
        public static FileNameExtensionFilter filter = new FileNameExtensionFilter("Image
File","png","jpg","jfif","jpeg","gif");
        private static PriorityQueue<Patient> myPatients  = new PriorityQueue<Patient>();
        private static String myFile = "patientData.txt";
```

```
private Database()
{
	//no variables to initialize
}

public static void createImageFolders()
{
	File picDir = new File("Pictures/");
	File tempDir = new File("TempPics/");

	if (!picDir.exists())
		picDir.mkdirs();
	if (!tempDir.exists())
		tempDir.mkdirs();
}

//saves patient data to the text file
public static void saveData()
{
	//save patients
	Iterator<Patient> iter = myPatients.iterator();
	PrintWriter writer;
	try
	{
		writer = new PrintWriter(new File(myFile));
		while(iter.hasNext())
		{
			Patient pat = iter.next();
			System.out.println(pat);
			writer.print(pat.toString());
		}
		writer.close();
	}
	catch (Exception e)
	{
		System.out.println("Error: " + e.getMessage());
	}
	printList();


	//save photos
	File tempDir = new File("TempPics/");
	File saveDir = new File ("Pictures/");
	File[] tempPics = tempDir.listFiles();
```

```
		if (tempPics != null)
		{
			for (File pic: tempPics)
			{
				String oldPath = pic.getPath();
				String newPath = saveDir.getAbsolutePath() + "/" + pic.getName();
				System.out.println(oldPath + ", " + newPath);
				try {
					Files.copy(Paths.get(oldPath), Paths.get(newPath),
StandardCopyOption.REPLACE_EXISTING);
				} catch (IOException e) {
					e.printStackTrace();
				}
			}
			deleteTempPics();
		}
	}

	//loads patient data from the text file
	public static void loadData()
	{
		Scanner scanner;
		myPatients = new PriorityQueue<Patient>();
		try
		{
			scanner = new Scanner(new File(myFile));
			while (scanner.hasNext())
			{
				String firstName = scanner.nextLine();
				String lastName = scanner.nextLine();
				String sex = scanner.nextLine();
				String procedure = scanner.nextLine();
				String color = scanner.nextLine();
				String bonding = scanner.nextLine();
				String material = scanner.nextLine();
				String teethToProcedure = (scanner.nextLine());
				int day = Integer.parseInt(scanner.nextLine());
				int month = Integer.parseInt(scanner.nextLine());
				int year = Integer.parseInt(scanner.nextLine());
				String notes = fromHexString(scanner.nextLine());

				Patient pat = new Patient(firstName,lastName,sex,procedure,

teethToProcedure,color,material,bonding,day,month,year, notes);
```

```
                        myPatients.add(pat);
                    }
                    scanner.close();
                }
                catch (Exception e)
                {
                    System.out.println("Error: " + e.getMessage());
                }
        }

        //returns list of patients from the database in the form of an ArrayList
        public static ArrayList<Patient> provideList()
        {
            ArrayList<Patient> patientList = new ArrayList<Patient>();
            PriorityQueue<Patient> patientQueue = new PriorityQueue<Patient>(myPatients);
            while (!patientQueue.isEmpty())
                patientList.add(patientQueue.remove());
            return patientList;
        }

        //adds patient data to the database
        public static void addData(Patient pat)
        {
            myPatients.add(pat);
        }

        //delete patient from the database
        public static void deleteData(Patient pat)
        {
            myPatients.remove(pat);

            //removes the photo of patient whether it has been saved or not
            for (String extension: filter.getExtensions())
            {
                File picture1 = new File("Pictures/" + pat.getLastName() + pat.getFirstName() +
pat.getMonth() + pat.getDay() + pat.getYear() + "." + extension);
                File picture2 = new File("TempPics/" + pat.getLastName() + pat.getFirstName()
+ pat.getMonth() + pat.getDay() + pat.getYear() + "." + extension);
                picture1.delete();
                picture2.delete();
            }

        }
```

```
//edits patient data by removing original patient and add the new patient
public static void editData(Patient original, Patient editted)
{
	deleteData(original);
	addData(editted);
}

//delete pictures that have not been saved
public static void deleteTempPics()
{
	File dir = new File("TempPics/");
	File[] tempPics = dir.listFiles();
	if (tempPics != null)
	{
		for (File pic: tempPics)
		{
			pic.delete();
		}
	}
}

//resets contents of the database
public static void reset()
{
	myPatients = new PriorityQueue<Patient>();
}

//debug method to check contents of the priority queue database as the program is running
public static void printList()
{
	loadData();
	Iterator<Patient> iter = myPatients.iterator();
	while (iter.hasNext())
	{
		System.out.println(iter.next());
	}
}

//methods to convert between hex and string (utf-8)
//source: https://stackoverflow.com/questions/923863/converting-a-string-to-hexadecimal-in-java

public static String toHexString(String text) {
   StringBuilder str = new StringBuilder();
   byte[] ba = text.getBytes(StandardCharsets.UTF_8);
```

```
  for(int i = 0; i < ba.length; i++)
     str.append(String.format("%02x", ba[i]));
  if (str.length() == 0)
        return "";
  else
        return str.toString();
 }

 public static String fromHexString(String hex) {
        if (hex.length() == 0)
              return "";
  StringBuilder str = new StringBuilder();
  for (int i = 0; i < hex.length(); i+=2) {
   str.append((char) Integer.parseInt(hex.substring(i, i + 2), 16));
  }
  return str.toString();
 }

 public static BufferedImage loadPhoto(Patient pat)
        {
              BufferedImage bufferedImage = null;

              try {
                    bufferedImage =
ImageIO.read(Database.class.getResourceAsStream("Default.png"));
              } catch (IOException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
              }

              File picFile = findImageFile(pat);
              if (picFile != null)
              {
                    try {
                          bufferedImage = ImageIO.read(picFile);
                    } catch (IOException e) {
                          // TODO Auto-generated catch block
                          e.printStackTrace();
                    }
              }
              return bufferedImage;
        }

 public static File findImageFile(Patient pat)
```

```
        {
                String header =  pat.getLastName() + pat.getFirstName() +
                                        pat.getMonth() + pat.getDay() + pat.getYear();
                for (String possibleFormat: filter.getExtensions())
                        {
                                File savedPic = new File("Pictures/" + header + "." + possibleFormat);
                                File tempPic = new File("TempPics/" + header + "." + possibleFormat);

                                if (savedPic.exists())
                                {
                                        return savedPic;
                                }
                                else if (tempPic.exists())
                                {
                                        return tempPic;
                                }
                                else
                                        System.out.println(possibleFormat + " does not exist");
                        }
                return null;
        }

}
```

Patient Class

```
public class Patient implements Comparable
{
        //array constants of the valid types for each data category
        public static final String[] CATEGORIES = new String[] { "Last Name", "First Name", "D.O.B",
"Sex","Procedure","Material","Bonding","Color" };
        public static final String[] SEX = new String[] {"-Please Select-", "Male", "Female", "Other"};
        public static final String[] PROCEDURES = new String[] { "-Please Select-", "Crown", "Filling",
"Other"};
        public static final String[] MATERIALS = new String[] { "-Please Select-", "Flowable
Composite", "Packable Composite", "Zirconia", "EMAX", "Other"};
        public static final String[] BONDINGS = new String[] { "-Please Select-", "Adhesive", "Cement"
, "Other"};
        public static final String[] COLORS = new String[] { "-Please Select-", "B1", "A1", "A2", "A3",
"A3.5", "BL", "Other"};


        //basic info
        private String myFirstName;
```

```
	private String myLastName;
	private String mySex;

	//DOB info
	private int myDay;
	private int myMonth;
	private int myYear;

	//procedure info
	private String myProcedure;
	private String myTeethToProcedure;
	private String myColor;
	private String myMaterial;
	private String myBonding;

	//other
	private String myNotes;

	public Patient()
	{
		//gives essentially NULL values to each data value
		myFirstName = "First Name";
		myLastName = "Last Name";
		mySex = "-Please Select-";
		myProcedure = "-Please Select-";
		myTeethToProcedure =
"00000000000000000000000000000000000000000000000000000000";
		myColor = "-Please Select-";
		myMaterial = "-Please Select-";
		myBonding = "-Please Select-";
		myDay = 1;
		myMonth = 1;
		myYear= 2001;
		myNotes = "";
	}

	public Patient(String firstName, String lastName, String sex, String procedure, String
teethToProcedure,
			String color, String material, String bonding, int day, int month, int year, String
notes)
	{
		myFirstName = firstName;
		myLastName = lastName;
		mySex = sex;
```

```
        myProcedure = procedure;
        myTeethToProcedure = teethToProcedure;
        myColor = color;
        myMaterial = material;
        myBonding = bonding;
        myDay = day;
        myMonth = month;
        myYear = year;
        myNotes = notes;
}

////////////////////    getter methods  ////////////////////////////////
public String getFirstName() {return myFirstName;}

public String getLastName() {return myLastName;}

public String getSex()  {return mySex;}

public String getProcedure() {return myProcedure;}

public String getColor() {return myColor;}

public String getMaterial() {return myMaterial;}

public String getBonding() {return myBonding;}

public String getTeethToProcedure() {return myTeethToProcedure;}

public int getDay() {return myDay;}

public int getMonth() {return myMonth;}

public int getYear() {return myYear;}

public String getNotes() {return myNotes;}

////////////////////////////////////////////////////////////////////////////

//////////////////////////       setter methods   ////////////////////////////
public void setFirstName(String firstName) {myFirstName = firstName;}

public void setLastName(String lastName) {myLastName = lastName;}

public void setSex(String sex) {mySex = sex;}
```

```
	public void setProcedure(String procedure) {myProcedure = procedure;}

	public void setColor(String color) {myColor = color;}

	public void setMaterial(String material) {myMaterial = material;}

	public void setBonding(String bonding) {myBonding = bonding;}

	public void setTeethToProcedure(String teethToProcedure) {myTeethToProcedure =
teethToProcedure;}

	public void setDay(int day) {myDay = day;}

	public void setMonth(int month) {myMonth = month;}

	public void setYear(int year) {myYear = year;}

	public void setNotes(String notes) {myNotes = notes;}

	////////////////////////////////////////////////////////////////////////////

	//returns patient data in the form of a string (mostly for saving to text file)
	public String toString()
	{
		return myFirstName + "\n" + myLastName + "\n" + mySex + "\n" + myProcedure + "\n"
+
				myColor + "\n" + myBonding + "\n" + myMaterial + "\n" +
myTeethToProcedure + "\n"
				+ myDay + "\n" + myMonth + "\n" + myYear + "\n" +
Database.toHexString(myNotes) + "\n";
	}

	//compares patients based on their full names alphabetically
	public int compareTo(Object other)
	{
		if (this.myLastName.compareTo(((Patient)other).getLastName()) != 0)
			return this.myLastName.compareTo(((Patient)other).getLastName());
		else
			return this.myFirstName.compareTo(((Patient)other).getFirstName());
	}

	//ordered based on sorting priority for the main menu's sort features
	public String[] getPatientInfo()
```

```
	{
		return new String[]{myLastName, myFirstName, myMonth + "/" + myDay + "/" +
myYear, mySex, myProcedure,
					myMaterial, myBonding, myColor,  myTeethToProcedure,
					""+myDay, ""+myMonth, ""+myYear};
	}

}
```

Window Class

```
import java.awt.Toolkit;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Window extends JFrame {

	//default constructor for a window
	public Window()
	{
		setResizable(false);
		setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE);

setIconImage(Toolkit.getDefaultToolkit().getImage(getClass().getResource("icon.png")));
	}
}
```

AddEditPatientWindow Class

```
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.File;
import java.io.IOException;
import java.nio.file.Files;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;

import javax.imageio.ImageIO;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
```

```
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JSpinner;
import javax.swing.JTextField;
import javax.swing.SpinnerNumberModel;
import javax.swing.SwingConstants;
import javax.swing.border.EmptyBorder;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.JFileChooser;
import java.awt.Font;
import java.awt.Image;
import java.awt.Insets;
import java.awt.Color;
import java.awt.event.FocusAdapter;
import java.awt.event.FocusEvent;
import java.awt.image.BufferedImage;

import javax.swing.JTextArea;
import javax.swing.JScrollPane;

public class AddEditPatientWindow extends Window {

	private JPanel contentPane;

	//text fields
	private JTextField textFieldFirstName;
	private JTextField textFieldLastName;

	//spinners
	private JSpinner spinnerMonth;
	private JSpinner spinnerDay;
	private JSpinner spinnerYear;

	//combo boxes
	private JComboBox<String> comboBoxProcedure;
	private JComboBox<String> comboBoxSex;
	private JComboBox<String> comboBoxMaterial;
	private JComboBox<String> comboBoxBonding;
	private JComboBox<String> comboBoxColor;

	//buttons
	private JButton btnSelectTeeth;
	private JButton btnSelectPhoto;
	private JButton btnConfirm;
```

```
private JButton btnCancel;

//labels
private JLabel lblTitle;
private JLabel lblName;
private JLabel lblSex;
private JLabel lblDOB;
private JLabel lblProcedure;
private JLabel lblServiceTeeth;
private JLabel lblMaterial;
private JLabel lblBonding;
private JLabel lblColor;
private JLabel lblPhoto;
private JLabel lblNotesTitle;
private JLabel lblPhotoImage;

//extra windows
private TeethSelectionWindow teethWindow;
private JFileChooser photoChooser;

//data
private Patient myPatient;
private boolean isAdding;
private File fileToDelete;

//backgrounds
private JTextField infoBackground;
private JTextArea textAreaNotes;
private JTextArea titleBackground;
private JTextField photoBackground;
private JTextArea photoTitleBackground;
private JTextArea notesBackground;
private JScrollPane scrollPane;


/**
 * Create the frame.
 */
public AddEditPatientWindow(Patient pat) {

	myPatient = pat;

	//determine if client is adding or editing
	isAdding = false;
```

```
		if (myPatient.getSex().equals("-Please Select-"))
			isAdding = true;

		fileToDelete = null;

		//creates window to select teeth
		teethWindow = new TeethSelectionWindow(myPatient.getTeethToProcedure());

		//creates window to choose patient photo
		photoChooser = new JFileChooser();
		photoChooser.addChoosableFileFilter(Database.filter);
		photoChooser.setAcceptAllFileFilterUsed(false);

		File imageFile = Database.findImageFile(pat);

		if (imageFile != null)
		{
			File copyDir = new File("Copy/");
			copyDir.mkdirs();
			try {
				Files.copy(Paths.get(imageFile.getPath()), Paths.get("Copy/" +
imageFile.getName()), StandardCopyOption.REPLACE_EXISTING);
				fileToDelete = new File("Copy/" + imageFile.getName());
				System.out.println(fileToDelete.getPath());
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		}
		else
			System.out.println("no pic");

		photoChooser.setSelectedFile(fileToDelete);


		setGUI();
	}

	//initializes the GUI elements and features
	public void setGUI()
	{
		setBounds(100, 100, 776, 434);
		contentPane = new JPanel();
		contentPane.setBackground(new Color(240, 255, 240));
```

```
contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

setContentPane(contentPane);
contentPane.setLayout(null);

setLabels();
setTextFields();
setComboBoxes();
setButtons();
setSpinners();

infoBackground = new JTextField();
infoBackground.setEditable(false);
infoBackground.setEnabled(false);
infoBackground.setBackground(new Color(245, 245, 220));
infoBackground.setBounds(10, 56, 349, 295);
contentPane.add(infoBackground);
infoBackground.setColumns(10);

titleBackground = new JTextArea();
titleBackground.setEnabled(false);
titleBackground.setEditable(false);
titleBackground.setBackground(new Color(245, 245, 220));
titleBackground.setBounds(10, 11, 741, 34);
contentPane.add(titleBackground);

lblNotesTitle = new JLabel("Notes:");
lblNotesTitle.setForeground(new Color(0, 0, 0));
lblNotesTitle.setHorizontalAlignment(SwingConstants.CENTER);
lblNotesTitle.setBackground(new Color(245, 245, 220));
lblNotesTitle.setBounds(367, 58, 209, 20);
contentPane.add(lblNotesTitle);

notesBackground = new JTextArea();
notesBackground.setForeground(new Color(0, 0, 0));
notesBackground.setEditable(false);
notesBackground.setEnabled(false);
notesBackground.setBackground(new Color(245, 245, 220));
notesBackground.setBounds(369, 56, 209, 22);
contentPane.add(notesBackground);

scrollPane = new JScrollPane();
scrollPane.setBounds(369, 83, 209, 268);
contentPane.add(scrollPane);
```

```
		textAreaNotes = new JTextArea(myPatient.getNotes());
		scrollPane.setViewportView(textAreaNotes);
		textAreaNotes.setLineWrap(true);
		textAreaNotes.setMargin(new Insets(10,10,10,10));

		photoTitleBackground = new JTextArea();
		photoTitleBackground.setForeground(Color.BLACK);
		photoTitleBackground.setEnabled(false);
		photoTitleBackground.setEditable(false);
		photoTitleBackground.setBackground(new Color(245, 245, 220));
		photoTitleBackground.setBounds(586, 56, 165, 22);
		contentPane.add(photoTitleBackground);

		lblPhotoImage = new JLabel("");
		lblPhotoImage.setHorizontalAlignment(SwingConstants.CENTER);
		lblPhotoImage.setBackground(new Color(245, 245, 220));
		lblPhotoImage.setBounds(594, 95, 150, 207);

		BufferedImage bufferedImage = Database.loadPhoto(myPatient);
		Image image = bufferedImage.getScaledInstance(140, lblPhotoImage.getHeight(),
Image.SCALE_DEFAULT);
		lblPhotoImage.setIcon(new ImageIcon(image));

		contentPane.add(lblPhotoImage);

		photoBackground = new JTextField();
		photoBackground.setEnabled(false);
		photoBackground.setEditable(false);
		photoBackground.setColumns(10);
		photoBackground.setBackground(new Color(245, 245, 220));
		photoBackground.setBounds(586, 83, 165, 268);
		contentPane.add(photoBackground);
	}

	//sets up all the labels
	private void setLabels()
	{
		lblTitle = new JLabel();
		lblTitle.setForeground(new Color(0, 128, 0));
		lblTitle.setFont(new Font("Perpetua", Font.PLAIN, 27));
		if (isAdding)
		{
			lblTitle.setText("Add Patient");
```

```
            this.setTitle("Add Patient");
}
else
{
            lblTitle.setText("Edit Patient");
            this.setTitle("Edit Patient");
}

lblTitle.setHorizontalAlignment(SwingConstants.CENTER);
lblTitle.setBounds(10, 11, 741, 46);
contentPane.add(lblTitle);

lblName = new JLabel("Name: ");
lblName.setHorizontalAlignment(SwingConstants.LEFT);
lblName.setBounds(25, 68, 97, 26);
contentPane.add(lblName);

lblSex = new JLabel("Sex:");
lblSex.setHorizontalAlignment(SwingConstants.LEFT);
lblSex.setBounds(25, 98, 41, 26);
contentPane.add(lblSex);

lblDOB = new JLabel("Date of Birth: ");
lblDOB.setBounds(25, 138, 106, 14);
contentPane.add(lblDOB);

lblProcedure = new JLabel("Procedure: ");
lblProcedure.setBounds(25, 177, 97, 14);
contentPane.add(lblProcedure);

lblServiceTeeth = new JLabel("Teeth to Service: ");
lblServiceTeeth.setBounds(25, 214, 117, 14);
contentPane.add(lblServiceTeeth);

lblMaterial = new JLabel("Material: ");
lblMaterial.setBounds(25, 252, 106, 14);
contentPane.add(lblMaterial);

lblBonding = new JLabel("Bonding: ");
lblBonding.setBounds(25, 288, 106, 14);
contentPane.add(lblBonding);

lblColor = new JLabel("Color: ");
lblColor.setBounds(25, 321, 46, 14);
```

```
		contentPane.add(lblColor);

		lblPhoto = new JLabel("Photo:");
		lblPhoto.setHorizontalAlignment(SwingConstants.CENTER);
		lblPhoto.setBounds(586, 61, 165, 14);
		contentPane.add(lblPhoto);
}

//sets up all the text fields
private void setTextFields()
{
		textFieldFirstName = new JTextField(myPatient.getFirstName());
		textFieldFirstName.addFocusListener(new FocusAdapter() {
			@Override
			public void focusGained(FocusEvent e)
			{
				if (textFieldFirstName.getText().equals("First Name"))
					textFieldFirstName.setText("");
			}
			@Override
			public void focusLost(FocusEvent e)
			{
				if (textFieldFirstName.getText().equals(""))
					textFieldFirstName.setText(myPatient.getFirstName());
			}
		});
		textFieldFirstName.setBounds(151, 71, 86, 20);
		contentPane.add(textFieldFirstName);
		textFieldFirstName.setColumns(10);

		textFieldLastName = new JTextField(myPatient.getLastName());
		textFieldLastName.addFocusListener(new FocusAdapter() {
			@Override
			public void focusGained(FocusEvent e)
			{
				if (textFieldLastName.getText().equals("Last Name"))
					textFieldLastName.setText("");
			}
			@Override
			public void focusLost(FocusEvent e)
			{
				if (textFieldLastName.getText().equals(""))
					textFieldLastName.setText(myPatient.getLastName());
			}
```

```
        });
        textFieldLastName.setBounds(247, 71, 101, 20);
        contentPane.add(textFieldLastName);
        textFieldLastName.setColumns(10);
}

//sets up all the combo boxes
private void setComboBoxes()
{
        comboBoxProcedure = new JComboBox(Patient.PROCEDURES);
        comboBoxProcedure.setSelectedItem(myPatient.getProcedure());
        comboBoxProcedure.setBounds(152, 173, 196, 22);
        contentPane.add(comboBoxProcedure);

        comboBoxSex = new JComboBox(Patient.SEX);
        comboBoxSex.setSelectedItem(myPatient.getSex());
        comboBoxSex.setBounds(152, 102, 196, 22);
        contentPane.add(comboBoxSex);

        comboBoxMaterial = new JComboBox(Patient.MATERIALS);
        comboBoxMaterial.setSelectedItem(myPatient.getMaterial());
        comboBoxMaterial.setBounds(152, 248, 196, 22);
        contentPane.add(comboBoxMaterial);

        comboBoxBonding = new JComboBox(Patient.BONDINGS);
        comboBoxBonding.setSelectedItem(myPatient.getBonding());
        comboBoxBonding.setBounds(152, 284, 196, 22);
        contentPane.add(comboBoxBonding);

        comboBoxColor = new JComboBox(Patient.COLORS);
        comboBoxColor.setSelectedItem(myPatient.getColor());
        comboBoxColor.setBounds(152, 317, 196, 22);
        contentPane.add(comboBoxColor);
}

//sets up all the buttons
private void setButtons()
{
        btnSelectTeeth = new JButton("Select Teeth");
        btnSelectTeeth.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e) {
                        teethWindow.setVisible(true);
                }
        });
```

```
		btnSelectTeeth.setBounds(152, 210, 196, 23);
		contentPane.add(btnSelectTeeth);

		btnSelectPhoto = new JButton("Select Photo");
		btnSelectPhoto.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				int selected = photoChooser.showSaveDialog(null);
				if (selected == JFileChooser.APPROVE_OPTION)
				{
					File selectedPhoto = photoChooser.getSelectedFile();
			try {
					Image image =
(ImageIO.read(selectedPhoto)).getScaledInstance(lblPhotoImage.getWidth(), lblPhotoImage.getHeight(),
Image.SCALE_DEFAULT);
					lblPhotoImage.setIcon(new ImageIcon(image));
				} catch (IOException exception) {
					// TODO Auto-generated catch block
					exception.printStackTrace();
				}
				}
		else
		  System.out.println("the user cancelled the operation");
			}
		});
		btnSelectPhoto.setBounds(595, 320, 150, 23);
		contentPane.add(btnSelectPhoto);

		btnConfirm = new JButton("Confirm");
		btnConfirm.setBounds(247, 362, 89, 23);
		btnConfirm.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e) {
				boolean addedEditted = addPatient();
				if (addedEditted) {
					MainMenu menu = new MainMenu();
					menu.setVisible(true);
					dispose();
				}
			}
		});
		contentPane.add(btnConfirm);

		btnCancel = new JButton("Cancel");
		btnCancel.setBounds(439, 362, 89, 23);
		btnCancel.addActionListener(new ActionListener() {
```

```
                public void actionPerformed(ActionEvent e) {
                        quit();
                }
        });
        contentPane.add(btnCancel);
}

//sets up all the spinners
private void setSpinners()
{
        spinnerMonth = new JSpinner();
        spinnerMonth.setModel(new SpinnerNumberModel(1, 1, 12, 1));
        spinnerMonth.setValue(myPatient.getMonth());
        spinnerMonth.setBounds(151, 135, 41, 20);
        contentPane.add(spinnerMonth);

        spinnerDay = new JSpinner();
        spinnerDay.setModel(new SpinnerNumberModel(1, 1, 31, 1));
        spinnerDay.setValue(myPatient.getDay());
        spinnerDay.setBounds(213, 135, 41, 20);
        contentPane.add(spinnerDay);

        spinnerYear = new JSpinner();
        spinnerYear.setModel(new SpinnerNumberModel(2001, 1900, 3000, 1));
        spinnerYear.setValue(myPatient.getYear());
        spinnerYear.setBounds(272, 135, 76, 20);
        contentPane.add(spinnerYear);
}

//adds (or edits) patient to database using info from input fields
private boolean addPatient() {


        //check if the teeth selection window is still open
        if (teethWindow.isVisible())
        {
                JOptionPane.showMessageDialog(getRootPane(), "      Error: Finish choosing
the teeth to procedure", "Select Teeth", JOptionPane.ERROR_MESSAGE);
                return false;
        }

        //get all data from input fields
        String firstName = capitalize(textFieldFirstName.getText());
        String lastName = capitalize(textFieldLastName.getText());
```

```
		String procedure = comboBoxProcedure.getSelectedItem().toString();
		String sex = comboBoxSex.getSelectedItem().toString();
		String color = comboBoxColor.getSelectedItem().toString();
		String adhesive = comboBoxBonding.getSelectedItem().toString();
		String compositeMaterial = comboBoxMaterial.getSelectedItem().toString();
		String teethSelected = teethWindow.getSelectedTeeth();
		int day = (int) spinnerDay.getValue();
		int month = (int) spinnerMonth.getValue();
		int year = (int) spinnerYear.getValue();
		String notes = textAreaNotes.getText();

		//check if all required data has been added
		if (firstName.equals("First Name") || lastName.equals("Last Name")
				|| comboBoxSex.getSelectedIndex() == 0 ||
comboBoxProcedure.getSelectedIndex() == 0
				|| comboBoxColor.getSelectedIndex() == 0 ||
comboBoxBonding.getSelectedIndex() == 0
				|| comboBoxMaterial.getSelectedIndex() == 0)
		{
			JOptionPane.showMessageDialog(getRootPane(),
					"	Error: Empty or invalid fields", "Invalid fields",
JOptionPane.ERROR_MESSAGE);
			return false;
		}

		//create new patient using the data
		Patient pat = new Patient(firstName, lastName, sex, procedure, teethSelected, color,
compositeMaterial,
					adhesive, day, month, year, notes);

		//adds patient if adding; otherwise edit patient
		if (isAdding)
			Database.addData(pat);
		else
			Database.editData(myPatient, pat);

		//add image for user
		if (photoChooser.getSelectedFile() != null)
			addImage(pat);

		if (fileToDelete != null)
		{
			fileToDelete.delete();
			File copyDir = new File("Copy/");
```

```
                copyDir.delete();
            }

            return true;
        }

        //adds image for patient given selected image file and patient name & DOB
        private void addImage(Patient pat)
        {
            String header = pat.getLastName() + pat.getFirstName() + pat.getMonth() + pat.getDay()
+ pat.getYear();
            String path = photoChooser.getSelectedFile().getPath();
            String fileType = path.substring(path.indexOf("."));
            String newPath = "TempPics/" + header + fileType;
            System.out.println(newPath);

            try {
                Files.copy(Paths.get(path), Paths.get(newPath),
StandardCopyOption.REPLACE_EXISTING);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }

        //confirms if user wants to go back to main menu
        public void quit() {
            int dialogResult = JOptionPane.showConfirmDialog(getRootPane(),
                    "      Are you sure you cancel?\n(Any modifications will be erased.)",
"Cancel Confirmation",
                    JOptionPane.ERROR_MESSAGE);
            if (dialogResult == 1) {
                return;
            }
            MainMenu menu = new MainMenu();
            menu.setVisible(true);
            dispose();
        }

        //makes a string have first letter capitalize (proper proper noun convention)
        private String capitalize(String text)
        {
            String firstLetter = ("" + text.charAt(0)).toUpperCase();
            if (text.length() > 1)
```

```
		{
			return firstLetter + text.substring(1).toLowerCase();
		}
		else
			return firstLetter;
	}
}
```

TeethSelectionWindow Class

```
import java.awt.Component;
import java.awt.Toolkit;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JOptionPane;
import javax.swing.JTabbedPane;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JCheckBox;
import java.awt.Color;

public class TeethSelectionWindow extends Window {

	private JPanel contentPane;

	//panes
	private JTabbedPane TabsPane;

	//panel
	private JPanel PrimaryPanel;
	private JPanel PermanentPanel;

	//check boxes
	private JCheckBox chckbxA;
	private JCheckBox chckbxB;
	private JCheckBox chckbxC;
	private JCheckBox chckbxD;
	private JCheckBox chckbxE;
	private JCheckBox chckbxF;
	private JCheckBox chckbxG;
	private JCheckBox chckbxH;
```

```
private JCheckBox chckbxI;
private JCheckBox chckbxJ;
private JCheckBox chckbxK;
private JCheckBox chckbxL;
private JCheckBox chckbxM;
private JCheckBox chckbxN;
private JCheckBox chckbxO;
private JCheckBox chckbxP;
private JCheckBox chckbxQ;
private JCheckBox chckbxR;
private JCheckBox chckBoxS;
private JCheckBox chckBoxT;
private JCheckBox chckbx1;
private JCheckBox chckbx2;
private JCheckBox chckbx3;
private JCheckBox chckbx5;
private JCheckBox chckbx4;
private JCheckBox chckbx6;
private JCheckBox chckbx7;
private JCheckBox chckbx8;
private JCheckBox chckbx9;
private JCheckBox chckbx10;
private JCheckBox chckbx11;
private JCheckBox chckbx12;
private JCheckBox chckbx13;
private JCheckBox chckbx14;
private JCheckBox chckbx15;
private JCheckBox chckbx16;
private JCheckBox chckbx17;
private JCheckBox chckbx18;
private JCheckBox chckbx19;
private JCheckBox chckbx20;
private JCheckBox chckbx21;
private JCheckBox chckbx22;
private JCheckBox chckbx23;
private JCheckBox chckbx24;
private JCheckBox chckbx25;
private JCheckBox chckbx26;
private JCheckBox chckbx27;
private JCheckBox chckbx28;
private JCheckBox chckbx29;
private JCheckBox chckbx30;
private JCheckBox chckbx31;
private JCheckBox chckbx32;
```

```
//buttons
private JButton btnSave;
private JButton btnCancel;
private JButton btnReset;

//data
private char[] primaryTeeth;
private char[] permanentTeeth;
private String allTeeth;


/**
 * Create the frame.
 */
public TeethSelectionWindow(String selectTeeth) {

	allTeeth = selectTeeth;
	primaryTeeth = allTeeth.substring(0,20).toCharArray();
	permanentTeeth = allTeeth.substring(20,52).toCharArray();

	setGUI();
}

public void setGUI()
{
	setBounds(100, 100, 296, 313);
	setPanes();
	setPanels();
	setCheckBoxes();
	setButtons();
}

private void setPanes()
{
	contentPane = new JPanel();
	contentPane.setBackground(new Color(240, 255, 240));
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));

	setContentPane(contentPane);
	contentPane.setLayout(null);

	TabsPane = new JTabbedPane(JTabbedPane.TOP);
	TabsPane.setBounds(0, 11, 259, 191);
```

```
        contentPane.add(TabsPane);
}

private void setPanels()
{
        PrimaryPanel = new JPanel();
        TabsPane.addTab("Primary", null, PrimaryPanel, null);
        PrimaryPanel.setLayout(null);

        PermanentPanel = new JPanel();
        TabsPane.addTab("Permanent", null, PermanentPanel, null);
        PermanentPanel.setLayout(null);
}

private void setCheckBoxes()
{
        chckbxA = new JCheckBox("A");
        chckbxA.setBounds(6, 7, 40, 23);
        PrimaryPanel.add(chckbxA);

        chckbxB = new JCheckBox("B");
        chckbxB.setBounds(6, 33, 40, 23);
        PrimaryPanel.add(chckbxB);

        chckbxC = new JCheckBox("C");
        chckbxC.setBounds(6, 59, 40, 23);
        PrimaryPanel.add(chckbxC);

        chckbxD = new JCheckBox("D");
        chckbxD.setBounds(6, 85, 40, 23);
        PrimaryPanel.add(chckbxD);

        chckbxE = new JCheckBox("E");
        chckbxE.setBounds(48, 7, 40, 23);
        PrimaryPanel.add(chckbxE);

        chckbxF = new JCheckBox("F");
        chckbxF.setBounds(48, 33, 40, 23);
        PrimaryPanel.add(chckbxF);

        chckbxG = new JCheckBox("G");
        chckbxG.setBounds(48, 59, 40, 23);
        PrimaryPanel.add(chckbxG);
```

```
chckbxH = new JCheckBox("H");
chckbxH.setBounds(48, 85, 40, 23);
PrimaryPanel.add(chckbxH);

chckbxI = new JCheckBox("I");
chckbxI.setBounds(90, 7, 40, 23);
PrimaryPanel.add(chckbxI);

chckbxJ = new JCheckBox("J");
chckbxJ.setBounds(90, 33, 40, 23);
PrimaryPanel.add(chckbxJ);

chckbxK = new JCheckBox("K");
chckbxK.setBounds(90, 59, 40, 23);
PrimaryPanel.add(chckbxK);

chckbxL = new JCheckBox("L");
chckbxL.setBounds(90, 85, 40, 23);
PrimaryPanel.add(chckbxL);

chckbxM = new JCheckBox("M");
chckbxM.setBounds(132, 7, 40, 23);
PrimaryPanel.add(chckbxM);

chckbxN = new JCheckBox("N");
chckbxN.setBounds(132, 33, 40, 23);
PrimaryPanel.add(chckbxN);

chckbxO = new JCheckBox("O");
chckbxO.setBounds(132, 59, 40, 23);
PrimaryPanel.add(chckbxO);

chckbxP = new JCheckBox("P");
chckbxP.setBounds(132, 85, 40, 23);
PrimaryPanel.add(chckbxP);

chckbxQ = new JCheckBox("Q");
chckbxQ.setBounds(174, 7, 40, 23);
PrimaryPanel.add(chckbxQ);

chckbxR = new JCheckBox("R");
chckbxR.setBounds(174, 33, 40, 23);
PrimaryPanel.add(chckbxR);
```

```
chckBoxS = new JCheckBox("S");
chckBoxS.setBounds(174, 59, 40, 23);
PrimaryPanel.add(chckBoxS);

chckBoxT = new JCheckBox("T");
chckBoxT.setBounds(174, 85, 40, 23);
PrimaryPanel.add(chckBoxT);

chckbx1 = new JCheckBox("1");
chckbx1.setBounds(6, 7, 37, 23);
PermanentPanel.add(chckbx1);

chckbx2 = new JCheckBox("2");
chckbx2.setBounds(6, 33, 37, 23);
PermanentPanel.add(chckbx2);

chckbx3 = new JCheckBox("3");
chckbx3.setBounds(6, 59, 37, 23);
PermanentPanel.add(chckbx3);

chckbx4 = new JCheckBox("4");
chckbx4.setBounds(6, 85, 37, 23);
PermanentPanel.add(chckbx4);

chckbx5 = new JCheckBox("5");
chckbx5.setBounds(6, 111, 37, 23);
PermanentPanel.add(chckbx5);

chckbx6 = new JCheckBox("6");
chckbx6.setBounds(6, 137, 37, 23);
PermanentPanel.add(chckbx6);

chckbx7 = new JCheckBox("7");
chckbx7.setBounds(45, 7, 37, 23);
PermanentPanel.add(chckbx7);

chckbx8 = new JCheckBox("8");
chckbx8.setBounds(45, 33, 37, 23);
PermanentPanel.add(chckbx8);

chckbx9 = new JCheckBox("9");
chckbx9.setBounds(45, 59, 37, 23);
PermanentPanel.add(chckbx9);
```

```
chckbx10 = new JCheckBox("10");
chckbx10.setBounds(45, 85, 43, 23);
PermanentPanel.add(chckbx10);

chckbx11 = new JCheckBox("11");
chckbx11.setBounds(45, 111, 43, 23);
PermanentPanel.add(chckbx11);

chckbx12 = new JCheckBox("12");
chckbx12.setBounds(45, 137, 43, 23);
PermanentPanel.add(chckbx12);

chckbx13 = new JCheckBox("13");
chckbx13.setBounds(84, 7, 43, 23);
PermanentPanel.add(chckbx13);

chckbx14 = new JCheckBox("14");
chckbx14.setBounds(84, 33, 43, 23);
PermanentPanel.add(chckbx14);

chckbx15 = new JCheckBox("15");
chckbx15.setBounds(84, 59, 43, 23);
PermanentPanel.add(chckbx15);

chckbx16 = new JCheckBox("16");
chckbx16.setBounds(84, 85, 43, 23);
PermanentPanel.add(chckbx16);

chckbx17 = new JCheckBox("17");
chckbx17.setBounds(84, 111, 43, 23);
PermanentPanel.add(chckbx17);

chckbx18 = new JCheckBox("18");
chckbx18.setBounds(84, 137, 43, 23);
PermanentPanel.add(chckbx18);

chckbx19 = new JCheckBox("19");
chckbx19.setBounds(125, 7, 43, 23);
PermanentPanel.add(chckbx19);

chckbx20 = new JCheckBox("20");
chckbx20.setBounds(125, 33, 43, 23);
PermanentPanel.add(chckbx20);
```

```
chckbx21 = new JCheckBox("21");
chckbx21.setBounds(125, 59, 43, 23);
PermanentPanel.add(chckbx21);

chckbx22 = new JCheckBox("22");
chckbx22.setBounds(125, 85, 43, 23);
PermanentPanel.add(chckbx22);

chckbx23 = new JCheckBox("23");
chckbx23.setBounds(125, 111, 43, 23);
PermanentPanel.add(chckbx23);

chckbx24 = new JCheckBox("24");
chckbx24.setBounds(125, 137, 43, 23);
PermanentPanel.add(chckbx24);

chckbx25 = new JCheckBox("25");
chckbx25.setBounds(166, 7, 43, 23);
PermanentPanel.add(chckbx25);

chckbx26 = new JCheckBox("26");
chckbx26.setBounds(166, 33, 43, 23);
PermanentPanel.add(chckbx26);

chckbx27 = new JCheckBox("27");
chckbx27.setBounds(166, 59, 43, 23);
PermanentPanel.add(chckbx27);

chckbx28 = new JCheckBox("28");
chckbx28.setBounds(166, 85, 43, 23);
PermanentPanel.add(chckbx28);

chckbx29 = new JCheckBox("29");
chckbx29.setBounds(166, 111, 43, 23);
PermanentPanel.add(chckbx29);

chckbx30 = new JCheckBox("30");
chckbx30.setBounds(166, 137, 43, 23);
PermanentPanel.add(chckbx30);

chckbx31 = new JCheckBox("31");
chckbx31.setBounds(211, 7, 43, 23);
PermanentPanel.add(chckbx31);
```

```
		chckbx32 = new JCheckBox("32");
		chckbx32.setBounds(211, 33, 43, 23);
		PermanentPanel.add(chckbx32);

		loadTeeth();
}

private void setButtons()
{
		btnSave = new JButton("Save");
		btnSave.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				save();
			}
		});
		btnSave.setBounds(10, 213, 74, 23);
		contentPane.add(btnSave);

		btnCancel = new JButton("Cancel");
		btnCancel.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				quit();
			}
		});
		btnCancel.setBounds(94, 213, 74, 23);
		contentPane.add(btnCancel);

		btnReset = new JButton("Reset Selection");
		btnReset.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				reset();
			}
		});
		btnReset.setBounds(10, 247, 126, 23);
		contentPane.add(btnReset);
}

private void loadTeeth()
{
		for (Component c : PrimaryPanel.getComponents()) {
		   if (c instanceof JCheckBox) {
```

```
                    if (primaryTeeth[toAscii(((JCheckBox)c).getText())] == '1')
                        ((JCheckBox) c).setSelected(true);
                }
            }
            for (Component c : PermanentPanel.getComponents()) {
                if (c instanceof JCheckBox) {
                    if (permanentTeeth[Integer.parseInt(((JCheckBox)c).getText())-1] == '1')
                        ((JCheckBox) c).setSelected(true);
                }
            }
        }

        private void save()
        {
            for (Component c : PrimaryPanel.getComponents()) {
                if (c instanceof JCheckBox) {
                    primaryTeeth[toAscii(((JCheckBox)c).getText())] = ((JCheckBox) c).isSelected() ?
'1' : '0';
                }
            }
            for (Component c : PermanentPanel.getComponents()) {
                if (c instanceof JCheckBox) {
                    permanentTeeth[Integer.parseInt(((JCheckBox)c).getText())-1] = ((JCheckBox)
c).isSelected() ? '1' : '0';
                }
            }
            allTeeth = new String(primaryTeeth) + new String(permanentTeeth);
            setVisible(false);
        }

        public void quit()
        {
            int dialogResult = JOptionPane.showConfirmDialog(getRootPane(),
                    "      Are you sure you cancel?\n(Any modifications will be erased.)",
"Cancel Confirmation",
                    JOptionPane.ERROR_MESSAGE);
            if (dialogResult == 1) {
                return;
            }
            dispose();
        }

        private void reset()
        {
```

```
                int dialogResult = JOptionPane.showConfirmDialog(getRootPane(),
                                "      Are you sure you reset?\n       (All data will be erased.)", "Reset
Confirmation",
                                JOptionPane.ERROR_MESSAGE);
                if (dialogResult == 1) {
                        return;
                }

                for (Component c : PrimaryPanel.getComponents()) {
                   if (c instanceof JCheckBox) {
                        ((JCheckBox) c).setSelected(false);
                   }
                }
                for (Component c : PermanentPanel.getComponents()) {
                   if (c instanceof JCheckBox) {
                        ((JCheckBox) c).setSelected(false);
                   }
                }
        }

        private int toAscii(String character)
        {
                return (int)(character.charAt(0))-65;
        }

        public String getSelectedTeeth()
        {
                return allTeeth;
        }
}
```

PatientInfoWindow Class

```
import java.text.DateFormatSymbols;
import java.util.ArrayList;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.border.EmptyBorder;
import javax.swing.JLabel;
import javax.swing.SwingConstants;
import javax.swing.JButton;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
```

```
import java.io.File;
import java.io.IOException;
import java.awt.event.ActionEvent;

import javax.imageio.ImageIO;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JTextArea;
import java.awt.Toolkit;
import java.awt.Font;
import java.awt.Image;
import java.awt.Insets;
import java.awt.Color;
import javax.swing.JTextField;
import javax.swing.UIManager;
import javax.swing.JTabbedPane;
import javax.swing.DropMode;
import javax.swing.JScrollPane;
import javax.swing.JTextPane;
import javax.swing.JPopupMenu;
import java.awt.Component;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.Canvas;

public class PatientInfoWindow extends Window {

	//panes
	private JPanel contentPane;
	private JTabbedPane tabbedPane;
	private JPanel infoPanel;
	private JScrollPane scrollPaneNotes;

	//buttons
	private JButton btnEdit;
	private JButton btnBack;
	private JButton btnPrevious;
	private JButton btnNext;

	//data
	private Patient myPatient;
	private ArrayList<Patient> myPatientList;
	private int myIndex;
```

```
//labels && text areas
private JLabel lblPatientName;
private JLabel lblSexTitle;
private JLabel lblDateTitle;
private JLabel lblSex;
private JLabel lblDOB;
private JLabel lblProcedureTitle;
private JLabel lblProcedure;
private JLabel lblTeethServiceTitle;
private JLabel lblMaterialTitle;
private JLabel lblBondingTitle;
private JLabel lblColorTitle;
private JLabel lblColor;
private JLabel lblBonding;
private JLabel lblMaterial;
private JLabel lblPhoto;
private JTextArea textAreaTeethToService;
private JTextArea textAreaNotes;
private JTextField photoBackground;
private JTextField nameBackground;


/**
 * Create the frame.
 */
public PatientInfoWindow(Patient pat, int index) {
	myPatientList = Database.provideList();
	myIndex = index;


	myPatient = pat;
	setGUI();
}

//initializes the GUI and related elements
public void setGUI()
{
	setTitle("Patient Information");

	setBounds(100, 100, 514, 432);
	contentPane = new JPanel();
	contentPane.setBackground(new Color(240, 255, 240));
	contentPane.setBorder(new EmptyBorder(5, 5, 5, 5));
```

```
		setContentPane(contentPane);
		contentPane.setLayout(null);

		setPanels();
		setButtons();
		setLabels();
		setPictures();
	}

	//sets up the panels/panes
	private void setPanels()
	{
		tabbedPane = new JTabbedPane(JTabbedPane.TOP);
		tabbedPane.setBounds(10, 58, 240, 290);
		contentPane.add(tabbedPane);

		infoPanel = new JPanel();
		infoPanel.setBackground(new Color(245, 245, 220));
		tabbedPane.addTab("Information", null, infoPanel, null);
		tabbedPane.setBackgroundAt(0, new Color(245, 245, 220));
		infoPanel.setLayout(null);

		scrollPaneNotes = new JScrollPane();
		tabbedPane.addTab("Notes", null, scrollPaneNotes, null);
		tabbedPane.setBackgroundAt(1, new Color(245, 245, 220));
	}

	//sets up the labels (and also text areas)
	private void setLabels()
	{

		lblPatientName = new JLabel(myPatient.getLastName() + ", " +
myPatient.getFirstName());
		lblPatientName.setBackground(new Color(245, 245, 220));
		lblPatientName.setForeground(new Color(0, 128, 0));
		lblPatientName.setFont(new Font("Corbel Light", Font.BOLD | Font.ITALIC, 35));
		lblPatientName.setHorizontalAlignment(SwingConstants.CENTER);
		lblPatientName.setBounds(10, 17, 478, 36);
		contentPane.add(lblPatientName);

		textAreaTeethToService = new JTextArea(convertTeethSelect());
		textAreaTeethToService.setWrapStyleWord(true);
		textAreaTeethToService.setLineWrap(true);
		textAreaTeethToService.setFocusable(false);
```

```
		textAreaTeethToService.setEditable(false);
		textAreaTeethToService.setBackground(new Color(245, 245, 220));
		textAreaTeethToService.setBounds(5, 98, 203, 83);
		infoPanel.add(textAreaTeethToService);

		lblSexTitle = new JLabel("Sex:");
		lblSexTitle.setFont(new Font("Dialog", Font.BOLD, 13));
		lblSexTitle.setBounds(5, 11, 46, 14);
		infoPanel.add(lblSexTitle);

		lblSex = new JLabel(myPatient.getSex());
		lblSex.setHorizontalAlignment(SwingConstants.RIGHT);
		lblSex.setFont(new Font("Dialog", Font.PLAIN, 13));
		lblSex.setBounds(67, 7, 158, 23);
		infoPanel.add(lblSex);

		lblDateTitle = new JLabel("Date of Birth:");
		lblDateTitle.setFont(new Font("Dialog", Font.BOLD, 13));
		lblDateTitle.setBounds(5, 33, 100, 14);
		infoPanel.add(lblDateTitle);

		lblDOB = new JLabel(new DateFormatSymbols().getMonths()[myPatient.getMonth()-1]
+ " " + myPatient.getDay() + ", " + myPatient.getYear());
		lblDOB.setHorizontalAlignment(SwingConstants.RIGHT);
		lblDOB.setFont(new Font("Dialog", Font.PLAIN, 13));
		lblDOB.setBounds(70, 29, 155, 23);
		infoPanel.add(lblDOB);

		lblProcedure = new JLabel(myPatient.getProcedure());
		lblProcedure.setHorizontalAlignment(SwingConstants.RIGHT);
		lblProcedure.setFont(new Font("Dialog", Font.PLAIN, 13));
		lblProcedure.setBounds(91, 53, 134, 23);
		infoPanel.add(lblProcedure);

		lblProcedureTitle = new JLabel("Procedure:");
		lblProcedureTitle.setFont(new Font("Dialog", Font.BOLD, 13));
		lblProcedureTitle.setBounds(5, 57, 77, 14);
		infoPanel.add(lblProcedureTitle);

		lblTeethServiceTitle = new JLabel("Teeth to Service:");
		lblTeethServiceTitle.setVerticalAlignment(SwingConstants.TOP);
		lblTeethServiceTitle.setFont(new Font("Dialog", Font.BOLD, 13));
		lblTeethServiceTitle.setBounds(5, 78, 108, 23);
		infoPanel.add(lblTeethServiceTitle);
```

```
        lblMaterialTitle = new JLabel(" Material:");
        lblMaterialTitle.setFont(new Font("Dialog", Font.BOLD, 13));
        lblMaterialTitle.setBounds(5, 192, 127, 14);
        infoPanel.add(lblMaterialTitle);

        lblBondingTitle = new JLabel("Bonding:");
        lblBondingTitle.setFont(new Font("Dialog", Font.BOLD, 13));
        lblBondingTitle.setBounds(5, 217, 127, 14);
        infoPanel.add(lblBondingTitle);

        lblColorTitle = new JLabel("Color:");
        lblColorTitle.setFont(new Font("Dialog", Font.BOLD, 13));
        lblColorTitle.setBounds(5, 242, 127, 14);
        infoPanel.add(lblColorTitle);

        lblColor = new JLabel(myPatient.getColor());
        lblColor.setHorizontalAlignment(SwingConstants.RIGHT);
        lblColor.setFont(new Font("Dialog", Font.PLAIN, 13));
        lblColor.setBounds(70, 242, 155, 14);
        infoPanel.add(lblColor);

        lblBonding = new JLabel(myPatient.getBonding());
        lblBonding.setHorizontalAlignment(SwingConstants.RIGHT);
        lblBonding.setFont(new Font("Dialog", Font.PLAIN, 13));
        lblBonding.setBounds(67, 217, 158, 14);
        infoPanel.add(lblBonding);

        lblMaterial = new JLabel(myPatient.getMaterial());
        lblMaterial.setHorizontalAlignment(SwingConstants.RIGHT);
        lblMaterial.setFont(new Font("Dialog", Font.PLAIN, 13));
        lblMaterial.setBounds(67, 192, 158, 14);
        infoPanel.add(lblMaterial);

        textAreaNotes = new JTextArea(myPatient.getNotes());
        textAreaNotes.setEditable(false);
        textAreaNotes.setLineWrap(true);
        textAreaNotes.setWrapStyleWord(true);
        textAreaNotes.setMargin(new Insets(10,10,10,10));
        scrollPaneNotes.setViewportView(textAreaNotes);
}

//sets up the pictures
private void setPictures()
```

```
	{
		lblPhoto = new JLabel("");
		lblPhoto.setBackground(new Color(245, 245, 220));
		lblPhoto.setBounds(260, 58, 229, 290);
		contentPane.add(lblPhoto);
		loadPhoto();

		photoBackground = new JTextField();
		photoBackground.setBackground(new Color(245, 245, 220));
		photoBackground.setEnabled(false);
		photoBackground.setEditable(false);
		photoBackground.setBounds(259, 58, 229, 290);
		contentPane.add(photoBackground);
		photoBackground.setColumns(10);

		nameBackground = new JTextField();
		nameBackground.setEditable(false);
		nameBackground.setEnabled(false);
		nameBackground.setBackground(new Color(245, 245, 220));
		nameBackground.setBounds(11, 11, 478, 40);
		contentPane.add(nameBackground);
		nameBackground.setColumns(10);
	}

	//loads photo of the patient if it exists
	private void loadPhoto()
	{
		BufferedImage bufferedImage = Database.loadPhoto(myPatient);
		Image patientImage = bufferedImage.getScaledInstance(240, lblPhoto.getHeight(),
Image.SCALE_DEFAULT);
		lblPhoto.setIcon(new ImageIcon(patientImage));
	}

	//sets up all the buttons
	private void setButtons()
	{
		btnNext = new JButton("->");
		btnNext.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				Patient afterPat = myPatientList.get(myIndex+1);
				PatientInfoWindow afterPatWindow = new PatientInfoWindow(afterPat,
myIndex+1);
				afterPatWindow.setVisible(true);
```

```
                        dispose();
                }
        });
        btnNext.setFont(new Font("Tahoma", Font.PLAIN, 13));
        btnNext.setBounds(426, 20, 51, 23);
        contentPane.add(btnNext);


        btnPrevious = new JButton("<-");
        btnPrevious.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                        Patient beforePat = myPatientList.get(myIndex-1);
                        PatientInfoWindow beforePatWindow = new
PatientInfoWindow(beforePat, myIndex-1);
                        beforePatWindow.setVisible(true);
                        dispose();
                }
        });
        btnPrevious.setFont(new Font("Tahoma", Font.PLAIN, 13));
        btnPrevious.setBounds(20, 19, 51, 23);
        contentPane.add(btnPrevious);

        if (myIndex == 0)
                btnPrevious.setEnabled(false);
        else if (myIndex == myPatientList.size()-1)
                btnNext.setEnabled(false);

        btnEdit = new JButton("Edit");
        btnEdit.setFont(new Font("Tahoma", Font.PLAIN, 13));
        btnEdit.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e)
                {
                        AddEditPatientWindow editWindow = new
AddEditPatientWindow(myPatient);
                        editWindow.setVisible(true);
                        dispose();
                }
        });

        btnEdit.setBounds(134, 359, 89, 23);
        contentPane.add(btnEdit);

        btnBack = new JButton("Back");
```

```
		btnBack.setFont(new Font("Tahoma", Font.PLAIN, 13));
		btnBack.addActionListener(new ActionListener() {
			public void actionPerformed(ActionEvent e)
			{
				quit();
			}
		});
		btnBack.setBounds(288, 359, 89, 23);
		contentPane.add(btnBack);
	}

	//quits the window
	public void quit()
	{
		MainMenu menu = new MainMenu();
		menu.setVisible(true);
		dispose();
	}

	//converts the array of selected teeth to user-friendly text
	private String convertTeethSelect()
	{
		String result = "";
		String teethSelect = myPatient.getTeethToProcedure();
		for (int i = 0; i < 20; i++)
		{
			if (teethSelect.charAt(i) == '1')
				result += (char)('A' + i) + ", ";
		}
		for (int j = 20; j < teethSelect.length(); j++)
		{
			if (teethSelect.charAt(j) == '1')
				result += j-19 + ", ";
		}

		if (result.length() != 0)
			result = result.substring(0,result.length()-2);
		else
			result = "None";

		return result;
	}
}
```