



Stage C: The Program

(Note: Class files are not in order)

```
/**
 * Begins the program
 * @author Ali Hussain
 *
 */
public class VehicleStats
{
    /**
     * Execution of the program.
     * @param args
     * @throws IOException
     * @throws FileNotFoundException
     */
    public static void main(String[] args) throws IOException,
FileNotFoundException
    {
        new UI().setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
    }
}
```

```
/**
 * This class outputs the data.
 * @author Ali Hussain
 *
 */
class ReadRandomFile extends JFrame
{
    private VehicleregistryUI2 userInterface;
    private RandomAccessFile input;
    private JButton next, previous;
    private int recordNumber =0;

    /**
     * This method creates the window for viewing the data.
     * @pre File exists.
     * @param infile
     * @post Window for the output is created.
     */
    public ReadRandomFile(RandomAccessFile infile)
    {
        super( "Read File" );
        input=infile;
        userInterface = new VehicleregistryUI2( 12 );
        getContentPane().add( userInterface );
        readRecord(1);
        previous = userInterface.getActionButton1();
        previous.setText( "Previous" );

        previous.addActionListener(
```

```

    new ActionListener()
    {

        public void actionPerformed( ActionEvent event )
        {
            recordNumber -= 1;
            readRecord2(recordNumber);
        }

    }

);

next = userInterface.getActionButton2();
next.setText( "Next" );
next.setEnabled( true );

next.addActionListener(

    new ActionListener()
    {

        public void actionPerformed( ActionEvent event )
        {
            recordNumber += 1;
            readRecord(recordNumber);
        }

    }

);

setSize( 200, 200 );

}

/**
 * This method outputs the data to the screen after pushing the next
 button.
 * @pre File exists.
 * @param num
 * @post Data is output to the screen.
 */
public void readRecord(int num)
{

    try
    {
        RandomAccessAccountRecord record = new
RandomAccessAccountRecord();
        recordNumber = num;
        input.seek( (num-1)*RandomAccessAccountRecord.size());
        record.read(input);
    }
}

```

```

        while ( record.getTopspeed() == 0 )
        {
            recordNumber += 1;
            input.seek((recordNumber-
*RandomAccessAccountRecord.size());
            record.read(input);
        }

        String values[] =
        {
            String.valueOf( record.getTopspeed() ),
            record.getvehiclename(),
            String.valueOf( record.getengine() ),
            String.valueOf( record.getprice() ),
            String.valueOf( record.getseatingcapacity() ),
            String.valueOf( record.getlength() ),
            String.valueOf( record.getWheelbase() ),
            String.valueOf( record.getWidth() ),
            String.valueOf( record.getHeight() ),
            String.valueOf( record.getCurbweight() ),

        };

            userInterface.setFieldValues( values );

        }

    catch ( EOFException eof )
    {
        JOptionPane.showMessageDialog( this, "No more records",
            "End-of-file reached",
            JOptionPane.INFORMATION_MESSAGE );
    }

    catch ( IOException ioException )
    {
        JOptionPane.showMessageDialog( this,
            "Error Reading File", "Error",
            JOptionPane.ERROR_MESSAGE );
    }

}

/**
 * This method outputs the data to the screen after pushing the
previous button.
 * @pre File exists.
 * @param num
 * @post Data is output to the screen.
 */
public void readRecord2(int num)
{
    try
    {

```

```

        RandomAccessAccountRecord record = new
RandomAccessAccountRecord();
        recordNumber = num;
        input.seek((num-1)*RandomAccessAccountRecord.size());
        record.read(input);

        while ( record.getTopspeed() == 0 )
        {
            recordNumber -= 1;

            input.seek((recordNumber-
1)*RandomAccessAccountRecord.size());

            record.read(input);
        }

        String values[] =
        {
            String.valueOf( record.getTopspeed() ),
            record.getvehiclename(),
            String.valueOf( record.getengine() ),
            String.valueOf( record.getprice() ),
            String.valueOf( record.getseatingcapacity() ),
            String.valueOf( record.getlength() ),
            String.valueOf( record.getWheelbase() ),
            String.valueOf( record.getWidth() ),
            String.valueOf( record.getHeight() ),
            String.valueOf( record.getCurbweight() ),

        };

        userInterface.setFieldValues( values );

    }

    catch ( EOFException eof )
    {
        JOptionPane.showMessageDialog( this, "No more records",
            "Beginning-of-file reached",
            JOptionPane.INFORMATION_MESSAGE );
    }

    catch ( IOException ioException )
    {
        JOptionPane.showMessageDialog( this,
            "Error Reading File", "Error",
            JOptionPane.ERROR_MESSAGE );
    }

}
}

/**
 * This class creates the fields for the input window.
 * @author Ali Hussain

```

```

*/
class VehicleregistryUI extends JPanel
{
    protected final static String names[] =
    {
        "Top Speed",
        "Vehicle Name",
        "Engine",
        "Price",
        "Seating Capacity",
        "Length",
        "Wheel Base",
        "Width",
        "Height",
        "Curb Weight" };

    protected JLabel labels[];
    protected JTextField fields[];
    protected JButton Action1, Action2;
    protected JPanel Center, Top;

    protected int size;

    public static final int TOPSPEED=0,
        VEHICLENAME=1,
        ENGINE=2,
        PRICE=3,
        SEATINGCAPACITY=4,
        LENGTH=5,
        WHEELBASE=6,
        WIDTH=7,
        HEIGHT=8,
        CURBWEIGHT=9;

    public VehicleregistryUI(int inSize)
    {
        size = inSize;
        labels= new JLabel[size];
        fields= new JTextField[size];

        for(int count=0; count<labels.length; count++)
            labels[count]= new JLabel(names[count]);

        Center = new JPanel();
        Center.setLayout(new GridLayout(size, 2));

        for(int count=0; count<size; count++)
        {
            Center.add(labels[count]);
            Center.add(fields[count]);
        }

        Action1= new JButton();
        Action2= new JButton();

        Top= new JPanel();
    }
}

```

```

        Top.add(Action1);
        Top.add(Action2);

        setLayout(new BorderLayout());
        add(Center, BorderLayout.CENTER);
        add(Top, BorderLayout.SOUTH);

        validate();
    }

/**
 * Returns the first button.
 * @pre Button exists.
 * @return The first button.
 * @post The button is used.
 */
    public JButton getActionButton1()
    {
        return Action1;
    }

/**
 * Returns the second button.
 * @pre Button exists.
 * @return The second button.
 * @post The button is used.
 */
    public JButton getActionButton2()
    {
        return Action2;
    }

/**
 * Returns the textfields.
 * @pre Textfields exist.
 * @return The textfields.
 * @post The textfields are used.
 */
    public JTextField[] getFields()
    {
        return fields;
    }

/**
 * Clears all the fields
 * @pre The fields contain values.
 * @post The fields are empty.
 *
 */
    public void clearFields()
    {
        for(int count=0; count<size; count++)
            fields[count].setText("");
    }

/**

```

```

* Sets fields lengths.
* @pre The length must be the same as the size and greater than 0.
* @param strings
* @throws IllegalArgumentException
* @post Fields are set to the proper length.
*/
    public void setFieldValues(String strings[]) throws
IllegalArgumentException
    {
        if(strings.length != size)
            throw new IllegalArgumentException("Must have" +size+
" strings in the array record");

        for(int count=0; count<size; count++)
            fields[count].setText(strings[count]);
    }

/**
* Gets the field values.
* @pre The fields must contain values.
* @return values
* @post The field values are obtained.
*/
    public String[] getFieldValues()
    {
        String values[] = new String[size];

        for(int count=0; count<size; count++)
            values[count]=fields[count].getText().trim();

        return values;
    }
}

/**
* This class creates the field for the output values.
* @author Ali Hussain
*
*/
class VehicleregistryUI2 extends JPanel
{
    protected final static String names[] = // Categories user inputs
data
    { "Top Speed",
      "Vehicle Name",
      "Engine",
      "Price",
      "Seating Capacity",
      "Length",
      "Wheel Base",
      "Width",
      "Height",
      "Curb Weight", };

    protected JLabel labels[];
    protected JTextField fields[];

```

```

protected JButton Action1, Action2;
protected JPanel Center, Bottom;

protected int size;

/**
 *
 */
public static final int // int names
TOPSPEED=0,
VEHICLENAME=1,
ENGINE=2,
PRICE=3,
SEATINGCAPACITY=4,
LENGTH=5,
WHEELBASE=6,
WIDTH=7,
HEIGHT=8,
CURBWEIGHT=9,

/**
 * Creation of the fields.
 * @pre The inSize is greater than 0.
 * @param inSize
 * @post The fields are displayed.
 */
public VehicleregistryUI2(int inSize) // GUI Fields
{
    size = inSize;
    labels= new JLabel[size];
    fields= new JTextField[size];

    for(int count=0; count<labels.length; count++)
        labels[count]= new JLabel(names[count]);

    Center = new JPanel();
    Center.setLayout(new GridLayout(size, 2));

    for(int count=0; count<size; count++)
    {
        Center.add(labels[count]);
        Center.add(fields[count]);
    }

    Action1= new JButton();
    Action2= new JButton();

    Bottom= new JPanel();
    Bottom.add(Action1);
    Bottom.add(Action2);

}

/**
 * Returns the first button.
 * @pre Button must exist.

```

```

* @return The first button.
* @post Button is used.
*/
    public JButton getActionButton1()
    {
        return Action1;
    }

/**
* Returns the second button.
* @pre Button must exist.
* @return The second button.
* @post Button is used.
*/
    public JButton getActionButton2()
    {
        return Action2;
    }

/**
* Returns the textfields.
* @pre Textfields exist.
* @return Textfields.
* @post Textfields used.
*/
    public JTextField[] getFields()
    {
        return fields;
    }

/**
* Clears all the fields
* @pre The fields contain values.
* @post The fields are empty.
*
*/
    public void clearFields()
    {
        for(int count=0; count<size; count++)
            fields[count].setText("");
    }

/**
* Sets fields lengths.
* @pre The length must be the same as the size and greater than 0.
* @param strings
* @throws IllegalArgumentException
* @post Fields are set to the proper length.
*/
    public void setFieldValues(String strings[]) throws
IllegalArgumentException
    {
        if(strings.length != size)
            throw new IllegalArgumentException("There must be"
+size+ " strings in the array");

        for(int count=0; count<size; count++)

```

```

        fields[count].setText(strings[count]);
    }

/**
 * Returns field values.
 * @pre Fields contain values.
 * @return Field Values.
 * @post Values returned
 */

/**
 * This class allows the user to input the data.
 * @author Ali Hussain
 */
class WriteRandomFile extends JFrame
{
    private RandomAccessFile output;
    private VehicleregistryUI userInterface;
    private JButton cancel, enter;
    int vehicletotal=1;

/**
 * Creates the window for input.
 * @pre When writing, the file created must exist.
 * @param The parameter must have correct data input
 * @post Buttons created.
 */
    public WriteRandomFile(RandomAccessFile infile)
    {
        super("Add vehicles");
        output=infile;
        userInterface = new VehicleregistryUI(10);
        getContentPane().add(userInterface, BorderLayout.CENTER);

        cancel = userInterface.getActionButton1();
        cancel.setText("Cancel");
        cancel.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                setVisible(false);
            }
        });

        enter=userInterface.getActionButton2();
        enter.setText("Enter");
        enter.setEnabled(true);
        enter.addActionListener(new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                addRecord();
            }
        });
    }
}

```

```

        }
    }
);

setSize(200,300);

}

public void addRecord()
{
    int TOPSPEED=0;
    String fields[] = userInterface.getFieldValues();
    RandomAccessAccountRecord record= new
RandomAccessAccountRecord();

        try
        {

TOPSPEED=Integer.parseInt(fields[VehicleRegistryUI.TOPSPEED]);
            record.setTopSpeed(TOPSPEED);

record.setVehicleName(fields[VehicleRegistryUI.VEHICLENAME]);

record.setEngine(Integer.parseInt(fields[VehicleRegistryUI.ENGINE
]));

record.setPrice(Integer.parseInt(fields[VehicleRegistryUI.PRICE]
));

record.setSeatingCapacity(Integer.parseInt(fields[VehicleRegistry
UI.SEATINGCAPACITY]));

record.setLength(Integer.parseInt(fields[VehicleRegistryUI.LENGTH
]));

record.setWheelbase(Integer.parseInt(fields[VehicleRegistryUI.WHE
ELBASE]));

record.setWidth(Integer.parseInt(fields[VehicleRegistryUI.WIDTH]
));

record.setHeight(Integer.parseInt(fields[VehicleRegistryUI.HEIGHT
]));

record.setCurbWeight(Integer.parseInt(fields[VehicleRegistryUI.CU
RBWEIGHT]));

            output.seek((TOPSPEED-1) *
RandomAccessAccountRecord.size());
            record.write(output);
            userInterface.clearFields();
        }
        catch(ErrorInput )
        {
            JOptionPane.showMessageDialog(this, "Incorrect
input",

```

```

                                "Invalid number format",
JOptionPane.ERROR_MESSAGE);
    }
    catch(IOException iox)
    {
        JOptionPane.showMessageDialog(this, "Writing
error", "Error writing",
                                JOptionPane.ERROR_MESSAGE);
    }
}

/**
 * This class deletes previously created records.
 * @author Ali Hussain
 *
 */
class DeleteFile extends JFrame
{
    private RandomAccessFile file;
    private VehicleregistryUI userinterface;

    /**
     * This method creates the window for deleting the record.
     * @pre File exists.
     * @param deletefile
     * @post Window for deleting the record is created and user chooses a
     selection.
     */
    JTextField TOPSPEEDField =
userinterface.getFields()[VehicleregistryUI.TOPSPEED];
    TOPSPEEDField.addActionListener
    (
        new ActionListener()
        {
            public void actionPerformed(ActionEvent e)
            {
                RandomAccessAccountRecord record =
getRecord();
            }
        }
    );

    setSize(300,100);
    setVisible(true);
}

public DeleteFile(RandomAccessFile deletefile)
{
    super("Delete vehicle");
    file = deletefile;
    userinterface = new VehicleregistryUI(1);
    getContentPane().add(userinterface, BorderLayout.CENTER);
    JButton delete = userinterface.getActionButton1();
    delete.setText("Delete Record");
    delete.addActionListener

```

```

        (
            new ActionListener()
            {
                public void actionPerformed(ActionEvent
e)
                    {
                        deleteRecord(getRecord());
                        setVisible(false);
                        userinterface.clearFields();
                    }
            }
        );

        JButton cancel = userinterface.getActionButton2();
        cancel.setText("Cancel");
        cancel.addActionListener
        (
            new ActionListener()
            {
                public void actionPerformed(ActionEvent e)
                {
                    setVisible(false);
                }
            }
        );

```

```

/**
 * This method retrieves the record to be deleted.
 * @pre File and record exists.
 * @return Returns the record
 * @post Record is returned.
 */
private RandomAccessAccountRecord getRecord()
{
    RandomAccessAccountRecord record= new
RandomAccessAccountRecord();

    try
    {
        JTextField TOPSPEEDfield =
userinterface.getFields()[VehicleRegistryUI.TOPSPEED];

        int TOPSPEED =
Integer.parseInt(TOPSPEEDfield.getText());
        file.seek((TOPSPEED-
1)*RandomAccessAccountRecord.size());
        record.read(file);
        if(record.getTopspeed()==0)
            JOptionPane.showMessageDialog(this, "Invalid
Number",
                                        "Error",
                                        JOptionPane.ERROR_MESSAGE);
    }

    catch(Errorinput )
    {

```

```

        JOptionPane.showMessageDialog(this, "Top Speed does
not exist",
                                "Invalid number format",
JOptionPane.ERROR_MESSAGE);
    }

    catch(IOException iox)
    {
        JOptionPane.showMessageDialog(this, "Error reading
file",
                                "Error", JOptionPane.ERROR_MESSAGE);
    }

    return record;
}

/**
 * This method deletes the record.
 * @pre Record exists.
 * @param record
 * @post Record is deleted.
 */
public void deleteRecord(RandomAccessAccountRecord record)
{
    if(record.getTopspeed() ==0)
        return;

    catch(IOException iox)
    {
        JOptionPane.showMessageDialog(this, "Error, ocured,
unable to write to file",
                                "Error", JOptionPane.ERROR_MESSAGE);
    }
}

    try
    {
        int TOPSPEED=record.getTopspeed();
        file.seek((TOPSPEED-
1)*RandomAccessAccountRecord.size());
        record.setTopspeed(0);
        record.write(file);
    }

/**
 * This class outlines how the records should be and has the get and
set methods.
 * @author Ali Hussain
 *
 */
class Record implements Serializable
{

    private String vehiclename;
    private int engine;

```

```

    private int price;
    private int seatingcapacity;
    private int length;
    private int wheelbase;
    private int height;
    private int width;
    private int curbweight;
    private int topspeed;

/**
 * The variables are inputted and set.
 * @pre All the variables for input must be correct.
 * @param topspeed
 * @param vehiclename
 * @param engine
 * @param price
 * @param seatingcapacity
 * @param length
 * @param wheelbase
 * @param width
 * @param height
 * @param curbweight
 *
 * @post All the variables are set.
 */
    public Record( int TOPSPEED, String vehiclename,
        int engine, int price,
        int seatingcapacity, int length, int wheelbase,
        int width, int height , int curbweight )
    {

        setTopspeed(TOPSPEED); setvehiclename ( vehiclename );
        setengine ( engine );setprice ( price );
        setseatingcapacity ( seatingcapacity );
        setlength ( length ); setWheelbase ( wheelbase );
        setWidth ( width ); setHeight ( height );
        setCurbweight ( curbweight );

    }

/**
 * This method gets the Top Speed.
 * @pre Top Speed exists.
 * @return Top Speed
 * @post Top Speed returned.
 */
    public int getTopspeed()
    {
        return topspeed;
    }

/**
 * This method sets the Top Speed.
 * @pre Top Speed for input exists.
 * @param topspeed
 * @post Top Speed is set.

```

```

*/
    public void setTopspeed(int TOPSPEED)
    {
        TOPSPEED = TOPSPEED;
    }

/**
 * This method sets the Vehicle Name.
 * @pre Vehicle Name for input exists.
 * @param vehiclename
 * @post Vehicle Name is set.
 */
    public void setvehiclename( String vehiclename)
    {
        vehiclename = vehiclename;
    }

/**
 * This method gets the Vehicle Name.
 * @pre Vehicle Name exists
 * @return Vehicle Name
 * @post Vehicle Name returned.
 */
    public String getvehiclename()
    {
        return vehiclename;
    }

/**
 * This method sets the engine.
 * @pre engine for input exists.
 * @param engine
 * @post engine is set.
 */
    public void setengine( int engine )
    {
        engine = engine;
    }

/**
 * This method gets the engine.
 * @pre engine exists.
 * @return engine
 * @post engine is returned.
 */
    public int getengine()
    {
        return engine;
    }

/**
 * This method sets the price.
 * @pre price for input exists.
 * @param price
 * @post price is set.
 */
    public void setprice( int price )

```

```

    {
        price = price;
    }

/**
 * This method gets the price.
 * @pre Price must exist.
 * @return The Price
 * @post Price is set.
 */
public int getprice()
{
    return price;
}

/**
 * This method sets the seatingcapacity.
 * @pre seatingcapacity for input exists.
 * @param seatingcapacity
 * @post seatingcapacity is set.
 */
public int setseatingcapacity( int seatingcapacity )
{
    int seatingcapacity2 = seatingcapacity;
    return seatingcapacity2;
}

/**
 * This method gets the seatingcapacity.
 * @pre seatingcapacity exists.
 * @return seatingcapacity
 * @post seatingcapacity is returned.
 */
public int getseatingcapacity()
{
    return seatingcapacity;
}

/**
 * This method sets the over time price.
 * @pre Over Time price for input exists.
 * @param length
 * @post Over Time price is set.
 */
public void setlength( int length )
{
    length = length;
}

/**
 * This method gets the over time price.
 * @pre Over Time price exists.
 * @return length
 * @post Over Time price is returned.
 */
public int getlength()
{

```

```

        return length;
    }

    /**
     * This method gets the wheelbase.
     * @pre wheelbase For exists.
     * @return wheelbase
     * @post wheelbase for is returned.
     */
    public int getWheelbase()
    {
        return wheelbase;
    }

    /**
     * This method sets the wheelbase.
     * @pre wheelbase input exists.
     * @param height
     * @post wheelbase is set.
     */
    public void setHeight( int height )
    {
        height = height;
    }

    /**
     * This method gets the height.
     * @pre height exists.
     * @return height
     * @post height is returned.
     */
    public int getHeight()
    {
        return height;
    }

    /**
     * This method gets the width.
     * @pre width For exists.
     * @return width
     * @post width For is returned.
     */
    public int getWidth()
    {
        return width;
    }

    /**
     * This method sets the width.
     * @pre width for input exists.
     * @param width
     * @post width For is set.
     */
    public void setWidth(int width)
    {

```

```

        width = width;aga
    }

/**
 * This method gets the curbweight value.
 * @pre curbweight exists.
 * @return curbweight
 * @post curbweight is returned.
 */
    public int getCurbweight()
    {
        return curbweight;
    }

/**
 * This method sets the width.
 * @pre width for input exists.
 * @param curbweight
 * @post width is set.
 */
    public void setCurbweight(int curbweight)
    {
        curbweight = curbweight;
    }
}

/**
 * This method reads the file.
 * @pre File exists.
 * @param file
 * @throws IOException
 * @post File is read and variables set.
 */
    public void read( RandomAccessFile file ) throws IOException
    {
        setTopspeed(file.readInt());
        setvehiclename ( padName(file) );
        setengine ( file.readInt() );
        setprice ( file.readInt() );
        setseatingcapacity ( file.readInt() );
        setlength ( file.readInt() );
        setWheelbase ( file.readInt() );
        setWidth ( file.readInt() );
        setHeight ( file.readInt() );
        setCurbweight ( file.readInt() );
    }

/**
 * This class does the actual reading and writing of the file.
 * @author Ali Hussain
 *
 */
class RandomAccessAccountRecord extends Record
{

```

```

/**
 * This shows how the random access account record should be.
 *
 */
    public RandomAccessAccountRecord()
    {
        this( 0, "", 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.00, 0.00 );
    }

/**
 * This method obtains all the variables.
 * @pre All the variables for input must exist.
 * @param topspeed
 * @param vehiclename
 * @param engine
 * @param price
 * @param seatingcapacity
 * @param length
 * @param wheelbasefor
 * @param width
 * @param height
 * @param curbweight
 * @post All variables are read in.
 */
    public RandomAccessAccountRecord(
        int TOPSPEED,
        String vehiclename,
        int engine,
        int price,
        int seatingcapacity,
        int length,
        int wheelbasefor,
        int width,
        int height ,
        int curbweight,
    )
    {
        super( TOPSPEED, vehiclename, engine, price,
seatingcapacity, length, wheelbasefor, width,
        height , curbweight);
    }

/**
 * This method reads the strings.
 * @pre File exists.
 * @param file
 * @return String
 * @throws IOException
 * @post String is read and returned.
 */
    private String padName(RandomAccessFile file) throws IOException
    {
        char name[] = new char[30], temp;
        for(int count=0; count<name.length; count++)
        {

```

```

        temp=file.readChar();
        name[count]=temp;
    }
    return new String(name).replace('\0', ' ');
}

/**
 * This method writes to the file.
 * @pre File exists.
 * @param file
 * @throws IOException
 * @post File is written to.
 */
public void write(RandomAccessFile file) throws IOException
{
    file.writeInt(getTopspeed());
    writeName(file, getvehiclename());
    file.writeInt(getengine());
    file.writeInt(getprice());
    file.writeInt(getseatingcapacity());
    file.writeInt(getlength());
    file.writeInt(getWheelbase());
    file.writeInt(getWidth());
    file.writeInt(getHeight());
    file.writeInt(getCurbweight());
}

/**
 * This method writes the strings.
 * @pre File and string exists.
 * @param file
 * @param name
 * @throws IOException
 * @post String is written to file.
 */
private void writeName(RandomAccessFile file, String name) throws
IOException
{
    StringBuffer buffer=null;
    if(name!=null)
        buffer=new StringBuffer(name);
    else
        buffer=new StringBuffer(30);
    buffer.setLength(30);
    file.writeChars(buffer.toString());
}

/**
 * @pre The records exist.
 * @return size
 * @post Size returned.
 */
public static int size()
{
    return 112;
}

```

```

}

/**
 * This class allows the addition or modifying of information to the
file.
 * @author Ali Hussain
 *
 */
class ModifyFile extends JFrame
{
    private RandomAccessFile output;
    private VehicleregistryUI userInterface;
    private JButton cancel, enter;
    int vehicletotal=1;

/**
 * This method creates the window for modifying the data.
 * @pre File exists.
 * @param infile
 * @post Window for the modifying is created.
 */

    public void actionPerformed(ActionEvent e)
    {
        modifyRecord();
    }
}
);

setSize(400,400);

}

public ModifyFile(RandomAccessFile infile)

{
    super("Modify vehicles");

    output=infile;
    userInterface = new VehicleregistryUI(10);
    getContentPane().add(userInterface, BorderLayout.CENTER);

    cancel = userInterface.getActionButton1();
    cancel.setText("Cancel");
    cancel.addActionListener(new ActionListener()
    {
        public void actionPerformed(ActionEvent e)
        {
            setVisible(false);
        }
    }
    );
}

```

```

        enter=userInterface.getActionButton2();
        enter.setText("Enter");
        enter.setEnabled(true);
        enter.addActionListener(new ActionListener()
        {

/**
 * This method modifies the file records.
 * @pre File exists and all the input is correct.
 * @post File is updated.
 *
 */
    public void modifyRecord()
    {
        int TOPSPEED=0;
        String vehiclename;
        int engine, price, seatingcapacity, length, wheelbasefor,
width, height, curbweight;
        int engine2, price2, seatingcapacity2, length2,
wheelbasefor2, width2, height2, curbweight2;
        String fields[] = userInterface.getFieldValues();
        RandomAccessAccountRecord record= new
RandomAccessAccountRecord();
        RandomAccessAccountRecord oldrecord = new
RandomAccessAccountRecord();
        RandomAccessAccountRecord record2 = new
RandomAccessAccountRecord();

        try // Which data types to parse
        {

            TOPSPEED=Integer.parseInt(fields[VehicleregistryUI.TOPSPEED]);
            record.setTopspeed(TOPSPEED);

            record.setvehiclename(fields[VehicleregistryUI.VEHICLENAME]);

            record.setengine(Integer.parseInt(fields[VehicleregistryUI.ENGINE
]));

            record.setprice(Integer.parseInt(fields[VehicleregistryUI.PRICE]
));

            record.setseatingcapacity(Integer.parseInt(fields[Vehicleregistry
UI.SEATINGCAPACITY]));

            record.setlength(Integer.parseInt(fields[VehicleregistryUI.LENGTH
]));

            record.setWheelbase(Integer.parseInt(fields[VehicleregistryUI.WHE
ELBASE]));

            record.setWidth(Integer.parseInt(fields[VehicleregistryUI.WIDTH]
));

            record.setHeight(Integer.parseInt(fields[VehicleregistryUI.HEIGHT
]));

```

```

        record.setCurbweight(Integer.parseInt(fields[VehicleRegistryUI.CU
RBWEIGHT]));

        vehiclename = record.getvehiclename(); // Add
to record

        engine = record.getengine();
        price = record.getprice();
        seatingcapacity = record.getseatingcapacity();
        length = record.getlength();
        wheelbasefor = record.getWheelbase();
        width = record.getWidth();
        height = record.getHeight();
        curbweight = record.getCurbweight();

        output.seek((TOPSPEED-1) *
RandomAccessAccountRecord.size());
        oldrecord.read(output);
        engine2 = oldrecord.getengine();
        price2 = oldrecord.getprice();
        seatingcapacity2 =
oldrecord.getseatingcapacity();
        length2 = oldrecord.getlength();
        wheelbasefor2 = oldrecord.getWheelbase();
        width2 = oldrecord.getWidth();
        height2 = oldrecord.getHeight();
        curbweight2 = oldrecord.getCurbweight();

        record2.setTopspeed(TOPSPEED);
        record2.setvehiclename(vehiclename);
        record2.setengine(engine+engine2);
        record2.setprice(price+price2);

        record2.setseatingcapacity(seatingcapacity+seatingcapacity2);
        record2.setlength(length+length2);

        record2.setWheelbase(wheelbasefor+wheelbasefor2);
        record2.setWidth(width+width2);
        record2.setHeight(height+height2);
        record2.setCurbweight(curbweight+curbweight2);

        output.seek((TOPSPEED-1) *
RandomAccessAccountRecord.size());
        record2.write(output);
        userInterface.clearFields();
    }
    catch(Errorinput )
    {
        JOptionPane.showMessageDialog(this, "Wrong
number entered",
                                "Number input incorrect",
JOptionPane.ERROR_MESSAGE);
    }
    catch(IOException iox)
    {

```

```

        JOptionPane.showMessageDialog(this, "Writing
error", "Error writing",
        JOptionPane.ERROR_MESSAGE);
    }
}

/**
 * This class contains the main interface the user interacts with.
 * @author Ali Hussain
 *
 */
class UI extends JFrame implements ActionListener
{
    JMenuBar mainBar = new JMenuBar();

    JMenu fileMenu = new JMenu("Start"); // Opens the main menu
    JMenuItem open = new JMenuItem("Open"); // Opens an existing
file type
    JMenuItem save = new JMenuItem("Save"); // Saves a record
    JMenuItem addVehicle = new JMenuItem("Add New Vehicle"); //
Allows the addition of a new entry
    JMenuItem modifyvehicle = new JMenuItem("Edit Existing
Entries"); // Edits information from a saved record
    JMenuItem showStats = new JMenuItem("Show Vehicle
Information"); // Shows vehicle data
    JMenuItem deletevehicle = new JMenuItem("Delete Record"); //
Removes a record
    JMenuItem exit = new JMenuItem("Exit"); // Exits program
    RandomAccessFile file;

/**
 * Adds all the items to the menu bar and allows the use of them.
 * @pre Menu bar and all of its contents exist.
 * @throws IOException
 * @post Allows user interaction to the rest of the program.
 */
    public UI() throws IOException
    {
        super("VehicleStats");

        setJMenuBar( mainBar );
        open.addActionListener(this);
        open.setActionCommand("open");
        fileMenu.add(open);

        save.addActionListener(this);
        save.setActionCommand("save");
        fileMenu.add(save);

        addVehicle.addActionListener(this);
        addVehicle.setActionCommand("addVehicle");
        fileMenu.add(addVehicle);

        modifyvehicle.addActionListener(this);
        modifyvehicle.setActionCommand("modifyvehicle");

```

```

fileMenu.add(modifyvehicle);

showStats.addActionListener(this);
showStats.setActionCommand("showStats");
fileMenu.add(showStats);

deletevehicle.addActionListener(this);
deletevehicle.setActionCommand("deletevehicle");
fileMenu.add(deletevehicle);

exit.addActionListener(this);
exit.setActionCommand("exit");
fileMenu.add(exit);

fileMenu.setMnemonic('H');
mainBar.add(fileMenu);

getContentPane().setBackground(Color.WHITE);

setSize(200,200);
setResizable(false);

}

public void actionPerformed(ActionEvent e)
{
    if(e.getActionCommand().equals("open"))
    {
        JFileChooser fileChooser = new JFileChooser();
        fileChooser.setSelectionMode(JFileChooser.FILES_ONLY);
        int result = fileChooser.showOpenDialog(this);
        if(result==JFileChooser.CANCEL_OPTION)
            return;

        File fileName=fileChooser.getSelectedFile();
        if( fileName == null || fileName.getName().equals( ""
) )

            JOptionPane.showMessageDialog( null,
                "Invalid File Name", "Invalid File Name",
                JOptionPane.ERROR_MESSAGE);

        else
        {
            try
            {
                file=new RandomAccessFile(fileName,
"rw");
            }
            catch(IOException iox)
            {
                JOptionPane.showMessageDialog(this, "File
does not exist",
                "Invalid File Name",
                JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}

```

```

else if(e.getActionCommand().equals("save"))
{
    JFileChooser accountHandler = new JFileChooser();
    accountHandler.setSelectionMode(
JFileChooser.FILES_ONLY );
    int Numb3;
    int result = accountHandler.showSaveDialog( null );

    if( result == JFileChooser.CANCEL_OPTION )
        return;

    File fileName = accountHandler.getSelectedFile();

    if( fileName == null || fileName.getName().equals( ""
) )

        JOptionPane.showMessageDialog( null,
            "Invalid File Name", "InvalidFile Name",
            JOptionPane.ERROR_MESSAGE);

    else
    {
        try
        {
            file =new RandomAccessFile( fileName,
"rw" );

            RandomAccessAccountRecord blankRecord =
new RandomAccessAccountRecord();

            for( int count = 0; count < 40; count++)
                blankRecord.write(file);

            JOptionPane.showMessageDialog( null,
                "Saved " + fileName, "Status",
                JOptionPane.INFORMATION_MESSAGE );

        }

        catch ( IOException ioException )
        {
            JOptionPane.showMessageDialog( null,
                "Error processing file", "Error
processing file",
                JOptionPane.ERROR_MESSAGE );

            System.exit(1);
        }
    }
}
else if(e.getActionCommand().equals("addVehicle")) // Adds
new vehicle
{
    new WriteRandomFile(file);
}

```

```

        else if(e.getActionCommand().equals("modifyvehicle")) //
Allows editing of existing entries
    {
        new ModifyFile(file);
    }
    else if(e.getActionCommand().equals("showStats")) // Displays
information
    {
        new ReadRandomFile(file);
    }
    else if(e.getActionCommand().equals("deletevehicle")) //
Removes an entry
    {
        new DeleteFile(file);
    }
    else if(e.getActionCommand().equals("exit"))
    {
        System.exit(0);
    }
}
}

```

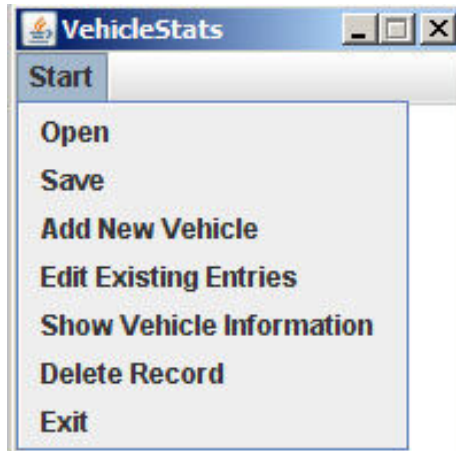
Usability

A GUI interface is used in the most basic manner which allows a simple and easy to navigate interface for users to input data.

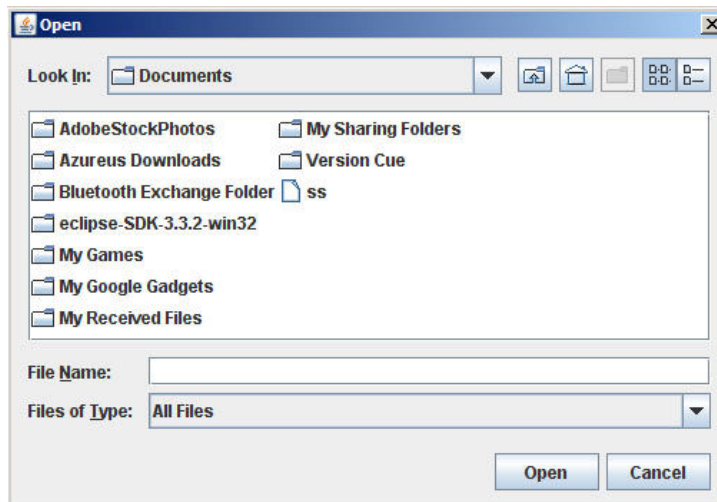


Program Start-up

The first part of the program begins with a blank screen with only a Start button.



Clicking on the "Start" button leads to a drop down menu displaying a variety of options that users may choose from.



The first button, "Open" brings up a window which allows the user to select previous files or entries created by the program.

Top Speed	<input type="text"/>
Vehicle Name	<input type="text"/>
Engine	<input type="text"/>
Price	<input type="text"/>
Seating Capacity	<input type="text"/>
Length	<input type="text"/>
Wheel Base	<input type="text"/>
Width	<input type="text"/>
Height	<input type="text"/>
Curb Weight	<input type="text"/>
<input type="button" value="Cancel"/> <input type="button" value="Enter"/>	

The main feature of the program is the “Add New Vehicle” option which brings up a window allowing users to input data as if it were a form.

Top Speed	<input type="text"/>
<input type="button" value="Delete Record"/> <input type="button" value="Cancel"/>	

If users wish to delete an entry, they may do so by going to the “Delete Record” section of the program and entering the data required. From here, the users enter the Top Speed of the vehicle they want to remove and the program searches for the entry, and promptly deletes it.

Handling Errors

There are several ways the program is able to catch potential user errors. In my program, I have the ability to input numbers and names. If a user incorrectly enters a number where letters should be or if he or she enters names where numbers should be then an error dialog pops up.

Source Code:

```
RandomAccessAccountRecord.size());
        record.write(output);
        userInterface.clearFields();
    }
    catch(Errorinput )
    {
        JOptionPane.showMessageDialog(this, "Incorrect
input",
```

This error is most likely to occur in the “Vehicle Name” section of the input form where users may accidentally provide a number instead of words.

Errors may also be created when inputting data in the modifying entry section of the program. If user input generates characters which are not suitable for the entry, then the following code issues either a “Wrong characters entered” message (for using numbers where they are not supposed to be contained” or using letters which prompts a “Number input incorrect” dialog box.

Source Code:

```
catch(Errorinput )
JOptionPane.showMessageDialog(this, "Wrong characters entered",
                             "Number input incorrect",
JOptionPane.ERROR_MESSAGE);
```

Another possible error may occur while deleting an entry. If a user inputs a record to be deleted yet the record does not exist an IO exception will occur.

Source Code:

```
        catch(IOException iox)
        {
JOptionPane.showMessageDialog(this, "Writing error", "Error writing",
                               JOptionPane.ERROR_MESSAGE);
        }
```

As well, users may incorrectly enter letters or words where numbers should be.

This will result in another error feature:

Source Code:

```
        catch(Errorinput )
        {
            JOptionPane.showMessageDialog(this, "Top Speed does not exist",
```



