AT2: Mining Simulator

REPORT

Daniel Ranieri
131600214 | DIPLOMA PROGRAMMING 2016

Contents

Introduction

This program (AT2 Mining Simulator) is a program that, on start-up, will create 20 trucks and put them into a list. This list contains trucks number with the status of the truck. These trucks don't leave this list box, but the status of each truck changes as the trucks move from queue to queue.

The program allows the user to double click a truck and put it into the transit to loading queue. The user can then click the list boxes with a truck in it to move it up the queue and out to the next queue depending where it is in real time.

There are service and return buttons that allow the user to take the first truck from the first spot in the queue to inactive or service, or from inactive status to service and back again with a double click.

There are radio buttons also that will show the selected trucks status also. The trucks also get saved to a binary file and imported on start-up.

Analysis: a statement of

What data items need to be inputted?

Data taken from the pre saved binary file is inputted into the truck array which holds the 20 trucks that are then put into the corresponding boxes.

What processes need to be performed?

- To begin the program there will be 20 trucks loaded into the main list box and they all begin with a status of 0 until changed.
- A single click on a truck will display its status on a radio button and populate the text boxes that display information about the truck.
- When the user double click the main list box on a certain truck, the truck is moved from the top of the transit to loading queue into the transit to loading list box queue at the bottom.
- When the user clicks on any of the bottom queues, the top truck in the queue will be moved from that queue into the next box and this process will continue until the truck is put into service or made inactive.
- Clicking the service and inactive button will do nothing unless a truck is selected from the top of the transit to loading queue. When clicked the truck will move to the correct location.
- To remove a truck from service, the user must double click a truck in the service list box.
- In the menu the file>new button will create a new instance of the program with all trucks set to status 0.
- In the menu the file>close button will close the program.
- In the menu the view>User Guide it will open the user guide

What output is required?

- When the form closes the truck array is written to file with the truck number and the status.
- As you change the status the radio buttons show the trucks status.
- When you open the program the trucks are created or read from a binary file and shown in the correct spots.
- Each of the list boxes show where each represented truck is and as you click will move the truck up the queue.

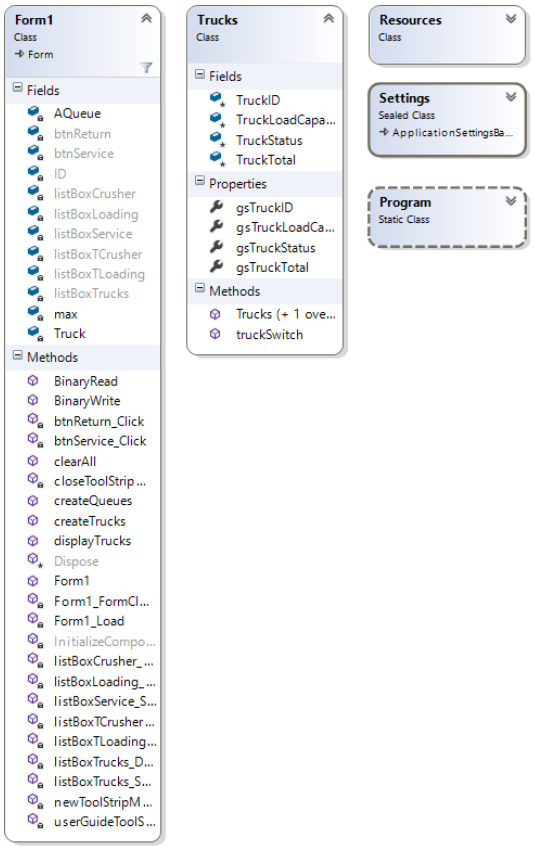
Project Plan

		14-Jun				15-Jun	16-Jun				17-Jun	18-Jun	19-Jun	20-Jun	21-Jun	22-Jun	23-Jun
Project Title:	AT2 Mining Simulator	0800	1000	1200	1400		0800	1000	1200	1400							
Start Date:	14/06/2016																
Estimated Completion Date:	23/06/2016																
TaskID	Task																
1	User Requirements																
1.1	Initial Meeting with Client																
1.2	Write draft SRS document																
1.3	Design Draft User Interface																
1.4	Email draft to Client for comment																
1.5	Make changes to draft																
1.6	Meet with client to sign contract																
2	Design User Interface																
3	Code Program																
4	Write Draft Documentation																
5	Test Application																
6	Finalise Documentation																
7	Deliver to Client for Signoff																

Resources
Project Manager
User Interface Designer
Programmer
Technical Writer
Client

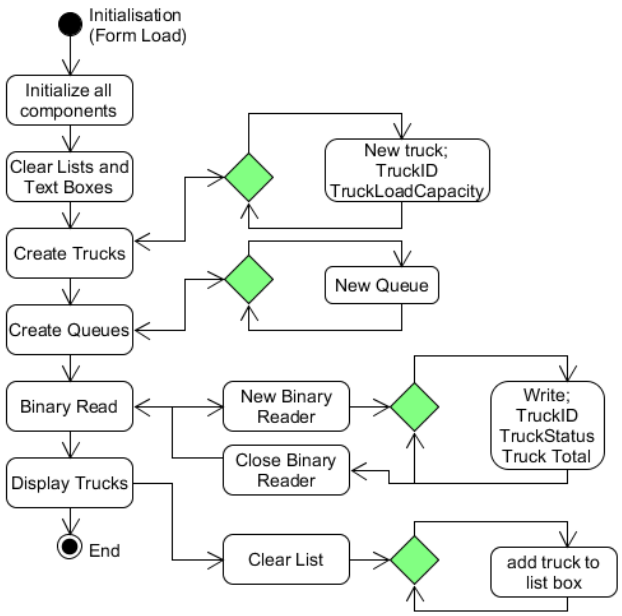
Design

Class Diagrams

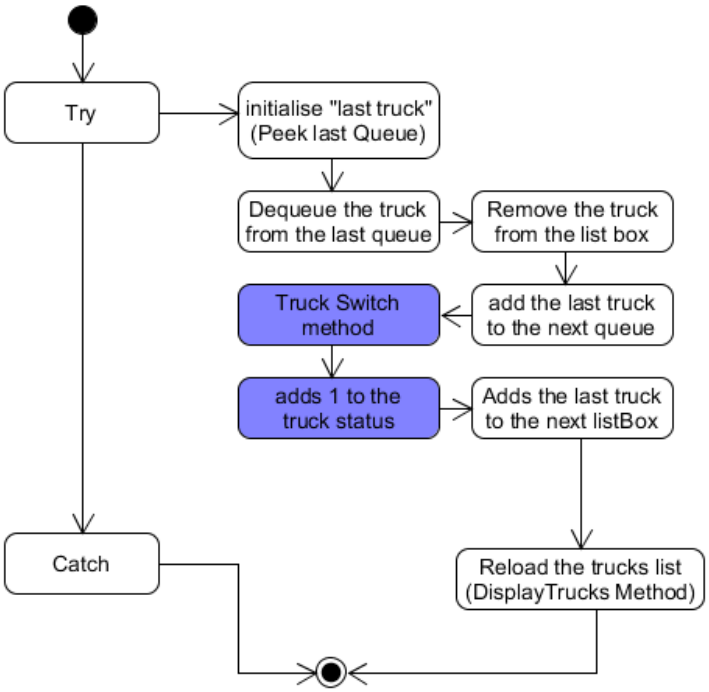


Activity Diagrams

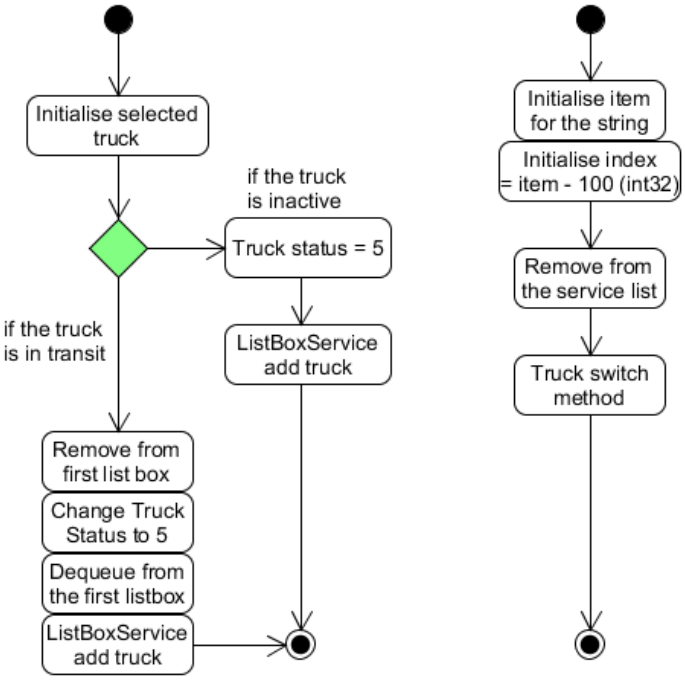
Initialisation (Display Trucks, Binary Read, Create Queues, Create Trucks, clear method):



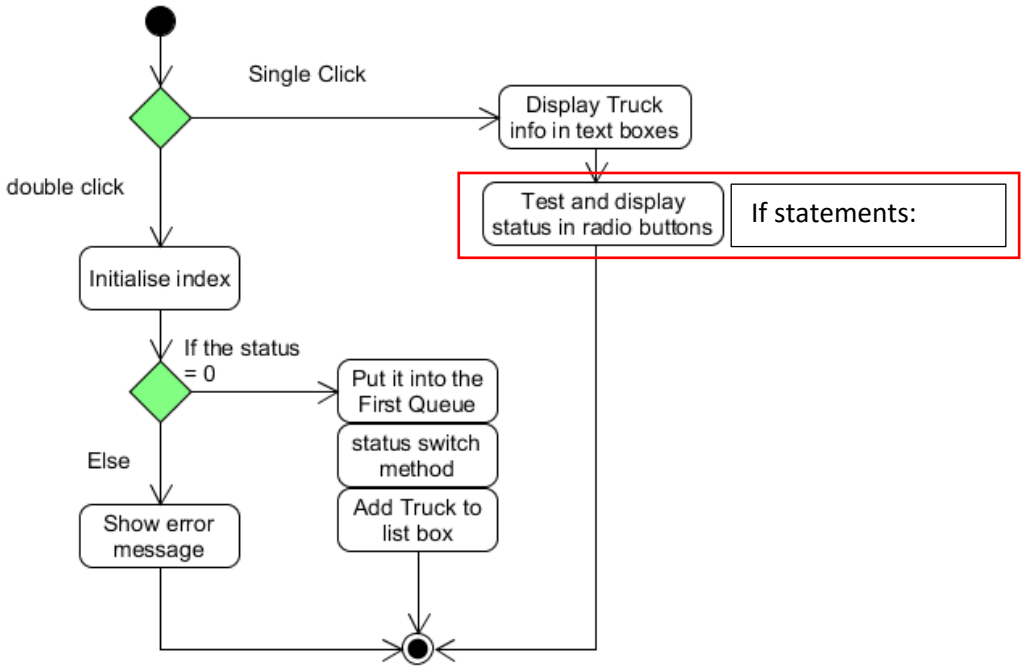
Update Trucks (Click method for the list box “Queues” & Truck switch method):



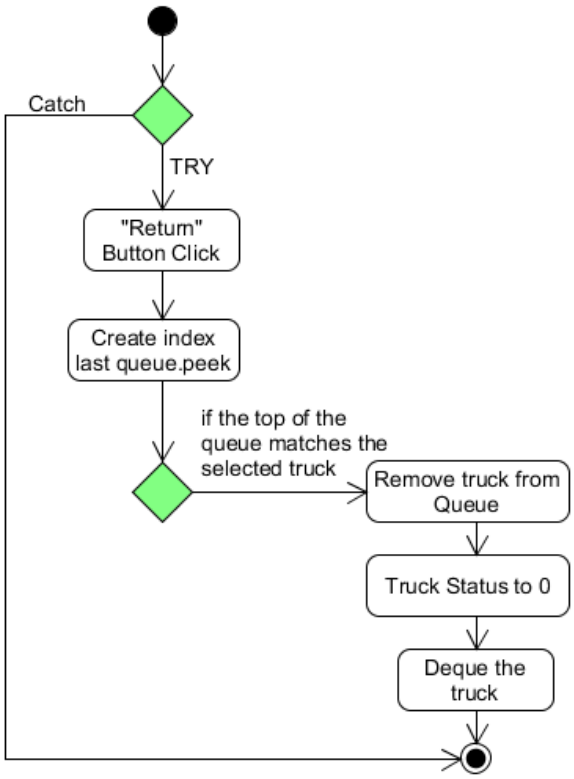
Service add & Service remove methods:



Main List box click events



Putting the trucks back into inactive:

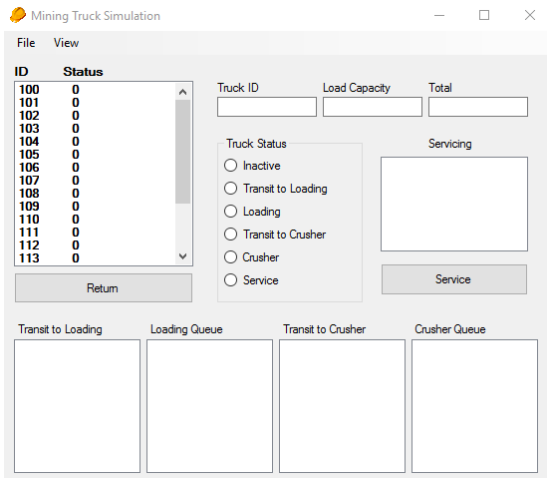


Test Data & Evidence

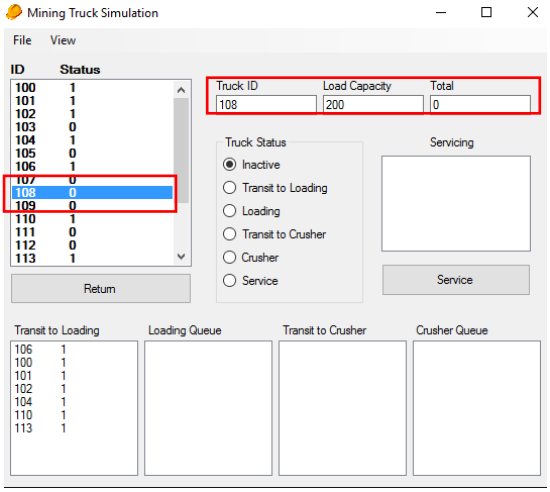
Testing the click events

Click events

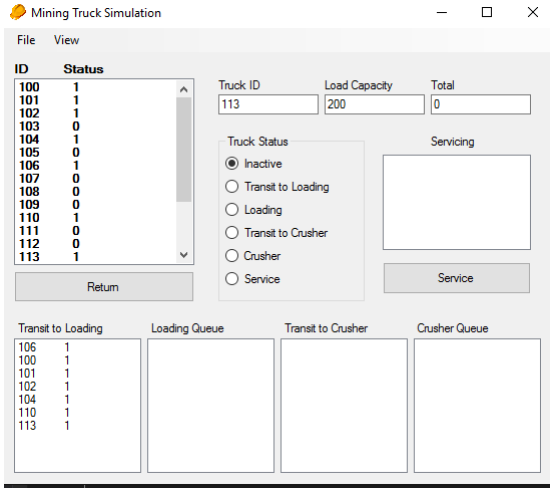
Open:



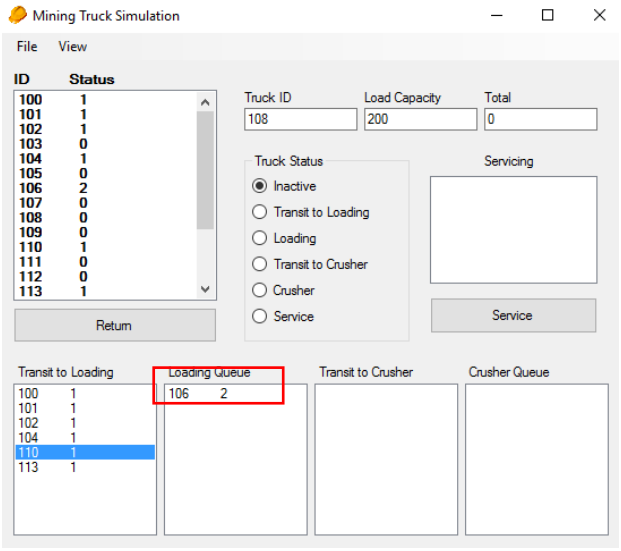
Single Click event in the main box



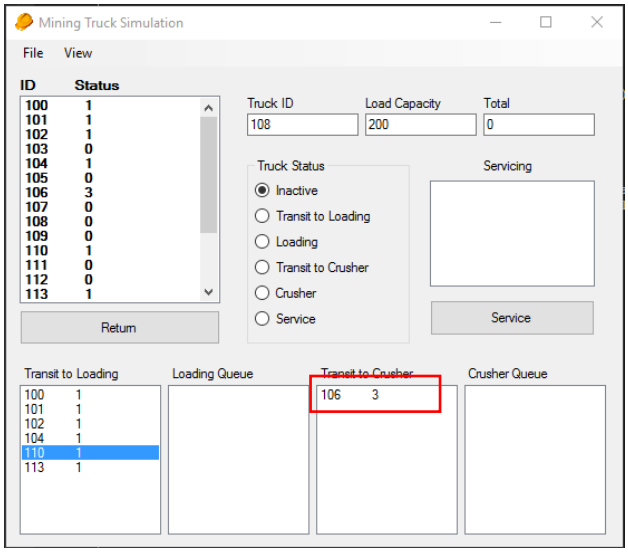
Double Click Event in the main Box



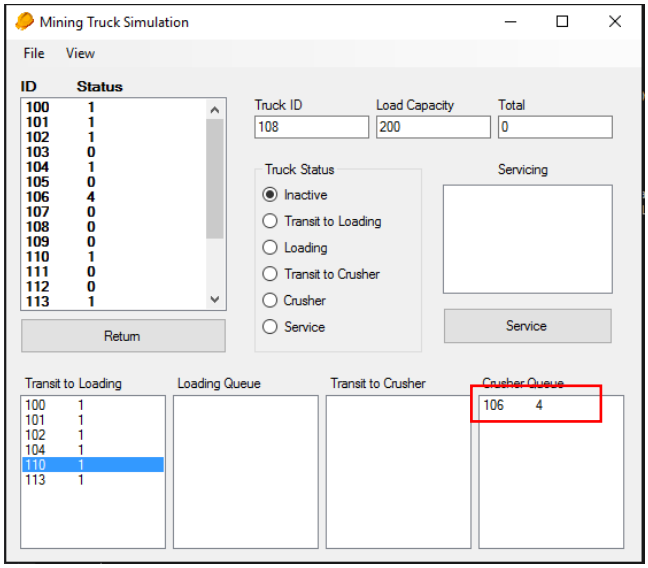
Clicke event (1 click) on the transit to loading list box



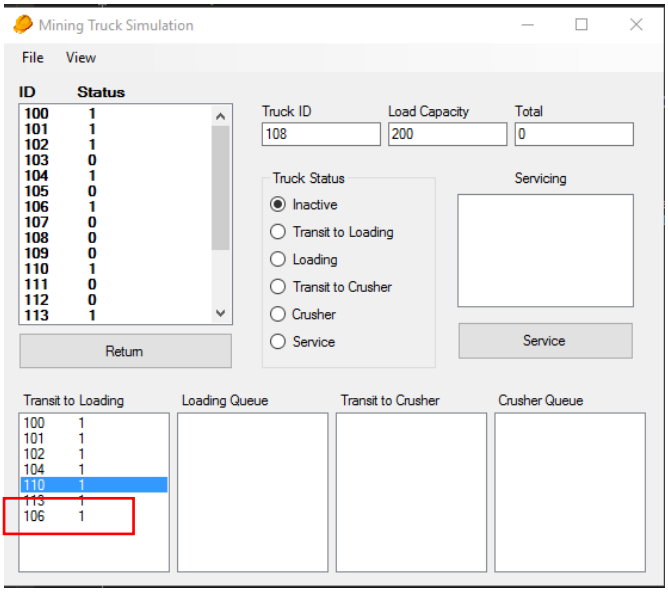
Click event of the Loading list box



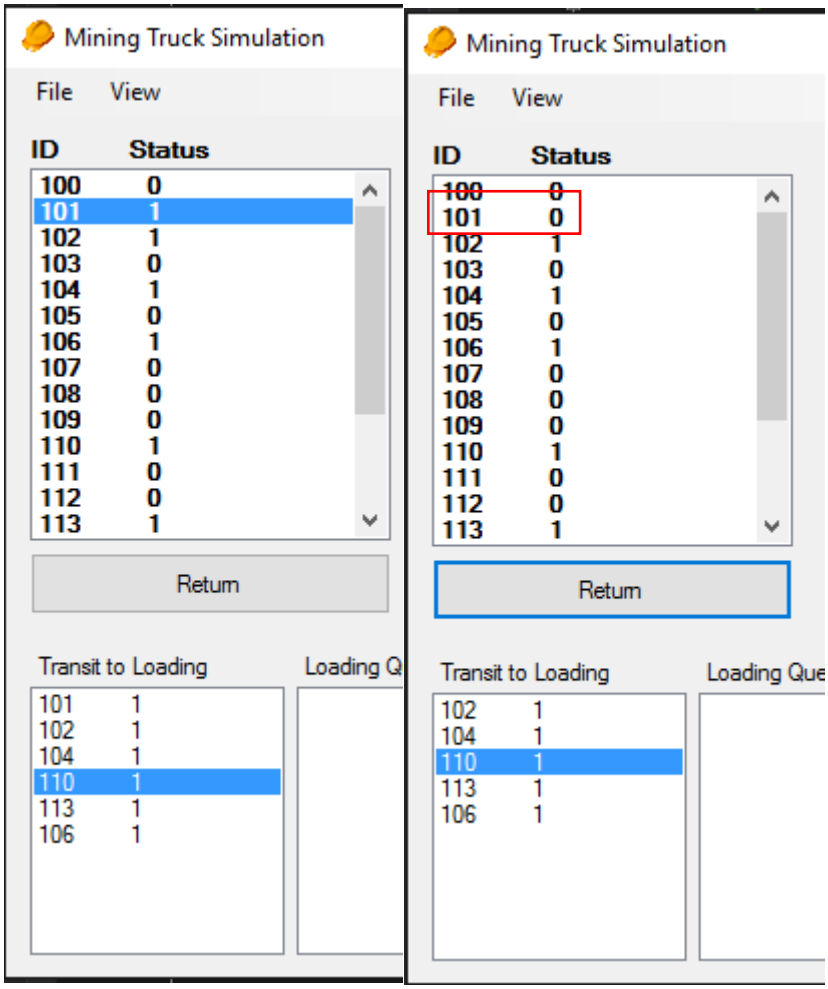
Click event of the transit to crusher list box



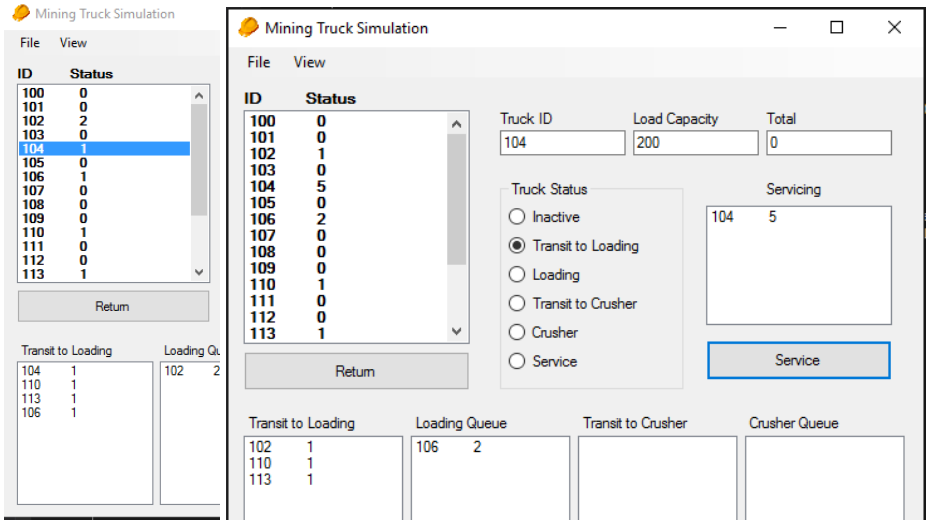
Click event on the Crusher list box



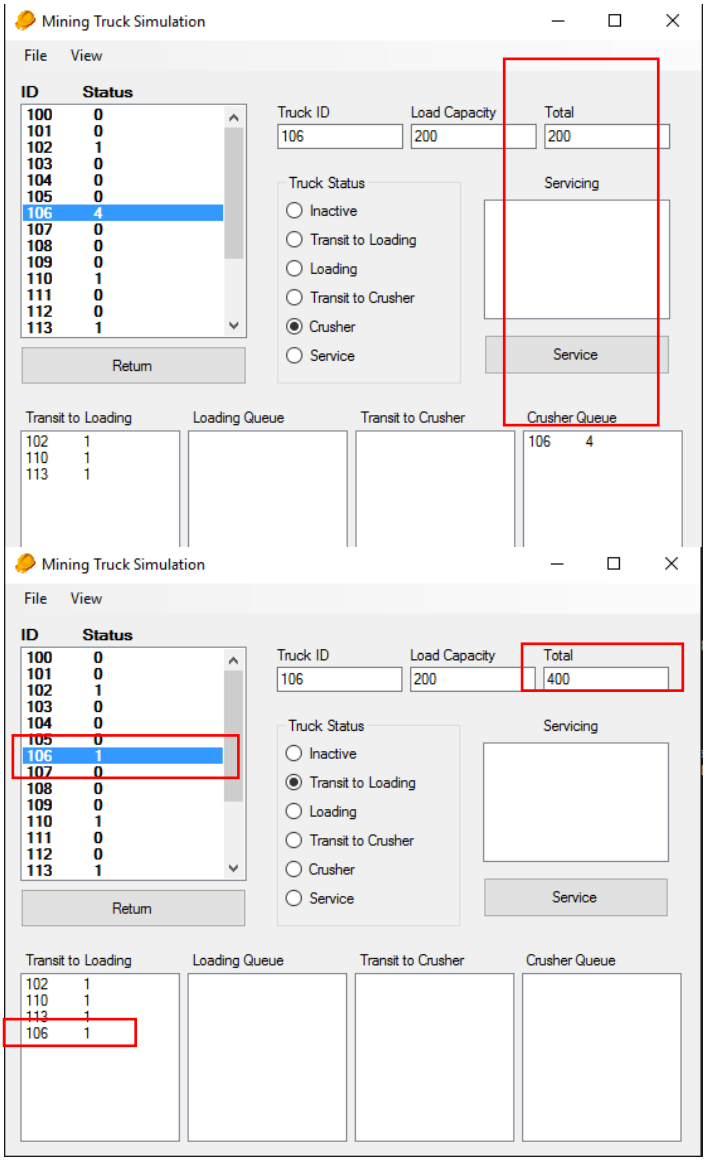
Click of the return button while the truck at the top of the transit to loading queue is selected in the main list box



Click of the return button while the truck at the top of the transit to loading queue is selected in the main list box



Add the total when truck moves from crusher queue back to transit to loading



Code

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Diagnostics;

///Daniel Ranieri
///131600214
///21.06.16
///AT2: Mining Simulator

///This is a program that will create a list of trucks numbered from 100 to whatever
you want
///and allows the user to put the trucks into queues and move them along the queues.
///The user is also allowd to put the trucks into service and make the trucks
inactive.
///The user will also be able to recover the truck list from the last use.

namespace AT2_Mining_Simulator
{
    [Serializable()]
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        /// <summary>
        /// Initializing components
        /// </summary>
        static int max = 20;
        Trucks[] Truck = new Trucks[max];
        Queue<Trucks>[] AQueue = new Queue<Trucks>[4];
        BinaryWriter bw;
        BinaryReader br;

        /// <summary>
        /// what will happen when the form loads
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void Form1_Load(object sender, EventArgs e)
        {
            clearAll();
            createTrucks();
            createQueues();
            BinaryRead();
            displayTrucks();
        }

        /// <summary>
```

```
/// Calls the binary save method on close
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void Form1_FormClosed(object sender, FormClosedEventArgs e)
{
    BinaryWrite();
}

/// <summary>
/// Method to wipe all lists clean
/// </summary>
public void clearAll()
{
    listBoxTrucks.Items.Clear();
    listBoxTLoading.Items.Clear();
    listBoxTCrusher.Items.Clear();
    listBoxLoading.Items.Clear();
    listBoxService.Items.Clear();
    listBoxCrusher.Items.Clear();
}

/// <summary>
/// Method to load a new instance (New button)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void newToolStripMenuItem_Click(object sender, EventArgs e)
{
    clearAll();
    createTrucks();
    displayTrucks();
    createQueues();
}

/// <summary>
/// Method to open the user guide (User Guide Button)
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void userGuideToolStripMenuItem_Click(object sender, EventArgs e)
{
    Process newProcess = new Process();
    try
    {
        newProcess.StartInfo.FileName = "";
        newProcess.Start();
    }
    catch (Exception E) { MessageBox.Show("File Not Found. "); }
}

/// <summary>
/// Close button method
/// </summary>
/// <param name="sender"></param>
/// <param name="e"></param>
private void closeToolStripMenuItem_Click(object sender, EventArgs e)
{
    Close();
}

/// <summary>
```

```
        /// creates the number of trucks
        /// </summary>
        public void createTrucks()
        {
            try
            {
                for (int i = 0; i < max; i++)
                {
                    Truck[i] = new Trucks();
                    Truck[i].gsTruckID = 100 + i;
                    Truck[i].gsTruckLoadCapacity = 200;
                }
            }
            catch { }
        }

        /// <summary>
        /// creates queues to cover each list box
        /// </summary>
        public void createQueues()
        {
            try
            {
                for (int i = 0; i < 4; i++)
                {
                    AQueue[i] = new Queue<Trucks>();
                }
            }
            catch { }
        }

        /// <summary>
        /// Display the trucks in the main list box with current get and set values
        /// </summary>
        public void displayTrucks()
        {
            try
            {
                listBoxTrucks.Items.Clear();
                for (int i = 0; i < max; i++)
                {
                    listBoxTrucks.Items.Add(Truck[i].gsTruckID + "\t" +
Truck[i].gsTruckStatus);
                }
            }
            catch { }
        }

        /// <summary>
        /// Get the information from the listbox and load it into the text boxes
        /// Also making the radio buttons correspond to the truck locations
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBoxTrucks_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                int index = listBoxTrucks.SelectedIndex;
                txtTruck.Text = listBoxTrucks.GetItemText(Truck[index].gsTruckID);
                txtLoadCap.Text =
listBoxTrucks.GetItemText(Truck[index].gsTruckLoadCapacity);
```

```
                txtTotal.Text = listBoxTrucks.GetItemText(Truck[index].gsTruckTotal);

                if (Truck[index].gsTruckStatus == 1)
                {
                    radioTLoading.PerformClick();
                }
                if (Truck[index].gsTruckStatus == 2)
                {
                    radioLoading.PerformClick();
                }
                if (Truck[index].gsTruckStatus == 3)
                {
                    radioTCrusher.PerformClick();
                }
                if (Truck[index].gsTruckStatus == 4)
                {
                    radioCrusher.PerformClick();
                }
                if (Truck[index].gsTruckStatus == 5)
                {
                    radioService.PerformClick();
                }
                if (Truck[index].gsTruckStatus == 0)
                {
                    radioInactive.PerformClick();
                }
            }
            catch { }
        }

        /// <summary>
        /// The method for the double click on the main trucks list
        /// moves the truck to the transit to loading queue and listbox
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBoxTrucks_DoubleClick(object sender, EventArgs e)
        {
            int index = listBoxTrucks.SelectedIndex;
            if (Truck[index].gsTruckStatus == 0)
            {
                AQueue[0].Enqueue(Truck[index]);
                Truck[index].truckSwitch();
                listBoxTLoading.Items.Add(Truck[index].gsTruckID + "\t" +
Truck[index].gsTruckStatus);
            }
            else
            {
                MessageBox.Show("Mate, that wont work. its a duplicate!");
            }
            displayTrucks();
        }

        /// <summary>
        /// The method for the click on the transit to loading box
        /// trucks list moves the truck to the loading queue and listbox
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBoxTLoading_Click(object sender, EventArgs e)
        {
            try
```

```
            {
                Trucks lastTruck = AQueue[0].Peek();
                AQueue[0].Dequeue();
                listBoxTLoading.Items.Remove(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);
                AQueue[1].Enqueue(lastTruck);
                lastTruck.truckSwitch();
                listBoxLoading.Items.Add(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);
                displayTrucks();
            }
            catch { }
        }

        /// <summary>
        /// The method for the click on the loading box
        /// trucks list moves the truck to the loading queue and listbox
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBoxLoading_Click(object sender, EventArgs e)
        {
            try
            {
                Trucks lastTruck = AQueue[1].Peek();
                AQueue[1].Dequeue();
                listBoxLoading.Items.Remove(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);
                AQueue[2].Enqueue(lastTruck);
                lastTruck.truckSwitch();
                listBoxTCrusher.Items.Add(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);
                displayTrucks();
            }
            catch { }
        }

        /// <summary>
        /// The method for the click on the transit to crusher box
        /// trucks list moves the truck to the transit to crusher queue and listbox
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBoxTCrusher_Click(object sender, EventArgs e)
        {
            try
            {
                Trucks lastTruck = AQueue[2].Peek();
                AQueue[2].Dequeue();
                listBoxTCrusher.Items.Remove(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);
                AQueue[3].Enqueue(lastTruck);
                lastTruck.truckSwitch();
                listBoxCrusher.Items.Add(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);

                displayTrucks();
            }
            catch { }
        }

        /// <summary>
```

```
        /// The method for the click on the crusher box
        /// trucks list moves the truck to the transit to loading queue and listbox
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBoxCrusher_Click(object sender, EventArgs e)
        {
            try
            {
                Trucks lastTruck = AQueue[3].Peek();
                AQueue[3].Dequeue();
                listBoxCrusher.Items.Remove(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);
                AQueue[0].Enqueue(lastTruck);
                lastTruck.truckSwitch();
                listBoxTLoading.Items.Add(lastTruck.gsTruckID + "\t" +
lastTruck.gsTruckStatus);
                lastTruck.gsTruckTotal = lastTruck.gsTruckTotal +
lastTruck.gsTruckLoadCapacity;

                displayTrucks();
            }
            catch { }
        }

        /// <summary>
        /// the button for when the item in the service box is clicked. This will
        /// cause the selected item to be deleted from the box on double click
        /// and change the trucks status back to 0
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void listBoxService_SelectedIndexChanged(object sender, EventArgs e)
        {
            try
            {
                string itemR = listBoxService.SelectedItem.ToString().Remove(3);
                int index = Int32.Parse(itemR) - 100;
                listBoxService.Items.Remove(Truck[index].gsTruckID + "\t" +
Truck[index].gsTruckStatus);
                Truck[index].truckSwitch();

                displayTrucks();
            }
            catch { }
        }

        /// <summary>
        /// This will return a truck to the main box only when the button is clicked
        /// while the truck is in the transit to loading queue at the top.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnReturn_Click(object sender, EventArgs e)
        {
            try
            {
                int index = listBoxTrucks.SelectedIndex;
                Trucks lastTruck = AQueue[0].Peek();
                if (lastTruck.gsTruckID == Truck[index].gsTruckID)
                {
```

```
                    listBoxTLoading.Items.Remove(Truck[index].gsTruckID + "\t" +
Truck[index].gsTruckStatus);
                    Truck[index].gsTruckStatus = 0;
                    AQueue[0].Dequeue();
                }
                displayTrucks();
            }
            catch { }
        }

        /// <summary>
        /// The button for when the service button is clicked. This will make the
truck
        /// get added to the service box and change the status to 5
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void btnService_Click(object sender, EventArgs e)
        {
            try
            {
                int index = listBoxTrucks.SelectedIndex;
                //Trucks lastTruck = AQueue[0].Peek();
                if(Truck[index].gsTruckStatus == 0)
                {
                Truck[index].gsTruckStatus = 5;
                listBoxService.Items.Add(Truck[index].gsTruckID + "\t" +
Truck[index].gsTruckStatus);
                }
                else if (AQueue[0].Peek() == Truck[index])
                {
                    listBoxTLoading.Items.Remove(Truck[index].gsTruckID + "\t" +
Truck[index].gsTruckStatus);
                    Truck[index].gsTruckStatus = 5;
                    AQueue[0].Dequeue();
                    listBoxService.Items.Add(Truck[index].gsTruckID + "\t" +
Truck[index].gsTruckStatus);
                }
                displayTrucks();
            }
            catch { }
        }

        /// <summary>
        /// Binary Writer method to write the file to a binary format
        /// </summary>
        public void BinaryWrite()
        {
            try
            {
                bw = new BinaryWriter(new FileStream("Trucks.dat", FileMode.Create));
            }
            catch (Exception fe)
            {
                MessageBox.Show(fe.Message + "\n Cannot Write to file.");
                return;
            }

            foreach (Trucks Truck in Truck)
            {
                try
                {
```

```
                    bw.Write(Truck.gsTruckID);
                    bw.Write(Truck.gsTruckStatus);
                    bw.Write(Truck.gsTruckTotal);
                }
                catch (Exception fe)
                {
                    return;
                }
            }
            bw.Close();
        }

        /// <summary>
        /// A method to read the file from a binary format
        /// </summary>
        public void BinaryRead()
        {
            listBoxTrucks.Items.Clear();
            try
            {
                br = new BinaryReader(new FileStream("Trucks.dat", FileMode.Open));
            }
            catch (Exception fe)
            {
                MessageBox.Show(fe.Message + "\n Cannot open file for reading");
                return;
            }
            try
            {
                for (int i = 0; i < 20; i++)
                {
                    Truck[i].gsTruckID = br.ReadInt32();
                    Truck[i].gsTruckStatus = br.ReadInt32();
                    Truck[i].gsTruckTotal = br.ReadInt32();
                }
            }
            catch (Exception fe) { }
            br.Close();

            //puts the trucks back into their queues
            for (int i = 0; i < 20; i++)
            {
                switch (Truck[i].gsTruckStatus)
                {
                    case 1:
                        AQueue[0].Enqueue(Truck[i]);
                        listBoxTLoading.Items.Add(Truck[i].gsTruckID + "\t" +
Truck[i].gsTruckStatus);
                        break;
                    case 2:
                        AQueue[1].Enqueue(Truck[i]);
                        listBoxLoading.Items.Add(Truck[i].gsTruckID + "\t" +
Truck[i].gsTruckStatus);
                        break;
                    case 3:
                        AQueue[2].Enqueue(Truck[i]);
                        listBoxTCrusher.Items.Add(Truck[i].gsTruckID + "\t" +
Truck[i].gsTruckStatus);
                        break;
                    case 4:
                        AQueue[3].Enqueue(Truck[i]);
```

```
                        listBoxCrusher.Items.Add(Truck[i].gsTruckID + "\t" +
Truck[i].gsTruckStatus);
                        break;
                    case 5:
                        listBoxService.Items.Add(Truck[i].gsTruckID + "\t" +
Truck[i].gsTruckStatus);
                        break;
                    default: break;
                }
            }
        }
    }
}
```