**Course Title:** Computer Science Level 6

**Module:** Object Orientated Programming

**Course Code:** 6N2108

**Tutor Name:** Keith Quille

**Assignment Title:** Skill Demo 1

**Student:** James Fleming

**Student PPSN:** 7706728A

**Date:** 16/04/2017

**Contents**

1. **Code**

* **PetrolPump**
* **Main Class**

1. **Testing**

* **Insufficient order amount**
* **Not enough fuel in pump tank**
* **Sufficient order price displays**
* **Fuel added to pump tank**

1. **User Manual**

* **Manual**
* **Class Diagram**
* **XML**

**CODE**

**Petrol Pump**

//James Fleming

//CS-021

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOP\_Skills\_Demo\_1

{

class PetrolPump

{

//member variables

private double petrolPrice;

private double fuelLevel;

private double minOrder;

private string fuelType;

//constructors

/// <summary>

/// default constructor for when the object is created and the creator doesn't input any values

/// </summary>

public PetrolPump()

{

this.petrolPrice = 0.0;

this.fuelLevel = 0.0;

this.minOrder = 0.0;

this.fuelType = "";

}

/// <summary>

/// overloaded constructor to allow user input the values when the object is created

/// </summary>

/// <param name="price">Petrol price : double</param>

/// <param name="level">Fuel in tank : double</param>

/// <param name="order">Minimum order customer can make : double</param>

/// <param name="type">what fuel it is : string</param>

public PetrolPump(double price, double level, double order, string type)

{

petrolPrice = price;

fuelLevel = level;

minOrder = order;

fuelType = type;

}

//gets and sets

//gets

/// <summary>

/// Gets the petrol Price

/// </summary>

/// <returns>Price of Petrol : double</returns>

public double getPetrolPrice()

{

return this.petrolPrice;

}

/// <summary>

/// Gets how much is left in the tank

/// </summary>

/// <returns>Current tank Capacity : double</returns>

public double getFuelLevel()

{

return this.fuelLevel;

}

/// <summary>

/// finds out what the minimum order is

/// </summary>

/// <returns>What the minimum order : double</returns>

public double getMinOrder()

{

return this.minOrder;

}

/// <summary>

/// finds out what fuel type it is

/// </summary>

/// <returns>what type of fuel it is : string</returns>

public string getFuelType()

{

return this.fuelType;

}

//set

/// <summary>

/// sets the price of petrol

/// </summary>

/// <param name="petrolPriceIn">price of petrol : double</param>

public void setPetrolPrice(double petrolPriceIn)

{

this.petrolPrice = petrolPriceIn;

}

/// <summary>

/// sets the quantity of fuel in the tank

/// </summary>

/// <param name="fuelLevelIn">quantity of fuel in tank : double</param>

public void setFuelLevel(double fuelLevelIn)

{

this.fuelLevel = fuelLevelIn;

}

/// <summary>

/// sets the minimum order on a fill of fuel

/// </summary>

/// <param name="minOrderIn">minimum order of fuel : double</param>

public void setMinOrder(double minOrderIn)

{

this.minOrder = minOrderIn;

}

/// <summary>

/// what type of fuel is in the tank

/// </summary>

/// <param name="fuelTypeIn">what type of fuel is in the tank : string</param>

public void setFuelType(string fuelTypeIn)

{

this.fuelType = fuelTypeIn;

}

//behaviours

/// <summary>

/// adds the inputted amount to the tank of fuel

/// </summary>

/// <param name="fuelAdded"> amount of fuel added : double</param>

public void addFuel(double fuelAdded)

{

this.setFuelLevel(getFuelLevel() + fuelAdded);

}

/// <summary>

/// Checks required each time fuel is pumped from the tank

/// </summary>

/// <param name="custFill">the amount of fuel the customer wants : double</param>

public void fillCarFromPump(double custFill)

{

//checks if what the customers order is less than the minimum order and if it is prints appropriate error message to screen

if(custFill< minOrder)

{

Console.Clear();

Console.WriteLine("Apologies we have a minimum order of " + getMinOrder() + " Litres");

}

//checks if there's enough fuel in the tank for the customers order if there isn't print appropriate error message to screen

if(fuelLevel<custFill)

{

Console.Clear();

Console.WriteLine("Apologies there is insufficient Fuel for your order.");

}

//checks if there's enough fuel for customer's order and if it is greater than or equal to the minimum order if it is

//it removes customer's fuel from tank's current capacity and prints to screen the price the customer is charged

if(custFill<fuelLevel && custFill>=minOrder)

{

this.setFuelLevel(getFuelLevel() - custFill);

Console.Clear();

Console.WriteLine("Your purchase comes to : " + (getPetrolPrice() \* custFill)+ "euro");

}

}

}

}

**Main Class**

//James Fleming

//CS-021

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOP\_Skills\_Demo\_1

{

class Program

{

static void Main(string[] args)

{

//creates an instance of a petrol pump called pump1 with the relevant data

PetrolPump pump1 = new PetrolPump(0.75, 20000.00, 2.0, "Unleaded 98");

//requests the user to input how many litres of fuel they want

Console.WriteLine("How many Litres of " + pump1.getFuelType() + " would you like?");

//inputs the users input into the fillCarFromPump method

pump1.fillCarFromPump(Convert.ToInt32(Console.ReadLine()));

//prompts the user to press any key to continue

Console.WriteLine("Press any key to continue");

//detects if user presses a key

Console.ReadKey();

//adds a 1000 litres of fuel to the Pumps tank

pump1.addFuel(1000.0);

//clears the screen

Console.Clear();

//displays to the user how much fuel is in the Pumps Tank after the 1000 litres of fuel is added

Console.WriteLine("There is "+pump1.getFuelLevel()+" Litres of fuel in the Pumps Tank");

Console.ReadKey();

}

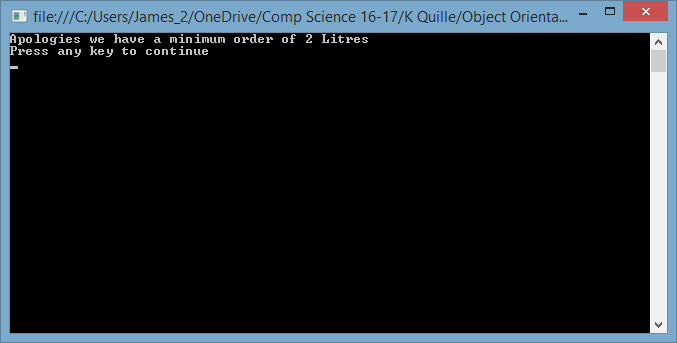
}

}

**Testing**

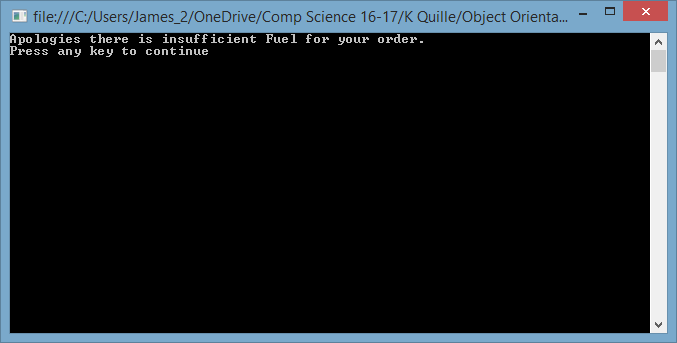
**Insufficient order amount**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| Order an amount of fuel less than minimum order (2 litres) | "Apologies we have a minimum order of 2 litres" | Appropriate  error message displayed | Test Success |

****

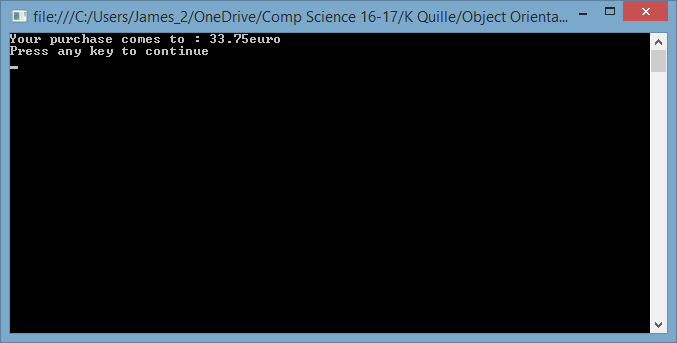
**Not enough fuel in pump tank**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| Order more fuel than what is in the Pump's tank | "Apologies there is insufficient Fuel for your order." | Appropriate  error message displayed | Test Success |

****

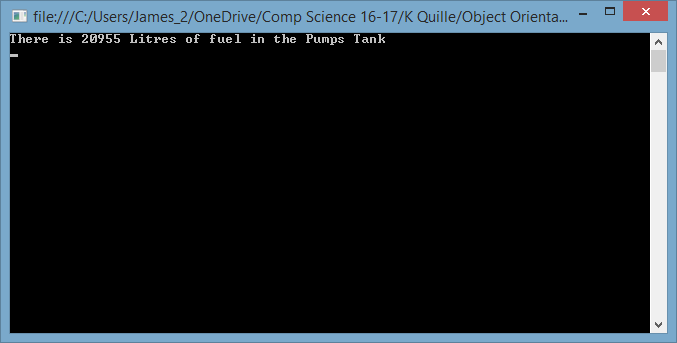
**Sufficient order price displays**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| Order 45 litres of fuel | Message displaying the price of the fuel ordered (33·75) | Message displayed | Test Success |

****

**Fuel added to pump tank**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test** | **Expected Result** | **Result** | **Conclusion** |
| To see if the addFuel() method works | 45 litres removed after purchase then 1000 litres added means there should 20955 litres in the pumps tank | There is 20955 litres of fuel in the pumps tank | Test Success |

****

**User Manual**

**Class Diagram**

****

**XML**

<?xml version="1.0"?>

<doc>

<assembly>

<name>OOP\_Skills\_Demo\_1</name>

</assembly>

<members>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.#ctor">

<summary>

default constructor for when the object is created and the creator doesn't input any values

</summary>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.#ctor(System.Double,System.Double,System.Double,System.String)">

<summary>

overloaded constructor to allow user input the values when the object is created

</summary>

<param name="price">Petrol price : double</param>

<param name="level">Fuel in tank : double</param>

<param name="order">Minimum order customer can make : double</param>

<param name="type">what fuel it is : string</param>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.getPetrolPrice">

<summary>

Gets the petrol Price

</summary>

<returns>Price of Petrol : double</returns>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.getFuelLevel">

<summary>

Gets how much is left in the tank

</summary>

<returns>Current tank Capacity : double</returns>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.getMinOrder">

<summary>

finds out what the minimum order is

</summary>

<returns>What the minimum order : double</returns>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.getFuelType">

<summary>

finds out what fuel type it is

</summary>

<returns>what type of fuel it is : string</returns>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.setPetrolPrice(System.Double)">

<summary>

sets the price of petrol

</summary>

<param name="petrolPriceIn">price of petrol : double</param>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.setFuelLevel(System.Double)">

<summary>

sets the quantity of fuel in the tank

</summary>

<param name="fuelLevelIn">quantity of fuel in tank : double</param>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.setMinOrder(System.Double)">

<summary>

sets the minimum order on a fill of fuel

</summary>

<param name="minOrderIn">minimum order of fuel : double</param>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.setFuelType(System.String)">

<summary>

what type of fuel is in the tank

</summary>

<param name="fuelTypeIn">what type of fuel is in the tank : string</param>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.addFuel(System.Double)">

<summary>

adds the inputted amount to the tank of fuel

</summary>

<param name="fuelAdded"> amount of fuel added : double</param>

</member>

<member name="M:OOP\_Skills\_Demo\_1.PetrolPump.fillCarFromPump(System.Double)">

<summary>

Checks required each time fuel is pumped from the tank

</summary>

<param name="custFill">the amount of fuel the customer wants : double</param>

</member>

</members>

</doc>