1. MDA-EFSM model for the *GasPump* components

i. A list of meta events for the MDA-EFSM

MDA-EFSM Events:

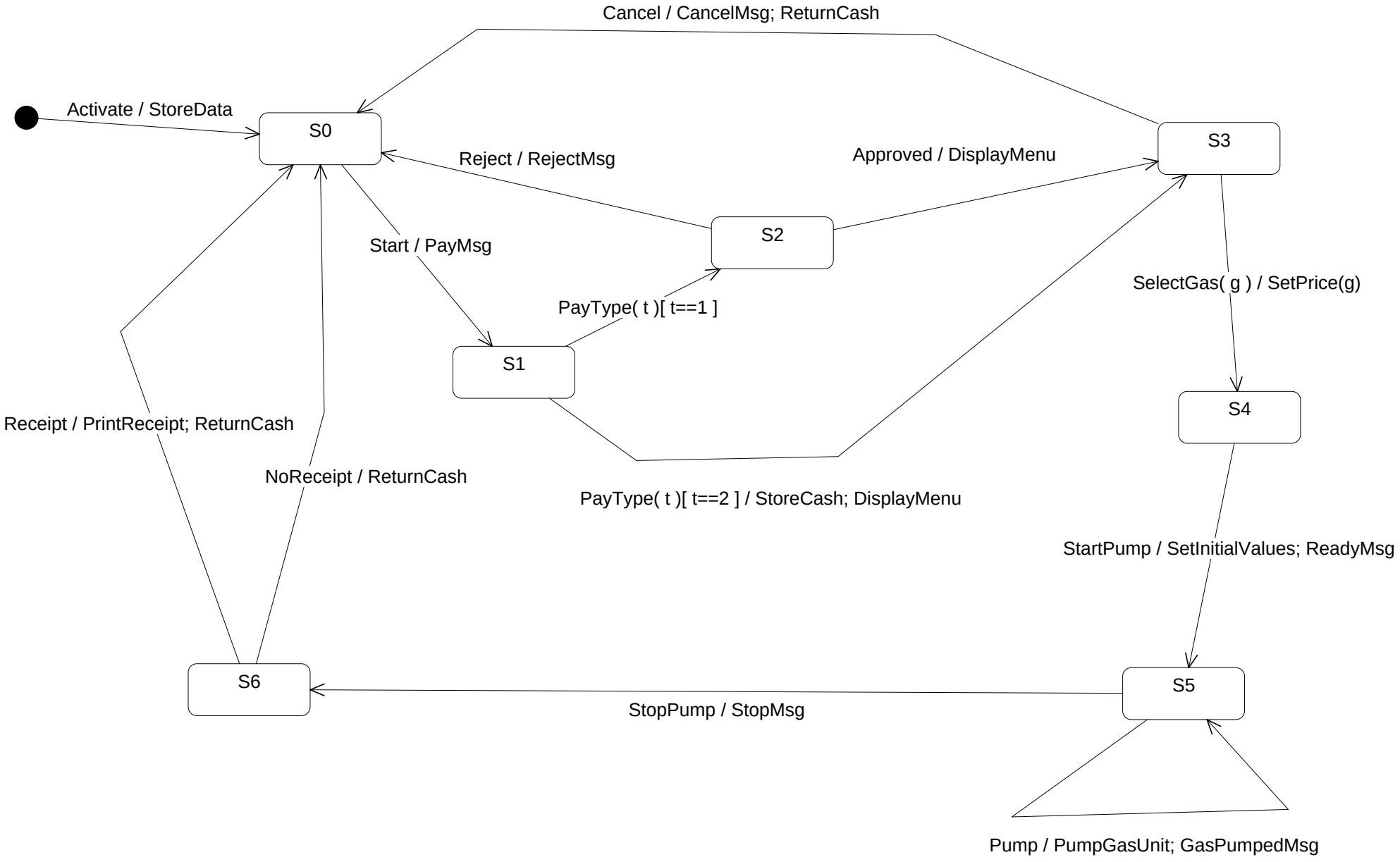
Activate()
Start()
PayType(int t) //credit: t=1; cash: t=2
Reject()
Cancel()
Approved()
StartPump()
Pump()
StopPump()
SelectGas(int g)
Receipt()
NoReceipt()

ii. A list of meta actions for the MDA-EFSM with their descriptions

MDA-EFSM Actions:

StoreData	// stores price(s) for the gas from the temporary data store
PayMsg	// displays a type of payment method
StoreCash	// stores cash from the temporary data store
DisplayMenu	// display a menu with a list of selections
RejectMsg	// displays credit card not approved message
SetPrice(int g)	// set the price for the gas identified by *g* identifier
ReadyMsg	// displays the ready for pumping message
SetInitialValues	// set *G* (or *L*) and *total* to 0
PumpGasUnit	// disposes unit of gas and counts # of units disposed
GasPumpedMsg	// displays the amount of disposed gas
StopMsg	// stop pump message and receipt? msg (optionally)
PrintReceipt	// print a receipt
CancelMsg	// displays a cancellation message
ReturnCash	// returns the remaining cash

iii. A state diagram of the MDA-EFSM



MDA-EFSM for Gas Pumps

iv. Pseudo-code of all operations of Input Processors of *GasPump-1* and *GasPump-2*

Operations of the Input Processor
(GasPump-1)

```
Activate(float a, float b) {
        if ((a>0)&&(b>0)) {
                d->temp_a=a;
                d->temp_b=b;
                m->Activate()
        }
}

Start() {
        m->Start();
}

PayCredit() {
        m->PayType(1);
}

Reject() {
        m->Reject();
}

Cancel() {
        m->Cancel();
}

Approved() {
        m->Approved();
}

Super() {
        m->SelectGas(2)
}

Regular() {
        m->SelectGas(1)
}

StartPump() {
        m->StartPump();
}

PumpGallon() {
        m->Pump();

StopPump() {
        m->StopPump();
        m->Receipt();
}
```

Notice:
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

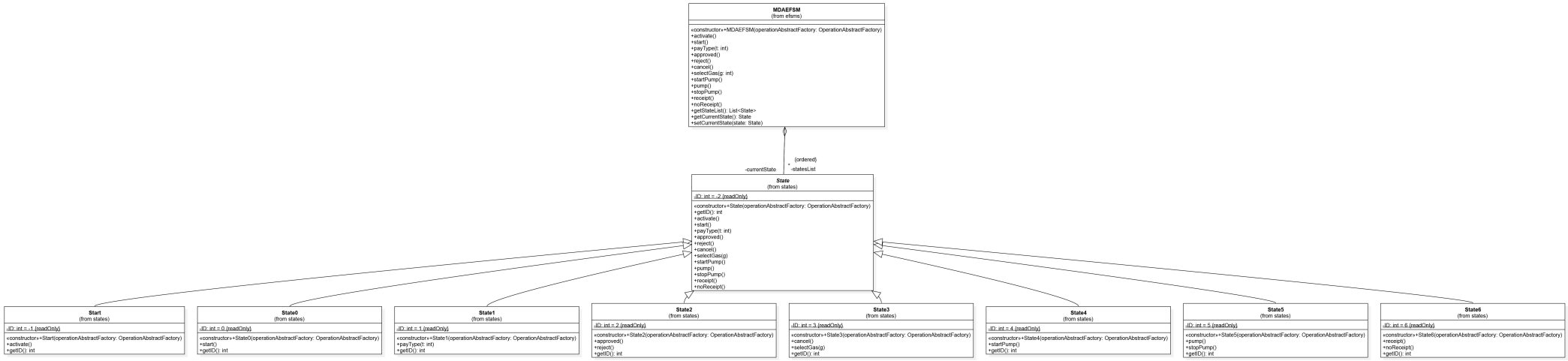
Operations of the Input Processor
(GasPump-2)

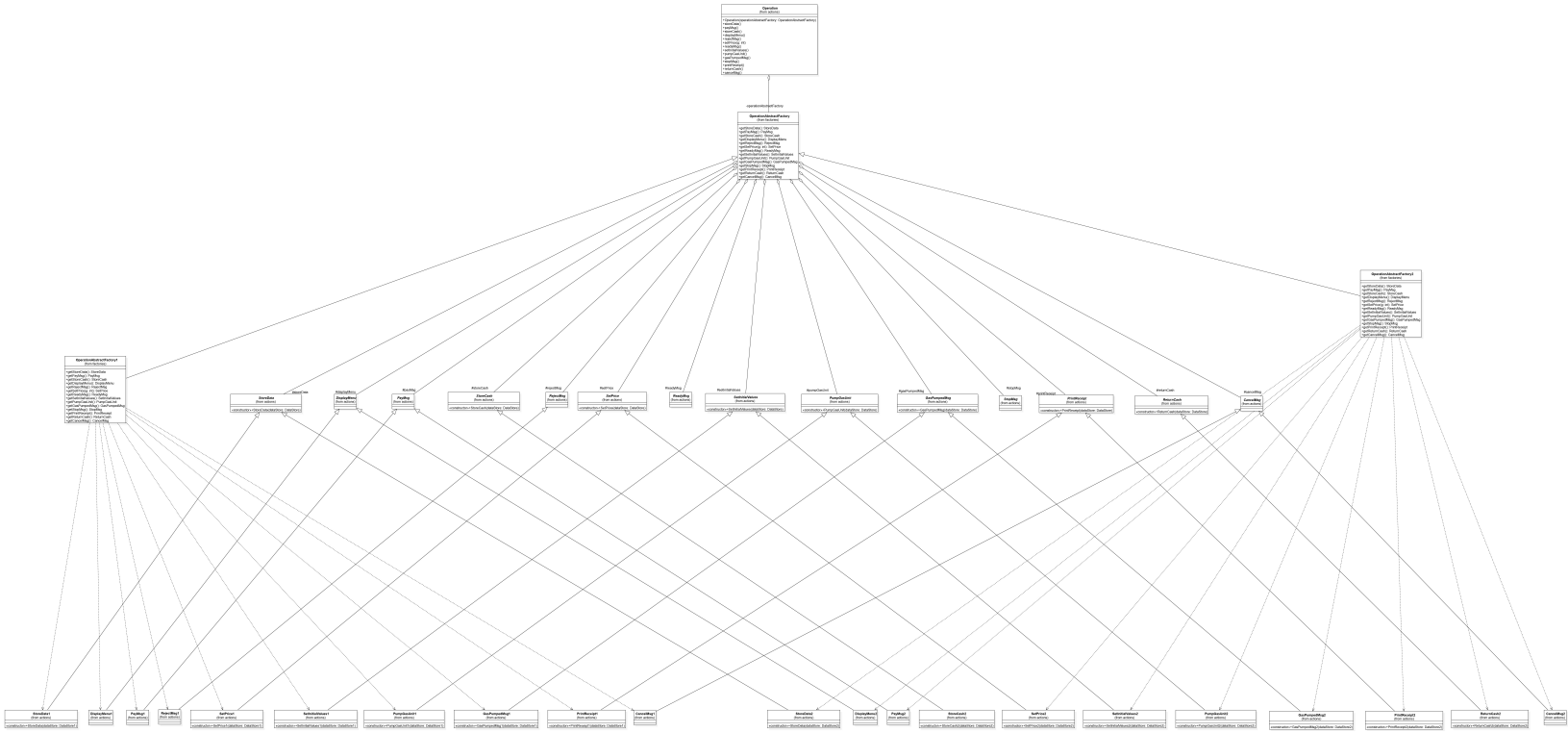
```
Activate(int a, int b, int c) {
        if ((a>0)&&(b>0)&&(c>0)) {
                d->temp_a=a;
                d->temp_b=b;
                d->temp_c=c
                m->Activate()
        }
}

Start() {
        m->Start();
}

PayCash(float c) {
        if (c>0) {
                d->temp_cash=c;
                m->PayType(2)
        }
}

Cancel() {
        m->Cancel();
}

Super() {
        m->SelectGas(2);
}

Premium() {
        m->SelectGas(3);
}

Regular() {
        m->SelectGas(1);
}

StartPump() {
        m->StartPump();
}

PumpLiter() {
        if (d->cash<(d->L+1)*d->price)
                m->StopPump();
        else m->Pump()
}

Stop() {
        m->StopPump();
}

Receipt() {
        m->Receipt();
}

NoReceipt() {
        m->NoReceipt();
}
```

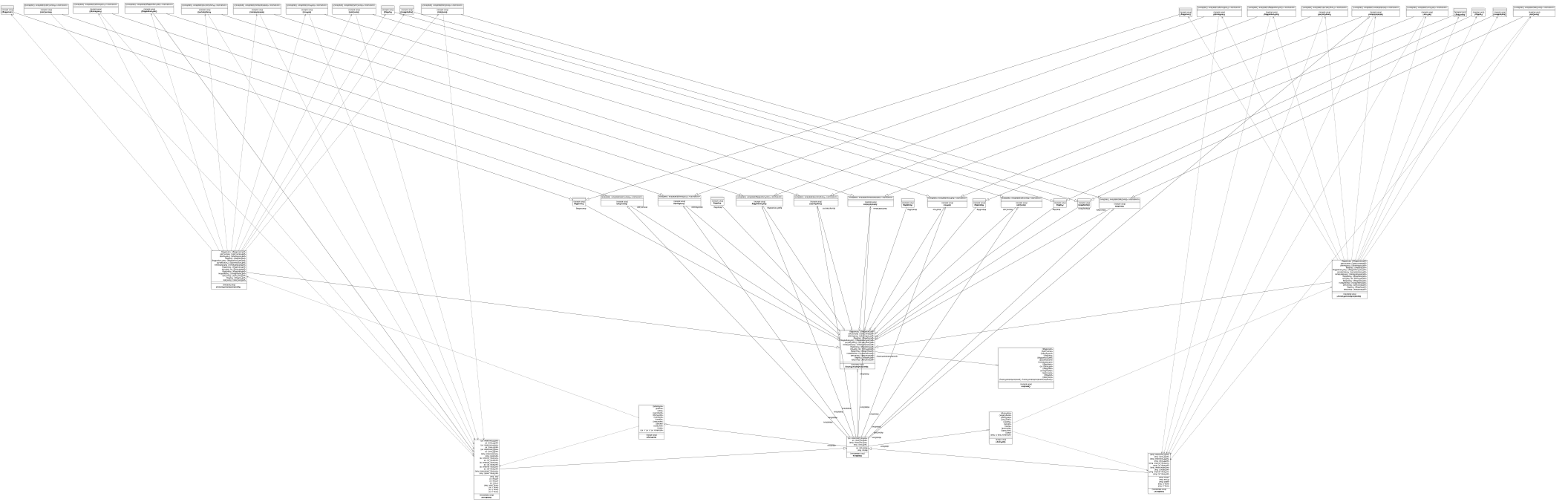
Notice:
cash: contains the value of cash deposited
price: contains the price of the selected gas
L: contains the number of liters already pumped

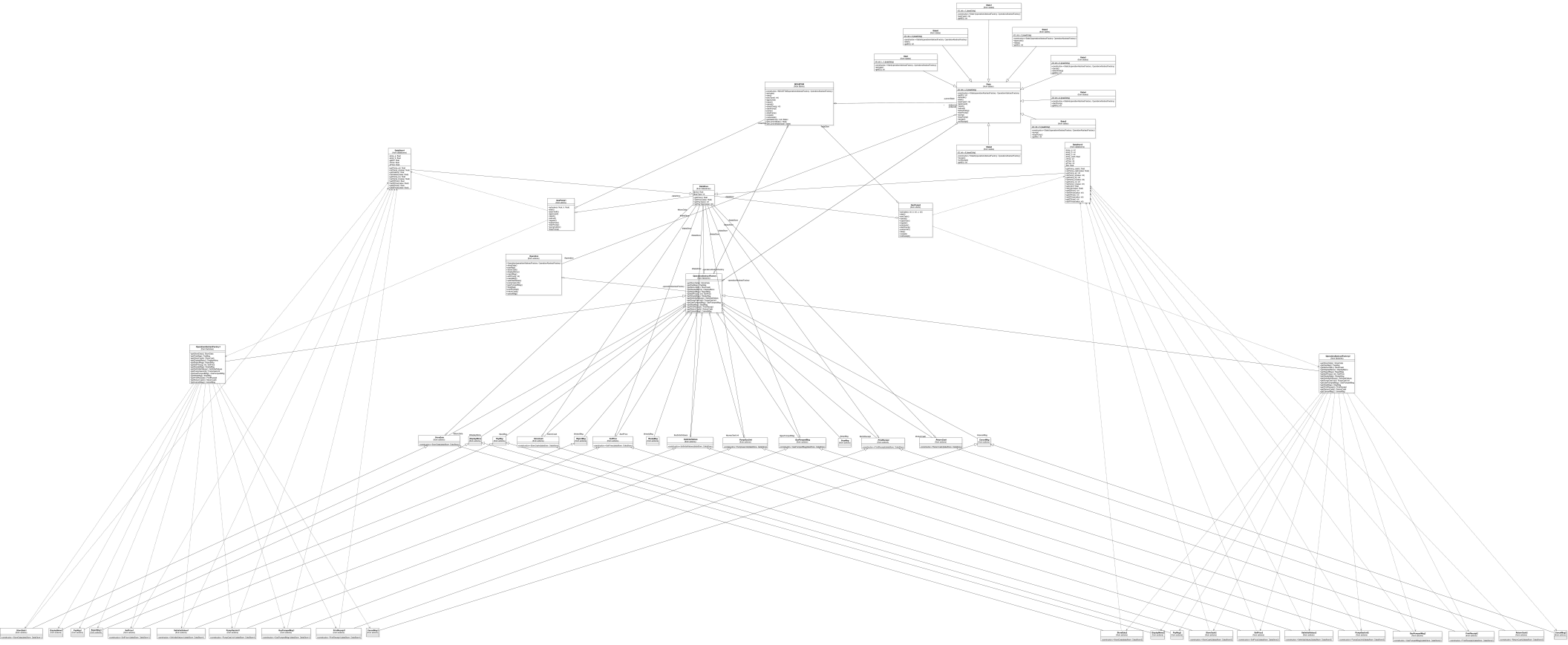
cash , *L*, *price* are in the data store
m: is a pointer to the MDA-EFSM object
d: is a pointer to the Data Store object

2. Class diagram(s) of the MDA of the *GasPump* components.


MDAEFSM
(from efsms)
«constructor»+MDAEFSM(operationAbstractFactory: OperationAbstractFactory)
+activate()
+start()
+payType(t: int)
+approved()
+reject()
+cancel()
+selectGas(g: int)
+startPump()
+pump()
+stopPump()
+receipt()
+noReceipt()
+getStateList(): List<State>
+getCurrentState(): State
+setCurrentState(state: State)
-currentState
{ordered}
-statesList
State
(from states)
-ID: int = -2 {readOnly}
«constructor»+State(operationAbstractFactory: OperationAbstractFactory)
+getID(): int
+activate()
+start()
+payType(t: int)
+approved()
+reject()
+cancel()
+selectGas(g)
+startPump()
+pump()
+stopPump()
+receipt()
+noReceipt()
Start
(from states)
-ID: int = -1 {readOnly}
«constructor»+Start(operationAbstractFactory: OperationAbstractFactory)
+activate()
+getID(): int
State0
(from states)
-ID: int = 0 {readOnly}
«constructor»+State0(operationAbstractFactory: OperationAbstractFactory)
+start()
+getID(): int
State1
(from states)
-ID: int = 1 {readOnly}
«constructor»+State1(operationAbstractFactory: OperationAbstractFactory)
+payType(t: int)
+getID(): int
State2
(from states)
-ID: int = 2 {readOnly}
«constructor»+State2(operationAbstractFactory: OperationAbstractFactory)
+approved()
+reject()
+getID(): int
State3
(from states)
-ID: int = 3 {readOnly}
«constructor»+State3(operationAbstractFactory: OperationAbstractFactory)
+cancel()
+selectGas(g)
+getID(): int
State4
(from states)
-ID: int = 4 {readOnly}
«constructor»+State4(operationAbstractFactory: OperationAbstractFactory)
+startPump()
+getID(): int
State5
(from states)
-ID: int = 5 {readOnly}
«constructor»+State5(operationAbstractFactory: OperationAbstractFactory)
+pump()
+stopPump()
+getID(): int
State6
(from states)
-ID: int = 6 {readOnly}
«constructor»+State6(operationAbstractFactory: OperationAbstractFactory)
+receipt()
+noReceipt()
+getID(): int


Operation
(from actions)
+ Operation(operationBrandFactory: OperationBrandFactory)
+ storeCase()
+ setBag()
+ setPrice()
+ displayBox()
+ setCash()
+ setPricing(int)
+ setPump()
+ setInfoStatus()
+ setPumpOne()
+ getTwoPumpBag()
+ setBox()
+ setPrice()
+ returnCash()
- getCoinBag()
-operationBrandFactory
OperationBrandFactory
(from factories)
+ getStoreCase( ): StoreCase
+ getPumpBag( ): PumpBag
+ getStoreCash( ): StoreCash
+ getPricingBag( ): PricingBag
+ getFreeBag( ): FreeBag
+ getPriceBag( ): PriceBag
+ getInfoStatusBag( ): InfoStatusBag
+ getPumpOneBag( ): PumpOneBag
+ getTwoPumpBag( ): TwoPumpBag
+ getDisplayBag( ): DisplayBag
+ getReturnCash( ): ReturnCash
+ getCoinBag( ): CoinBag
OperationBrandFactory1
(from factories)
+ getStoreCase( ): StoreCase
+ getPumpBag( ): PumpBag
+ getStoreCash( ): StoreCash
+ getChangeMoney( ): DisplayMoney
+ getBox( ): BoxBag
+ getPricing(int): SetPrice
+ getPump( ): SetPump
+ getInfoStatus( ): SetInfoStatus
+ getPumpOne( ): SetPumpOne
+ getTwoPumpBag( ): GetPumpOneBag
+ getDisplay( ): DisplayBag
+ getPriceBag( ): PriceBagLogic
+ getReturnCash( ): ReturnCash
+ getCoinBag( ): CoinBag
StoreCase
(from actions)
StoreCase
+ constructor = StoreCase(pageDisplay: DataStore)
DisplayBox
(from actions)
DisplayBox
PricingBag
(from actions)
PricingBag
FreeBag
(from actions)
FreeBag
StoreCash
(from actions)
StoreCash
+ constructor = StoreCash(pageDisplay: DataStore)
PumpBag
(from actions)
PumpBag
SetPrice
(from actions)
SetPrice
+ constructor = SetPrice(pageDisplay: DataStore)
PriceBag
(from actions)
PriceBag
InfoStatusBag
(from actions)
InfoStatusBag
+ constructor = InfoStatusBag(pageDisplay: DataStore)
PumpOneBag
(from actions)
PumpOneBag
+ constructor = PumpOneBag(pageDisplay: DataStore)
GetPumpOneBag
(from actions)
GetPumpOneBag
+ constructor = GetPumpOneBag(pageDisplay: DataStore)
DisplayMoney
(from actions)
DisplayMoney
BoxBag
(from actions)
BoxBag
ReturnCash
(from actions)
ReturnCash
+ constructor = ReturnCash(pageDisplay: DataStore)
CoinBag
(from actions)
CoinBag
StoreCase
(from actions)
+ constructor = StoreCase(pageDisplay: DataStore)
DisplayBox
(from actions)
PumpBag
(from actions)
FreeBag
(from actions)
StoreCash
(from actions)
+ constructor = StoreCash(pageDisplay: DataStore)
SetPrice
(from actions)
+ constructor = SetPrice(pageDisplay: DataStore)
InfoStatusBag
(from actions)
+ constructor = InfoStatusBag(pageDisplay: DataStore)
PumpOneBag
(from actions)
+ constructor = PumpOneBag(pageDisplay: DataStore)
GetPumpOneBag
(from actions)
+ constructor = GetPumpOneBag(pageDisplay: DataStore)
PriceBagLogic
(from actions)
+ constructor = PriceBagLogic(pageDisplay: DataStore)
ReturnCash1
(from actions)
+ constructor = ReturnCash1(pageDisplay: DataStore)
CoinBag1
(from actions)





3. For each class, description of the purpose of the class, and the responsibility of each operation supported by each class.

Page 1-13: Package iit.softwaresystarch.project.controller
Page 2-4: Class ButtonActivatePump1ClickedListener
Page 5-7: Class ButtonActivatePump2ClickedListener
Page 8-10: Class ButtonCashClickedListener
Page 11-13 : Class ButtonClickedListener

Page 14-75: Package iit.softwaresystarch.project.model.actions
Page 15: Class CancelMsg
Page 16: Class CancelMsg1
Page 17: Class CancelMsg2
Page 18: Class DisplayMenu
Page 19: Class DisplayMenu1
Page 20: Class DisplayMenu2
Page 21-22: Class GasPumpedMsg
Page 23-24: Class GasPumpedMsg1
Page 25-26: Class GasPumpedMsg2
Page 27-30: Class Operation
Page 31: Class PayMsg
Page 32 Class PayMsg1
Page 33: Class PayMsg2
Page 34-35: Class PrintReceipt
Page 36-37: Class PrintReceipt1
Page 38-39: Class PrintReceipt2
Page 40-41: Class PumpGasUnit
Page 42-43: Class PumpGasUnit1
Page 44-45: Class PumpGasUnit2
Page 46: Class ReadyMsg
Page 47: Class RejectMsg
Page 48: Class RejectMsg1
Page 49-50: Class ReturnCash
Page 51-52: Class ReturnCash2
Page 53-54: Class SetInitialValues
Page 55-56: Class SetInitialValues1
Page 57-58: Class SetInitialValues2
Page 59-60: Class SetPrice
Page 61-62: Class SetPrice1
Page 63-64: Class SetPrice2
Page 65: Class Class StopMsg
Page 66-67: Class StoreCash
Page 68-69: Class StoreCash2
Page 70-71: Class StoreData
Page 72-73: Class StoreData1
Page 74-75: Class StoreData2

Page 76-90: Package iit.softwaresystarch.project.model.datastores
Page 77-79: Class DataStore
Page 80-83: Class DataStore1
Page 84-90: Class DataStore2

Page 91-103: Package iit.softwaresystarch.project.model.efsms
Page 92-95: Class GasPump1
Page 96-99: Class GasPump2
Page 100-103: Class MDAEFSM

Page 104-119: Package iit.softwaresystarch.project.model.factories
Page 105-111: Class OperationAbstractFactory
Page 112-115: Class OperationAbstractFactory1
Page 116-119: Class OperationAbstractFactory2

Page 120-140: Package iit.softwaresystarch.project.model.states
Page 121-122: Class Start
Page 123-126: Class State
Page 127-128: Class State0
Page 129-130: Class State1
Page 131-132: Class State2
Page 133-134: Class State3
Page 135-136: Class State4
Page 137-138: Class State5
Page 139-140: Class State6

Page 141-182: Package iit.softwaresystarch.project.view
Page 142-145: Class Driver
Page 146-157: Class GasPump1Panel
Page 158-170: Class GasPump2Panel
Page 171-182: Class InputPanel

Package
iit.softwaresystarch.project.controller

is a package containing the classes mostly related to Controller design pattern.

iit.softwaresystarch.project.controller
Class ButtonActivatePump1ClickedListener

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.controller.ButtonActivatePump1ClickedListener
```

All Implemented Interfaces:
java.awt.event.ActionListener

---

public class **ButtonActivatePump1ClickedListener**
extends java.lang.Object
implements java.awt.event.ActionListener

The listener interface for receiving buttonActivatePump1Clicked events. The class that is interested in processing a buttonActivatePump1Clicked event implements this interface, and the object created with that class is registered with a component using the component's addButtonActivatePump1ClickedListener method. When the buttonActivatePump1Clicked event occurs, that object's appropriate method is invoked.

Field Summary

private	baos The baos.
private final	gasPump1 The gas pump 1.
private final	inputPanel The input panel.
private	origOut The orig out.
private	out The out.
private final	textPaneOutput The text pane output.

Constructor Summary

public	ButtonActivatePump1ClickedListener(GasPump1 gasPump1, javax.swing.JTextPane textPaneOutput, InputPanel inputPanel) Instantiates a new button activate pump 1 clicked listener.

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	endPrint() Print the printstream into the textPaneOutput.
void	startPrint() Redirect the system output to a prinstream.

java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface java.awt.event.ActionListener
actionPerformed

Fields

gasPump1

private final iit.softwaresystarch.project.model.efsms.GasPump1 **gasPump1**

The gas pump 1.

textPaneOutput

private final javax.swing.JTextPane **textPaneOutput**

The text pane output.

inputPanel

private final iit.softwaresystarch.project.view.InputPanel **inputPanel**

The input panel.

origOut

private java.io.PrintStream **origOut**

The orig out.

baos

private java.io.ByteArrayOutputStream **baos**

The baos.

out

private java.io.PrintStream **out**

The out.

Constructors

(continued on next page)

(continued from last page)

ButtonActivatePump1ClickedListener

```
public ButtonActivatePump1ClickedListener(GasPump1 gasPump1,
                                          javax.swing.JTextPane textPaneOutput,
                                          InputPanel inputPanel)
```

Instantiates a new button activate pump 1 clicked listener.

Parameters:

`gasPump1` - the gas pump 1
`textPaneOutput` - the text pane output
`inputPanel` - the input panel

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

startPrint

```
private void startPrint()
```

Redirect the system output to a prinstream.

endPrint

```
private void endPrint()
```

Print the printstream into the textPaneOutput.

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.controller
Class ButtonActivatePump2ClickedListener

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.controller.ButtonActivatePump2ClickedListener
```

All Implemented Interfaces:
java.awt.event.ActionListener

---

public class **ButtonActivatePump2ClickedListener**
extends java.lang.Object
implements java.awt.event.ActionListener

The listener interface for receiving buttonActivatePump2Clicked events. The class that is interested in processing a buttonActivatePump2Clicked event implements this interface, and the object created with that class is registered with a component using the component's addButtonActivatePump2ClickedListener method. When the buttonActivatePump2Clicked event occurs, that object's appropriate method is invoked.

Field Summary

private	baos The baos.
private final	gasPump2 The gas pump 2.
private final	inputPanel The input panel.
private	origOut The orig out.
private	out The out.
private final	textPaneOutput The text pane output.

Constructor Summary

public	ButtonActivatePump2ClickedListener(GasPump2 gasPump2, javax.swing.JTextPane textPaneOutput, InputPanel inputPanel) Instantiates a new button activate pump 2 clicked listener.

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	endPrint() Print the printstream into the textPaneOutput.
void	startPrint() Redirect the system output to a prinstream.

java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface java.awt.event.ActionListener
actionPerformed

Fields

gasPump2

private final iit.softwaresystarch.project.model.efsms.GasPump2 **gasPump2**

The gas pump 2.

textPaneOutput

private final javax.swing.JTextPane **textPaneOutput**

The text pane output.

inputPanel

private final iit.softwaresystarch.project.view.InputPanel **inputPanel**

The input panel.

origOut

private java.io.PrintStream **origOut**

The orig out.

baos

private java.io.ByteArrayOutputStream **baos**

The baos.

out

private java.io.PrintStream **out**

The out.

Constructors

(continued from last page)

ButtonActivatePump2ClickedListener

```
public ButtonActivatePump2ClickedListener(GasPump2 gasPump2,
                                          javax.swing.JTextPane textPaneOutput,
                                          InputPanel inputPanel)
```

Instantiates a new button activate pump 2 clicked listener.

Parameters:

`gasPump2` - the gas pump 2
`textPaneOutput` - the text pane output
`inputPanel` - the input panel

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

startPrint

```
private void startPrint()
```

Redirect the system output to a prinstream.

endPrint

```
private void endPrint()
```

Print the printstream into the textPaneOutput.

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.controller
Class ButtonCashClickedListener

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.controller.ButtonCashClickedListener
```

All Implemented Interfaces:
java.awt.event.ActionListener

---

public class **ButtonCashClickedListener**
extends java.lang.Object
implements java.awt.event.ActionListener

The listener interface for receiving buttonCashClicked events. The class that is interested in processing a buttonCashClicked event implements this interface, and the object created with that class is registered with a component using the component's addButtonCashClickedListener method. When the buttonCashClicked event occurs, that object's appropriate method is invoked.

Field Summary

private	baos The baos.
private final	gasPump2 The gas pump 2.
private final	inputPanel The input panel.
private	origOut The orig out.
private	out The out.
private final	textPaneOutput The text pane output.

Constructor Summary

public	ButtonCashClickedListener(GasPump2 gasPump2, javax.swing.JTextPane textPaneOutput, InputPanel inputPanel) Instantiates a new button cash clicked listener.

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	endPrint() Print the printstream into the textPaneOutput.
void	startPrint() Redirect the system output to a prinstream.

java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface java.awt.event.ActionListener
actionPerformed

Fields

gasPump2

private final iit.softwaresystarch.project.model.efsms.GasPump2 **gasPump2**

The gas pump 2.

textPaneOutput

private final javax.swing.JTextPane **textPaneOutput**

The text pane output.

inputPanel

private final iit.softwaresystarch.project.view.InputPanel **inputPanel**

The input panel.

origOut

private java.io.PrintStream **origOut**

The orig out.

baos

private java.io.ByteArrayOutputStream **baos**

The baos.

out

private java.io.PrintStream **out**

The out.

Constructors

(continued from last page)

ButtonCashClickedListener

```
public ButtonCashClickedListener(GasPump2 gasPump2,
                                 javax.swing.JTextPane textPaneOutput,
                                 InputPanel inputPanel)
```

Instantiates a new button cash clicked listener.

Parameters:

`gasPump2` - the gas pump 2
`textPaneOutput` - the text pane output
`inputPanel` - the input panel

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

startPrint

```
private void startPrint()
```

Redirect the system output to a prinstream.

endPrint

```
private void endPrint()
```

Print the printstream into the textPaneOutput.

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.controller
Class ButtonClickedListener

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.controller.ButtonClickedListener
```

All Implemented Interfaces:
java.awt.event.ActionListener

---

public class **ButtonClickedListener**
extends java.lang.Object
implements java.awt.event.ActionListener

The listener interface for receiving buttonClicked events. The class that is interested in processing a buttonClicked event implements this interface, and the object created with that class is registered with a component using the component's addButtonClickedListener method. When the buttonClicked event occurs, that object's appropriate method is invoked.

Field Summary

private	baos The baos.
private final	gasPump1 The gas pump 1.
private final	gasPump2 The gas pump 2.
private final	methodString The method string.
private	origOut The orig out.
private	out The out.
private final	textPaneOutput The text pane output.

Constructor Summary

public	ButtonClickedListener(java.lang.Object object, java.lang.String method, javax.swing.JTextPane textPaneOutput) Instantiates a new button clicked listener.

Method Summary

void	actionPerformed(java.awt.event.ActionEvent e)
void	endPrint() Print the printstream into the textPaneOutput.

void	startPrint() Redirect the system output to a prinstream.
java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface java.awt.event.ActionListener
actionPerformed

Fields

gasPump1

`private final iit.softwaresystarch.project.model.efsms.GasPump1` **`gasPump1`**

The gas pump 1.

gasPump2

`private final iit.softwaresystarch.project.model.efsms.GasPump2` **`gasPump2`**

The gas pump 2.

methodString

`private final java.lang.String` **`methodString`**

The method string.

textPaneOutput

`private final javax.swing.JTextPane` **`textPaneOutput`**

The text pane output.

origOut

`private java.io.PrintStream` **`origOut`**

The orig out.

baos

`private java.io.ByteArrayOutputStream` **`baos`**

The baos.

(continued from last page)

out

```
private java.io.PrintStream out
```

The out.

Constructors

ButtonClickedListener

```
public ButtonClickedListener(java.lang.Object object,
                             java.lang.String method,
                             javax.swing.JTextPane textPaneOutput)
```

Instantiates a new button clicked listener.

Parameters:

`object` - the object which can be either a GasPump1 or a GasPump2
`method` - the method to be called on click
`textPaneOutput` - the text pane output

Methods

actionPerformed

```
public void actionPerformed(java.awt.event.ActionEvent e)
```

---

startPrint

```
private void startPrint()
```

Redirect the system output to a prinstream.

---

endPrint

```
private void endPrint()
```

Print the printstream into the textPaneOutput.

---

toString

```
public java.lang.String toString()
```

Package

iit.softwaresystarch.project.model.actions

is a package containing the classes mostly related to the Strategy design pattern.

iit.softwaresystarch.project.model.actions
Class CancelMsg

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.CancelMsg
```

Direct Known Subclasses:
CancelMsg1, CancelMsg2

---

public abstract class **CancelMsg**
extends java.lang.Object

The abstract Class CancelMsg. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method cancelMsg.

Constructor Summary

public	CancelMsg() Instantiates a new CancelMsg.

Method Summary

void	cancelMsg() Displays a cancel message.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

CancelMsg

`public` **`CancelMsg`**`()`

Instantiates a new CancelMsg.

Methods

cancelMsg

`public void` **`cancelMsg`**`()`

Displays a cancel message.

iit.softwaresystarch.project.model.actions
Class CancelMsg1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.CancelMsg
    |-iit.softwaresystarch.project.model.actions.CancelMsg1
```

public class **CancelMsg1**
extends CancelMsg

The Class CancelMsg1. This Class is the implementation of the Interface CancelMsg for the GasPump1.

Constructor Summary

public	CancelMsg1() Instantiates a new CancelMsg1.

Method Summary

void	cancelMsg()

Methods inherited from class iit.softwaresystarch.project.model.actions.CancelMsg
cancelMsg

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

CancelMsg1

public **CancelMsg1**()

Instantiates a new CancelMsg1.

Methods

cancelMsg

public void **cancelMsg**()

Displays a cancel message.

iit.softwaresystarch.project.model.actions
Class CancelMsg2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.CancelMsg
    |-iit.softwaresystarch.project.model.actions.CancelMsg2
```

public class **CancelMsg2**
extends CancelMsg

The Class CancelMsg2. This Class is the implementation of the Interface CancelMsg for the GasPump2.

Constructor Summary

public	CancelMsg2() Instantiates a new CancelMsg2.

Method Summary

void	cancelMsg()

Methods inherited from class iit.softwaresystarch.project.model.actions.CancelMsg

cancelMsg

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

CancelMsg2

public **CancelMsg2**()

Instantiates a new CancelMsg2.

Methods

cancelMsg

public void **cancelMsg**()

Displays a cancel message.

iit.softwaresystarch.project.model.actions
Class DisplayMenu

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.DisplayMenu
```

Direct Known Subclasses:
DisplayMenu1, DisplayMenu2

---

public abstract class **DisplayMenu**
extends java.lang.Object

The anstract Class DisplayMenu. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method displayMenu.

Constructor Summary

public	DisplayMenu() Instantiates a new DisplayMenu.

Method Summary

void	displayMenu() Displays a menu.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

DisplayMenu

`public` **`DisplayMenu`**`()`

Instantiates a new DisplayMenu.

Methods

displayMenu

`public void` **`displayMenu`**`()`

Displays a menu.

iit.softwaresystarch.project.model.actions
Class DisplayMenu1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.DisplayMenu
    |-iit.softwaresystarch.project.model.actions.DisplayMenu1
```

public class **DisplayMenu1**
extends DisplayMenu

The Class DisplayMenu1. This Class is the implementation of the Interface DisplayMenu for the GasPump1.

Constructor Summary

public	DisplayMenu1() Instantiates a new DisplayMenu1.

Method Summary

void	displayMenu()

Methods inherited from class iit.softwaresystarch.project.model.actions.DisplayMenu
displayMenu

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

DisplayMenu1

public **DisplayMenu1**()

Instantiates a new DisplayMenu1.

Methods

displayMenu

public void **displayMenu**()

Displays a menu.

iit.softwaresystarch.project.model.actions
Class DisplayMenu2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.DisplayMenu
    |-iit.softwaresystarch.project.model.actions.DisplayMenu2
```

public class **DisplayMenu2**
extends DisplayMenu

The Class DisplayMenu2. This Class is the implementation of the Interface DisplayMenu for the GasPump2.

Constructor Summary

public	DisplayMenu2() Instantiates a new DisplayMenu2.

Method Summary

void	displayMenu()

Methods inherited from class iit.softwaresystarch.project.model.actions.DisplayMenu
displayMenu

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

DisplayMenu2

public **DisplayMenu2**()

Instantiates a new DisplayMenu2.

Methods

displayMenu

public void **displayMenu**()

Displays a menu.

iit.softwaresystarch.project.model.actions
Class GasPumpedMsg

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.GasPumpedMsg
```

Direct Known Subclasses:
GasPumpedMsg1, GasPumpedMsg2

---

public abstract class **GasPumpedMsg**
extends java.lang.Object

The abstract Class GasPumpedMsg. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method gasPumpedMsg.

Field Summary

protected final	dataStore The data store.

Constructor Summary

public	GasPumpedMsg(DataStore dataStore) Instantiates a new GasPumpedMsg.

Method Summary

void	gasPumpedMsg() Displays a message when gas is pumped.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

dataStore

`protected final iit.softwaresystarch.project.model.datastores.DataStore` **`dataStore`**

The data store.

Constructors

GasPumpedMsg

`public` **`GasPumpedMsg`**`(DataStore dataStore)`

Instantiates a new GasPumpedMsg.

(continued from last page)

Parameters:

`dataStore` - the data store

Methods

gasPumpedMsg

`public void` **`gasPumpedMsg`**`()`

Displays a message when gas is pumped.

iit.softwaresystarch.project.model.actions
Class GasPumpedMsg1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.GasPumpedMsg
    |-iit.softwaresystarch.project.model.actions.GasPumpedMsg1
```

public class **GasPumpedMsg1**
extends GasPumpedMsg

The Class GasPumpedMsg1. This Class is the implementation of the Interface GasPumpedMsg for the GasPump1.

Fields inherited from class `iit.softwaresystarch.project.model.actions.GasPumpedMsg`
`dataStore`

Constructor Summary

public	`GasPumpedMsg1(DataStore1 dataStore)` Instantiates a new GasPumpedMsg1.

Method Summary

void	`gasPumpedMsg()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.GasPumpedMsg`
`gasPumpedMsg`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

GasPumpedMsg1

`public` **`GasPumpedMsg1`**`(DataStore1 dataStore)`

Instantiates a new GasPumpedMsg1.

Parameters:
`dataStore` - the data store

Methods

gasPumpedMsg

`public void` **`gasPumpedMsg`**`()`

(continued from last page)

Displays a message when gas is pumped.

iit.softwaresystarch.project.model.actions
Class GasPumpedMsg2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.GasPumpedMsg
    |-iit.softwaresystarch.project.model.actions.GasPumpedMsg2
```

public class **GasPumpedMsg2**
extends GasPumpedMsg

The Class GasPumpedMsg2. This Class is the implementation of the Interface GasPumpedMsg for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.GasPumpedMsg`
`dataStore`

Constructor Summary

public	`GasPumpedMsg2(DataStore2 dataStore)` Instantiates a new GasPumpedMsg2.

Method Summary

void	`gasPumpedMsg()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.GasPumpedMsg`
`gasPumpedMsg`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

GasPumpedMsg2

`public` **`GasPumpedMsg2`**`(DataStore2 dataStore)`

Instantiates a new GasPumpedMsg2.

Parameters:
`dataStore` - the data store

Methods

gasPumpedMsg

`public void` **`gasPumpedMsg`**`()`

(continued from last page)

Displays a message when gas is pumped.

iit.softwaresystarch.project.model.actions
Class Operation

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.Operation
```

public class **Operation**
extends java.lang.Object

The Class Operation responsible for executing the operations build by the factory.

Field Summary

private final	operationAbstractFactory The operation abstract factory.

Constructor Summary

public	Operation(OperationAbstractFactory operationAbstractFactory) Instantiates a new Operation.

Method Summary

void	cancelMsg() Displays a cancel message if the factory has an implementation.
void	displayMenu() Displays a menu if the factory has an implementation.
void	gasPumpedMsg() Displays a gas pumped message if the factory has an implementation.
void	payMsg() Displays the pay message if the factory has an implementation.
void	printReceipt() Prints the receipt if the factory has an implementation.
void	pumpGasUnit() Pumps a gas unit if the factory has an implementation.
void	readyMsg() Displays a ready message if the factory has an implementation.
void	rejectMsg() Displays a reject message if the factory has an implementation.
void	returnCash() Returns the cash if the factory has an implementation.
void	setInitialValues() Sets the initial values if the factory has an implementation.

void	setPrice(int g) Sets the price if the factory has an implementation.
void	stopMsg() Displays a stop message if the factory has an implementation.
void	storeCash() Stores cash if the factory has an implementation.
void	storeData() Stores data if the factory has an implementation.
java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

operationAbstractFactory

```
private final iit.softwaresystarch.project.model.factories.OperationAbstractFactory
operationAbstractFactory
```

The operation abstract factory.

Constructors

Operation

```
public Operation(OperationAbstractFactory operationAbstractFactory)
```

Instantiates a new Operation.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

storeData

```
public void storeData()
```

Stores data if the factory has an implementation.

payMsg

```
public void payMsg()
```

Displays the pay message if the factory has an implementation.

(continued from last page)

storeCash

`public void **storeCash**()`

Stores cash if the factory has an implementation.

displayMenu

`public void **displayMenu**()`

Displays a menu if the factory has an implementation.

rejectMsg

`public void **rejectMsg**()`

Displays a reject message if the factory has an implementation.

cancelMsg

`public void **cancelMsg**()`

Displays a cancel message if the factory has an implementation.

setPrice

`public void **setPrice**(int g)`

Sets the price if the factory has an implementation.

Parameters:
`g` - the new price

readyMsg

`public void **readyMsg**()`

Displays a ready message if the factory has an implementation.

setInitialValues

`public void **setInitialValues**()`

Sets the initial values if the factory has an implementation.

pumpGasUnit

`public void **pumpGasUnit**()`

Pumps a gas unit if the factory has an implementation.

gasPumpedMsg

`public void **gasPumpedMsg**()`

Displays a gas pumped message if the factory has an implementation.

(continued from last page)

stopMsg

`public void stopMsg()`

Displays a stop message if the factory has an implementation.

printReceipt

`public void printReceipt()`

Prints the receipt if the factory has an implementation.

returnCash

`public void returnCash()`

Returns the cash if the factory has an implementation.

toString

`public java.lang.String toString()`

iit.softwaresystarch.project.model.actions
Class PayMsg

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.PayMsg
```

Direct Known Subclasses:
PayMsg1, PayMsg2

---

public abstract class **PayMsg**
extends java.lang.Object

The abstract Class PayMsg. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method payMsg.

Constructor Summary

public	PayMsg() Instantiates a new PayMsg.

Method Summary

void	payMsg() Displays a pay message.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

PayMsg

`public` **`PayMsg`**`()`

Instantiates a new PayMsg.

Methods

payMsg

`public void` **`payMsg`**`()`

Displays a pay message.

iit.softwaresystarch.project.model.actions
Class PayMsg1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.PayMsg
    |-iit.softwaresystarch.project.model.actions.PayMsg1
```

public class **PayMsg1**
extends PayMsg

The Class PayMsg1. This Class is the implementation of the Interface PayMsg for the GasPump1.

Constructor Summary

public	PayMsg1() Instantiates a new PayMsg1.

Method Summary

void	payMsg()

Methods inherited from class iit.softwaresystarch.project.model.actions.PayMsg
payMsg

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

PayMsg1

public **PayMsg1**()

Instantiates a new PayMsg1.

Methods

payMsg

public void **payMsg**()

Displays a pay message.

iit.softwaresystarch.project.model.actions
Class PayMsg2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.PayMsg
    |-iit.softwaresystarch.project.model.actions.PayMsg2
```

public class **PayMsg2**
extends PayMsg

The Class PayMsg2. This Class is the implementation of the Interface PayMsg for the GasPump2.

Constructor Summary

public	PayMsg2() Instantiates a new PayMsg2.

Method Summary

void	payMsg()

Methods inherited from class iit.softwaresystarch.project.model.actions.PayMsg
payMsg

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

PayMsg2

public **PayMsg2**()

Instantiates a new PayMsg2.

Methods

payMsg

public void **payMsg**()

Displays a pay message.

iit.softwaresystarch.project.model.actions
Class PrintReceipt

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.PrintReceipt
```

Direct Known Subclasses:
PrintReceipt1, PrintReceipt2

---

public abstract class **PrintReceipt**
extends java.lang.Object

The abstract Class PrintReceipt. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method printReceipt.

Field Summary

protected final	dataStore The data store.

Constructor Summary

public	PrintReceipt(DataStore dataStore) Instantiates a new PrintReceipt.

Method Summary

void	printReceipt() Prints the receipt.

Methods inherited from class `java.lang.Object`
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

---

Fields

dataStore

protected final iit.softwaresystarch.project.model.datastores.DataStore **dataStore**

The data store.

Constructors

PrintReceipt

public **PrintReceipt**(DataStore dataStore)

Instantiates a new PrintReceipt.

(continued from last page)

Parameters:

`dataStore` - the data store

Methods

printReceipt

`public void` **`printReceipt`**`()`

Prints the receipt.

iit.softwaresystarch.project.model.actions
Class PrintReceipt1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.PrintReceipt
    |-iit.softwaresystarch.project.model.actions.PrintReceipt1
```

public class **PrintReceipt1**
extends PrintReceipt

The Class PrintReceipt1. This Class is the implementation of the Interface PrintReceipt for the GasPump1.

Fields inherited from class `iit.softwaresystarch.project.model.actions.PrintReceipt`
`dataStore`

Constructor Summary

public	`PrintReceipt1(DataStore1 dataStore)` Instantiates a new PrintReceipt1.

Method Summary

void	`printReceipt()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.PrintReceipt`
`printReceipt`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

PrintReceipt1

`public` **`PrintReceipt1`**`(DataStore1 dataStore)`

Instantiates a new PrintReceipt1.

Parameters:
`dataStore` - the data store

Methods

printReceipt

`public void` **`printReceipt`**`()`

(continued from last page)

Prints the receipt.

iit.softwaresystarch.project.model.actions
Class PrintReceipt2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.PrintReceipt
    |-iit.softwaresystarch.project.model.actions.PrintReceipt2
```

public class **PrintReceipt2**
extends PrintReceipt

The Class PrintReceipt2. This Class is the implementation of the Interface PrintReceipt for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.PrintReceipt`
`dataStore`

Constructor Summary

public	`PrintReceipt2(DataStore2 dataStore)` Instantiates a new PrintReceipt2.

Method Summary

void	`printReceipt()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.PrintReceipt`
`printReceipt`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

PrintReceipt2

`public` **`PrintReceipt2`**`(DataStore2 dataStore)`

Instantiates a new PrintReceipt2.

Parameters:
`dataStore` - the data store

Methods

printReceipt

`public void` **`printReceipt`**`()`

(continued from last page)

Prints the receipt.

iit.softwaresystarch.project.model.actions
Class PumpGasUnit

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.PumpGasUnit
```

Direct Known Subclasses:
PumpGasUnit1, PumpGasUnit2

---

public abstract class **PumpGasUnit**
extends java.lang.Object

The abstract Class PumpGasUnit. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method pumpGasUnit.

Field Summary

protected final	dataStore The data store.

Constructor Summary

public	PumpGasUnit(DataStore dataStore) Instantiates a new PumpGasUnit.

Method Summary

void	pumpGasUnit() Pumps a gas unit.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

dataStore

`protected final iit.softwaresystarch.project.model.datastores.DataStore` **`dataStore`**

The data store.

Constructors

PumpGasUnit

`public` **`PumpGasUnit`**`(DataStore dataStore)`

Instantiates a new PumpGasUnit.

(continued from last page)

Parameters:

`dataStore` - the data store

Methods

pumpGasUnit

`public void` **`pumpGasUnit`**`()`

Pumps a gas unit.

iit.softwaresystarch.project.model.actions
Class PumpGasUnit1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.PumpGasUnit
    |-iit.softwaresystarch.project.model.actions.PumpGasUnit1
```

public class **PumpGasUnit1**
extends PumpGasUnit

The Class PumpGasUnit1. This Class is the implementation of the Interface PumpGasUnit for the GasPump1.

Fields inherited from class `iit.softwaresystarch.project.model.actions.PumpGasUnit`
`dataStore`

Constructor Summary

public	`PumpGasUnit1(DataStore1 dataStore)` Instantiates a new PumpGasUnit1.

Method Summary

void	`pumpGasUnit()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.PumpGasUnit`
`pumpGasUnit`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

PumpGasUnit1

`public` **`PumpGasUnit1`**`(DataStore1 dataStore)`

Instantiates a new PumpGasUnit1.

Parameters:
`dataStore` - the data store

Methods

pumpGasUnit

`public void` **`pumpGasUnit`**`()`

(continued from last page)

Pumps a gas unit.

iit.softwaresystarch.project.model.actions
Class PumpGasUnit2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.PumpGasUnit
    |-iit.softwaresystarch.project.model.actions.PumpGasUnit2
```

public class **PumpGasUnit2**
extends PumpGasUnit

The Class PumpGasUnit2. This Class is the implementation of the Interface PumpGasUnit for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.PumpGasUnit`
`dataStore`

Constructor Summary

public	`PumpGasUnit2(DataStore2 dataStore)` Instantiates a new PumpGasUnit2.

Method Summary

void	`pumpGasUnit()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.PumpGasUnit`
`pumpGasUnit`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

PumpGasUnit2

`public` **`PumpGasUnit2`**`(DataStore2 dataStore)`

Instantiates a new PumpGasUnit2.

Parameters:
`dataStore` - the data store

Methods

pumpGasUnit

`public void` **`pumpGasUnit`**`()`

(continued from last page)

Pumps a gas unit.

iit.softwaresystarch.project.model.actions
Class ReadyMsg

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.ReadyMsg
```

public abstract class **ReadyMsg**
extends java.lang.Object

The abstract Class ReadyMsg. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method readyMsg.

Constructor Summary

public	ReadyMsg() Instantiates a new ReadyMsg.

Method Summary

void	readyMsg() Displays a ready message.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

ReadyMsg

`public` **`ReadyMsg`**`()`

Instantiates a new ReadyMsg.

Methods

readyMsg

`public void` **`readyMsg`**`()`

Displays a ready message.

iit.softwaresystarch.project.model.actions
Class RejectMsg

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.RejectMsg
```

Direct Known Subclasses:
RejectMsg1

---

public abstract class **RejectMsg**
extends java.lang.Object

The abstract Class RejectMsg. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method rejectMsg.

Constructor Summary

public	RejectMsg() Instantiates a new RejectMsg.

Method Summary

void	rejectMsg() Displays a reject message.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

RejectMsg

`public` **`RejectMsg`**`()`

Instantiates a new RejectMsg.

Methods

rejectMsg

`public void` **`rejectMsg`**`()`

Displays a reject message.

iit.softwaresystarch.project.model.actions
Class RejectMsg1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.RejectMsg
    |-iit.softwaresystarch.project.model.actions.RejectMsg1
```

public class **RejectMsg1**
extends RejectMsg

The Class RejectMsg1. This Class is the implementation of the Interface RejectMsg for the GasPump1.

Constructor Summary

public	RejectMsg1() Instantiates a new RejectMsg1.

Method Summary

void	rejectMsg()

Methods inherited from class iit.softwaresystarch.project.model.actions.RejectMsg
rejectMsg

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

RejectMsg1

public **RejectMsg1**()

Instantiates a new RejectMsg1.

Methods

rejectMsg

public void **rejectMsg**()

Displays a reject message.

iit.softwaresystarch.project.model.actions
Class ReturnCash

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.ReturnCash
```

Direct Known Subclasses:
ReturnCash2

public abstract class **ReturnCash**
extends java.lang.Object

The Class ReturnCash. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method returnCash.

Field Summary

protected final	dataStore The data store.

Constructor Summary

public	ReturnCash(DataStore dataStore) Instantiates a new ReturnCash.

Method Summary

void	returnCash() Returns the cash.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

dataStore

`protected final iit.softwaresystarch.project.model.datastores.DataStore` **`dataStore`**

The data store.

Constructors

ReturnCash

`public` **`ReturnCash`**`(DataStore dataStore)`

Instantiates a new ReturnCash.

(continued from last page)

Parameters:

`dataStore` - the data store

Methods

returnCash

```
public void returnCash()
```

Returns the cash.

iit.softwaresystarch.project.model.actions
Class ReturnCash2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.ReturnCash
    |-iit.softwaresystarch.project.model.actions.ReturnCash2
```

public class **ReturnCash2**
extends ReturnCash

The Class ReturnCash2. This Class is the implementation of the Interface ReturnCash for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.ReturnCash`
`dataStore`

Constructor Summary

public	`ReturnCash2(DataStore2 dataStore)` Instantiates a new ReturnCash2.

Method Summary

void	`returnCash()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.ReturnCash`
`returnCash`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

ReturnCash2

`public` **`ReturnCash2`**`(DataStore2 dataStore)`

Instantiates a new ReturnCash2.

Parameters:
`dataStore` - the data store

Methods

returnCash

`public void` **`returnCash`**`()`

(continued from last page)

Returns the cash.

iit.softwaresystarch.project.model.actions
Class SetInitialValues

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.SetInitialValues
```

Direct Known Subclasses:
SetInitialValues1, SetInitialValues2

public abstract class **SetInitialValues**
extends java.lang.Object

The abstract Class SetInitialValues. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method setInitialValues.

Field Summary

protected final	dataStore The data store.

Constructor Summary

public	SetInitialValues(DataStore dataStore) Instantiates a new SetInitialValues.

Method Summary

void	setInitialValues() Sets the initial values.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

dataStore

`protected final iit.softwaresystarch.project.model.datastores.DataStore` **`dataStore`**

The data store.

Constructors

SetInitialValues

`public` **`SetInitialValues`**`(DataStore dataStore)`

Instantiates a new SetInitialValues.

(continued from last page)

Parameters:

`dataStore` - the data store

Methods

setInitialValues

`public void` **`setInitialValues`**`()`

Sets the initial values.

iit.softwaresystarch.project.model.actions
Class SetInitialValues1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.SetInitialValues
    |-iit.softwaresystarch.project.model.actions.SetInitialValues1
```

public class **SetInitialValues1**
extends SetInitialValues

The Class SetInitialValues1. This Class is the implementation of the Interface SetInitialValues for the GasPump1.

Fields inherited from class `iit.softwaresystarch.project.model.actions.SetInitialValues`
`dataStore`

Constructor Summary

public	`SetInitialValues1(DataStore1 dataStore)` Instantiates a new SetInitialValues1.

Method Summary

void	`setInitialValues()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.SetInitialValues`
`setInitialValues`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

SetInitialValues1

`public` **`SetInitialValues1`**`(DataStore1 dataStore)`

Instantiates a new SetInitialValues1.

Parameters:
`dataStore` - the data store

Methods

setInitialValues

`public void` **`setInitialValues`**`()`

(continued from last page)

Sets the initial values.

iit.softwaresystarch.project.model.actions
Class SetInitialValues2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.SetInitialValues
    |-iit.softwaresystarch.project.model.actions.SetInitialValues2
```

public class **SetInitialValues2**
extends SetInitialValues

The Class SetInitialValues2. This Class is the implementation of the Interface SetInitialValues for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.SetInitialValues`
`dataStore`

Constructor Summary

public	`SetInitialValues2(DataStore2 dataStore)` Instantiates a new SetInitialValues2.

Method Summary

void	`setInitialValues()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.SetInitialValues`
`setInitialValues`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

SetInitialValues2

`public` **`SetInitialValues2`**`(DataStore2 dataStore)`

Instantiates a new SetInitialValues2.

Parameters:
`dataStore` - the data store

Methods

setInitialValues

`public void` **`setInitialValues`**`()`

(continued from last page)

Sets the initial values.

iit.softwaresystarch.project.model.actions
Class SetPrice

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.SetPrice
```

Direct Known Subclasses:
SetPrice1, SetPrice2

public abstract class **SetPrice**
extends java.lang.Object

The abstract Class SetPrice. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method setPrice(int g).

Field Summary

protected final	dataStore The data store.

Constructor Summary

public	SetPrice(DataStore dataStore) Instantiates a new SetPrice.

Method Summary

void	setPrice(int g) Sets the price.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

dataStore

`protected final iit.softwaresystarch.project.model.datastores.DataStore` **`dataStore`**

The data store.

Constructors

SetPrice

`public` **`SetPrice`**`(DataStore dataStore)`

Instantiates a new SetPrice.

(continued from last page)

Parameters:
`dataStore` - the data store

Methods

setPrice

```
public void setPrice(int g)
```

Sets the price.

Parameters:
`g` - the new price

iit.softwaresystarch.project.model.actions
Class SetPrice1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.SetPrice
    |-iit.softwaresystarch.project.model.actions.SetPrice1
```

public class **SetPrice1**
extends SetPrice

The Class SetPrice1. This Class is the implementation of the Interface SetPrice for the GasPump1.

Fields inherited from class `iit.softwaresystarch.project.model.actions.SetPrice`
`dataStore`

Constructor Summary

public	`SetPrice1(DataStore1 dataStore)` Instantiates a new SetPrice1.

Method Summary

void	`setPrice(int g)`

Methods inherited from class `iit.softwaresystarch.project.model.actions.SetPrice`
`setPrice`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

SetPrice1

`public SetPrice1(DataStore1 dataStore)`

Instantiates a new SetPrice1.

Parameters:
`dataStore` - the data store

Methods

setPrice

`public void setPrice(int g)`

(continued from last page)

Sets the price.

iit.softwaresystarch.project.model.actions
Class SetPrice2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.SetPrice
    |-iit.softwaresystarch.project.model.actions.SetPrice2
```

public class **SetPrice2**
extends SetPrice

The Class SetPrice2. This Class is the implementation of the Interface SetPrice for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.SetPrice`
`dataStore`

Constructor Summary

public	`SetPrice2(DataStore2 dataStore)` Instantiates a new SetPrice2.

Method Summary

void	`setPrice(int g)`

Methods inherited from class `iit.softwaresystarch.project.model.actions.SetPrice`
`setPrice`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

SetPrice2

`public` **`SetPrice2`**`(DataStore2 dataStore)`

Instantiates a new SetPrice2.

Parameters:
`dataStore` - the data store

Methods

setPrice

`public void` **`setPrice`**`(int g)`

(continued from last page)

Sets the price.

iit.softwaresystarch.project.model.actions
Class StopMsg

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.StopMsg
```

public abstract class **StopMsg**
extends java.lang.Object

The abstract Class StopMsg. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method stopMsg.

Constructor Summary

public	StopMsg() Instantiates a new StopMsg.

Method Summary

void	stopMsg() Displays a stop message.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

StopMsg

`public StopMsg()`

Instantiates a new StopMsg.

Methods

stopMsg

`public void stopMsg()`

Displays a stop message.

iit.softwaresystarch.project.model.actions
Class StoreCash

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.StoreCash
```

Direct Known Subclasses:
StoreCash2

public abstract class **StoreCash**
extends java.lang.Object

The abstract Class StoreCash. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method storeCash.

Field Summary

protected final	dataStore The data store.

Constructor Summary

public	StoreCash(DataStore dataStore) Instantiates a new StoreCash.

Method Summary

void	storeCash() Stores the cash.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

dataStore

`protected final iit.softwaresystarch.project.model.datastores.DataStore` **`dataStore`**

The data store.

Constructors

StoreCash

`public` **`StoreCash`**`(DataStore dataStore)`

Instantiates a new StoreCash.

(continued from last page)

Parameters:

`dataStore` - the data store

Methods

storeCash

```
public void storeCash()
```

Stores the cash.

iit.softwaresystarch.project.model.actions
Class StoreCash2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.StoreCash
    |-iit.softwaresystarch.project.model.actions.StoreCash2
```

public class **StoreCash2**
extends StoreCash

The Class StoreCash2. This Class is the implementation of the Interface StoreCash for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.StoreCash`
`dataStore`

Constructor Summary

public	`StoreCash2(DataStore2 dataStore)` Instantiates a new StoreCash2.

Method Summary

void	`storeCash()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.StoreCash`
`storeCash`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

StoreCash2

`public StoreCash2(DataStore2 dataStore)`

Instantiates a new StoreCash2.

Parameters:
`dataStore` - the data store

Methods

storeCash

`public void storeCash()`

(continued from last page)

Stores the cash.

iit.softwaresystarch.project.model.actions
Class StoreData

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.actions.StoreData
```

Direct Known Subclasses:
StoreData1, StoreData2

public abstract class **StoreData**
extends java.lang.Object

The abstract Class StoreData. This Class is used as an Interface for the Strategy pattern. The goal is to have multiple implemtation of the method storeData.

Field Summary

protected	dataStore The data store.

Constructor Summary

public	StoreData(DataStore dataStore) Instantiates a new StoreData.

Method Summary

void	storeData() Stores the data.

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

dataStore

`protected iit.softwaresystarch.project.model.datastores.DataStore` **`dataStore`**

The data store.

Constructors

StoreData

`public` **`StoreData`**`(DataStore dataStore)`

Instantiates a new StoreData.

(continued from last page)

Parameters:

`dataStore` - the data store

Methods

storeData

`public void` **`storeData`**`()`

Stores the data.

iit.softwaresystarch.project.model.actions
Class StoreData1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.StoreData
    |-iit.softwaresystarch.project.model.actions.StoreData1
```

public class **StoreData1**
extends StoreData

The Class StoreData1. This Class is the implementation of the Interface StoreData for the GasPump1.

Fields inherited from class `iit.softwaresystarch.project.model.actions.StoreData`
`dataStore`

Constructor Summary

public	`StoreData1(DataStore1 dataStore)` Instantiates a new StoreData1.

Method Summary

void	`storeData()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.StoreData`
`storeData`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

StoreData1

`public StoreData1(DataStore1 dataStore)`

Instantiates a new StoreData1.

Parameters:
`dataStore` - the data store

Methods

storeData

`public void storeData()`

(continued from last page)

Stores the data.

iit.softwaresystarch.project.model.actions
Class StoreData2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.actions.StoreData
    |-iit.softwaresystarch.project.model.actions.StoreData2
```

public class **StoreData2**
extends StoreData

The Class StoreData2. This Class is the implementation of the Interface StoreData for the GasPump2.

Fields inherited from class `iit.softwaresystarch.project.model.actions.StoreData`
`dataStore`

Constructor Summary

public	`StoreData2(DataStore2 dataStore)` Instantiates a new StoreData2.

Method Summary

void	`storeData()`

Methods inherited from class `iit.softwaresystarch.project.model.actions.StoreData`
`storeData`

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Constructors

StoreData2

`public` **`StoreData2`**`(DataStore2 dataStore)`

Instantiates a new StoreData2.

Parameters:
`dataStore` - the data store

Methods

storeData

`public void` **`storeData`**`()`

(continued from last page)

Stores the data.

Package

iit.softwaresystarch.project.model.datastores

is a package containing the classes mostly related to both AbstractFactory and Strategy design patterns.

iit.softwaresystarch.project.model.datastores
Class DataStore

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.datastores.DataStore
```

Direct Known Subclasses:
DataStore1, DataStore2

public class **DataStore**
extends java.lang.Object

The Class DataStore. This Class is responsible for stocking the main common attributes of Gas Pumps.

Field Summary

protected	price The price.
protected	total The total.

Constructor Summary

public	DataStore() Instantiates a new data store.

Method Summary

float	getPrice() Gets the price.
float	getTotal() Gets the total.
void	setPrice(float price) Sets the price.
void	setTotal(float total) Sets the total.
java.lang.String	toString()

Methods inherited from class `java.lang.Object`

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

(continued from last page)

price

`protected float price`

The price.

total

`protected float total`

The total.

Constructors

DataStore

`public DataStore()`

Instantiates a new data store.

Methods

getPrice

`public float getPrice()`

Gets the price.

Returns:
the price

setPrice

`public void setPrice(float price)`

Sets the price.

Parameters:
`price` - the new price

getTotal

`public float getTotal()`

Gets the total.

Returns:
the total

setTotal

`public void setTotal(float total)`

Sets the total.

Parameters:
`total` - the new total

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.model.datastores
Class DataStore1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.datastores.DataStore
    |-iit.softwaresystarch.project.model.datastores.DataStore1
```

public class **DataStore1**
extends DataStore

The Class DataStore1. This Class is responsible for stocking the specifics attributes of GasPump1.

Field Summary

private	gallon The gallon pumped amount.
private	rPrice The regular price.
private	sPrice The suoer price.
private	temp_a The temp a.
private	temp_b The temp b.

Fields inherited from class iit.softwaresystarch.project.model.datastores.DataStore

price, total

Constructor Summary

public	DataStore1() Instantiates a new DataStore1.

Method Summary

float	getGallon() Gets the gallon pumped amount.
float	getRPrice() Gets the regular price.
float	getSPrice() Gets the super price.
float	getTemp_a() Gets the temp a.

float	getTemp_b() Gets the temp b.
void	setGallon(float gallon) Sets the gallon pumped amount.
void	setRPrice(float rPrice) Sets the reguler price.
void	setSPrice(float sPrice) Sets the super price.
void	setTemp_a(float temp_a) Sets the temp a.
void	setTemp_b(float temp_b) Sets the temp b.
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.datastores.DataStore
getPrice, getTotal, setPrice, setTotal, toString

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

temp_a

private float **temp_a**

The temp a.

temp_b

private float **temp_b**

The temp b.

rPrice

private float **rPrice**

The regular price.

sPrice

private float **sPrice**

The suoer price.

(continued from last page)

gallon

`private float gallon`

The gallon pumped amount.

Constructors

DataStore1

`public DataStore1()`

Instantiates a new DataStore1.

Methods

getTemp_a

`public float getTemp_a()`

Gets the temp a.

Returns:
the temp a

---

setTemp_a

`public void setTemp_a(float temp_a)`

Sets the temp a.

Parameters:
`temp_a` - the new temp a

---

getTemp_b

`public float getTemp_b()`

Gets the temp b.

Returns:
the temp b

---

setTemp_b

`public void setTemp_b(float temp_b)`

Sets the temp b.

Parameters:
`temp_b` - the new temp b

---

getRPrice

`public float getRPrice()`

Gets the regular price.

(continued from last page)

Returns:
the regular price

setRPrice

`public void setRPrice(float rPrice)`

Sets the reguler price.

Parameters:
`rPrice` - the new regular price

getSPrice

`public float getSPrice()`

Gets the super price.

Returns:
the super price

setSPrice

`public void setSPrice(float sPrice)`

Sets the super price.

Parameters:
`sPrice` - the new super price

getGallon

`public float getGallon()`

Gets the gallon pumped amount.

Returns:
the gallon pumped amount

setGallon

`public void setGallon(float gallon)`

Sets the gallon pumped amount.

Parameters:
`gallon` - the new gallon pumped amount

toString

`public java.lang.String toString()`

iit.softwaresystarch.project.model.datastores
Class DataStore2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.datastores.DataStore
    |-iit.softwaresystarch.project.model.datastores.DataStore2
```

public class **DataStore2**
extends DataStore

The Class DataStore2. This Class is responsible for stocking the specifics attributes of GasPump2.

Field Summary

private	cash The cash available.
private	liter The liter pumped amount.
private	pPrice The premium price.
private	rPrice The regular price.
private	sPrice The super price.
private	temp_a The temp a.
private	temp_b The temp b.
private	temp_c The temp c.
private	temp_cash The temp cash.

Fields inherited from class iit.softwaresystarch.project.model.datastores.DataStore
price, total

Constructor Summary

public	DataStore2() Instantiates a new data store 2.

Method Summary

float	getCash() Gets the cash available.
float	getLiter() Gets the liter pumped amount.
int	getPPrice() Gets the premium price.
int	getRPrice() Gets the regular price.
int	getSPrice() Gets the super price.
int	getTemp_a() Gets the temp a.
int	getTemp_b() Gets the temp b.
int	getTemp_c() Gets the temp c.
float	getTemp_cash() Gets the temp cash.
void	setCash(float cash) Sets the cash available.
void	setLiter(float liter) Sets the liter pumped amount.
void	setPPrice(int pPrice) Sets the premium price.
void	setRPrice(int rPrice) Sets the regular price.
void	setSPrice(int sPrice) Sets the super price.
void	setTemp_a(int temp_a) Sets the temp a.
void	setTemp_b(int temp_b) Sets the temp b.
void	setTemp_c(int temp_c) Sets the temp c.
void	setTemp_cash(float temp_cash) Sets the temp cash.
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.datastores.DataStore
getPrice, getTotal, setPrice, setTotal, toString

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

temp_a

`private int` **`temp_a`**

The temp a.

temp_b

`private int` **`temp_b`**

The temp b.

temp_c

`private int` **`temp_c`**

The temp c.

temp_cash

`private float` **`temp_cash`**

The temp cash.

rPrice

`private int` **`rPrice`**

The regular price.

sPrice

`private int` **`sPrice`**

The super price.

pPrice

`private int` **`pPrice`**

The premium price.

liter

`private float` **`liter`**

The liter pumped amount.

cash

`private float cash`

The cash available.

Constructors

DataStore2

`public DataStore2()`

Instantiates a new data store 2.

Methods

getTemp_a

`public int getTemp_a()`

Gets the temp a.

Returns:
the temp a

setTemp_a

`public void setTemp_a(int temp_a)`

Sets the temp a.

Parameters:
`temp_a` - the new temp a

getTemp_b

`public int getTemp_b()`

Gets the temp b.

Returns:
the temp b

setTemp_b

`public void setTemp_b(int temp_b)`

Sets the temp b.

Parameters:
`temp_b` - the new temp b

getTemp_c

`public int getTemp_c()`

Gets the temp c.

(continued from last page)

Returns:
the temp c

setTemp_c

`public void setTemp_c(int temp_c)`

Sets the temp c.

Parameters:
`temp_c` - the new temp c

getTemp_cash

`public float getTemp_cash()`

Gets the temp cash.

Returns:
the temp cash

setTemp_cash

`public void setTemp_cash(float temp_cash)`

Sets the temp cash.

Parameters:
`temp_cash` - the new temp cash

getRPrice

`public int getRPrice()`

Gets the regular price.

Returns:
the regular price

setRPrice

`public void setRPrice(int rPrice)`

Sets the regular price.

Parameters:
`rPrice` - the new regular price

getSPrice

`public int getSPrice()`

Gets the super price.

Returns:
the super price

setSPrice

`public void **setSPrice**(int sPrice)`

Sets the super price.

Parameters:
`sPrice` - the new super price

getPPrice

`public int **getPPrice**()`

Gets the premium price.

Returns:
the premium price

setPPrice

`public void **setPPrice**(int pPrice)`

Sets the premium price.

Parameters:
`pPrice` - the new premium price

getLiter

`public float **getLiter**()`

Gets the liter pumped amount.

Returns:
the liter pumped amount

setLiter

`public void **setLiter**(float liter)`

Sets the liter pumped amount.

Parameters:
`liter` - the new liter pumped amount

getCash

`public float **getCash**()`

Gets the cash available.

Returns:
the cash available

setCash

`public void **setCash**(float cash)`

(continued from last page)

Sets the cash available.

Parameters:

`cash` - the new cash available

toString

`public java.lang.String` **`toString`**`()`

Package
iit.softwaresystarch.project.model.efsms

is a package containing the classes mostly related to Model Driven Architecture.

iit.softwaresystarch.project.model.efsms
Class GasPump1

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.efsms.GasPump1
```

public class **GasPump1**
extends java.lang.Object

The Class GasPump1. This Class is the concrete representation of a GasPump1.

Field Summary

private final	dataStore The data store.
private final	mdaEfsm The mda efsm.

Constructor Summary

public	GasPump1() Instantiates a new GasPump1.

Method Summary

void	activate(float a, float b) Activate.
void	approved() Approved.
void	cancel() Cancel.
void	payCredit() Pay credit.
void	pumpGallon() Pump gallon.
void	regular() Regular.
void	reject() Reject.
void	start() Start.
void	startPump() Start pump.

void	stopPump() Stop pump.
void	superGas() Super gas.
java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

dataStore

```
private final iit.softwaresystarch.project.model.datastores.DataStore dataStore
```

The data store.

mdaEfsm

```
private final iit.softwaresystarch.project.model.efsms.MDAEFSM mdaEfsm
```

The mda efsm.

Constructors

GasPump1

```
public GasPump1()
```

Instantiates a new GasPump1.

Methods

activate

```
public void activate(float a,
         float b)
```

Activate.

Parameters:
a - the a
b - the b

start

```
public void start()
```

Start.

(continued from last page)

payCredit

`public void **payCredit**()`

Pay credit.

approved

`public void **approved**()`

Approved.

reject

`public void **reject**()`

Reject.

cancel

`public void **cancel**()`

Cancel.

regular

`public void **regular**()`

Regular.

superGas

`public void **superGas**()`

Super gas.

startPump

`public void **startPump**()`

Start pump.

pumpGallon

`public void **pumpGallon**()`

Pump gallon.

stopPump

`public void **stopPump**()`

Stop pump.

toString

`public java.lang.String **toString**()`

(continued from last page)

iit.softwaresystarch.project.model.efsms
Class GasPump2

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.efsms.GasPump2
```

public class **GasPump2**
extends java.lang.Object

The Class GasPump2. This Class is the concrete representation of a GasPump2.

Field Summary

private final	dataStore The data store.
private final	mdaEfsm The mda efsm.

Constructor Summary

public	GasPump2() Instantiates a new GasPump2.

Method Summary

void	activate(int a, int b, int c) Activate.
void	cancel() Cancel.
void	noReceipt() No receipt.
void	payCash(float c) Pay cash.
void	premium() Premium.
void	pumpLiter() Pump liter.
void	receipt() Receipt.
void	regular() Regular.
void	start() Start.

void	startPump() Start pump.
void	stop() Stop.
void	superGas() Super gas.
java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

dataStore

```
private final iit.softwaresystarch.project.model.datastores.DataStore dataStore
```

The data store.

mdaEfsm

```
private final iit.softwaresystarch.project.model.efsms.MDAEFSM mdaEfsm
```

The mda efsm.

Constructors

GasPump2

```
public GasPump2()
```

Instantiates a new GasPump2.

Methods

activate

```
public void activate(int a,
         int b,
         int c)
```

Activate.

Parameters:

a - the a
b - the b
c - the c

(continued from last page)

start

`public void **start**()`

Start.

payCash

`public void **payCash**(float c)`

Pay cash.

Parameters:
`c` - the c

cancel

`public void **cancel**()`

Cancel.

superGas

`public void **superGas**()`

Super gas.

regular

`public void **regular**()`

Regular.

premium

`public void **premium**()`

Premium.

startPump

`public void **startPump**()`

Start pump.

pumpLiter

`public void **pumpLiter**()`

Pump liter.

stop

`public void **stop**()`

Stop.

(continued from last page)

receipt

`public void receipt()`

Receipt.

noReceipt

`public void noReceipt()`

No receipt.

toString

`public java.lang.String toString()`

iit.softwaresystarch.project.model.efsms
Class MDAEFSM

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.efsms.MDAEFSM
```

public class **MDAEFSM**
extends java.lang.Object

The Class MDAEFSM.

Field Summary

private	currentState The current state.
private final	operationAbstractFactory The operation abstract factory.
private final	statesList The states list.

Constructor Summary

public	MDAEFSM(OperationAbstractFactory operationAbstractFactory) Instantiates a new MDAEFSM and instantiates .

Method Summary

void	activate() Activate.
void	approved() Approved.
void	cancel() Cancel.
State	getCurrentState() Gets the current state.
java.util.List	getStatesList() Gets the states list.
void	noReceipt() No receipt.
void	payType(int t) Pay type.
void	pump() Pump.

void	receipt() Receipt.
void	reject() Reject.
void	selectGas(int g) Select gas.
void	setCurrentState(State currentState) Sets the current state.
void	start() Start.
void	startPump() Start pump.
void	stopPump() Stop pump.
java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

statesList

private final java.util.List **statesList**

The states list.

currentState

private iit.softwaresystarch.project.model.states.State **currentState**

The current state.

operationAbstractFactory

private final iit.softwaresystarch.project.model.factories.OperationAbstractFactory **operationAbstractFactory**

The operation abstract factory.

Constructors

MDAEFSM

public **MDAEFSM**(OperationAbstractFactory operationAbstractFactory)

Instantiates a new MDAEFSM and instantiates .

(continued from last page)

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

activate

`public void **activate**()`

Activate.

start

`public void **start**()`

Start.

payType

`public void **payType**(int t)`

Pay type.

Parameters:
`t` - the t

approved

`public void **approved**()`

Approved.

reject

`public void **reject**()`

Reject.

cancel

`public void **cancel**()`

Cancel.

selectGas

`public void **selectGas**(int g)`

Select gas.

Parameters:
`g` - the g

startPump

`public void **startPump**()`

(continued from last page)

Start pump.

pump

`public void pump()`

Pump.

stopPump

`public void stopPump()`

Stop pump.

receipt

`public void receipt()`

Receipt.

noReceipt

`public void noReceipt()`

No receipt.

getStatesList

`public java.util.List getStatesList()`

Gets the states list.

Returns:
the states list

getCurrentState

`public State getCurrentState()`

Gets the current state.

Returns:
the current state

setCurrentState

`public void setCurrentState(State currentState)`

Sets the current state.

Parameters:
`currentState` - the new current state

toString

`public java.lang.String toString()`

Package

iit.softwaresystarch.project.model.factories

is a package containing the classes mostly related to AbstractFactory design pattern.

iit.softwaresystarch.project.model.factories
Class OperationAbstractFactory

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.factories.OperationAbstractFactory
```

Direct Known Subclasses:
OperationAbstractFactory1, OperationAbstractFactory2

---

public class **OperationAbstractFactory**
extends java.lang.Object

A factory for creating Operation objects. This Class is designed as an Interface for the concrete factories that will builds all the differents instances used by the operation class to make actions on the system.

Field Summary

protected	cancelMsg The cancel msg.
protected	dataStore The data store.
protected	displayMenu The display menu.
protected	gasPumpedMsg The gas pumped msg.
protected	payMsg The pay msg.
protected	printReceipt The print receipt.
protected	pumpGasUnit The pump gas unit.
protected	readyMsg The ready msg.
protected	rejectMsg The reject msg.
protected	returnCash The return cash.
protected	setInitialValues The set initial values.
protected	setPrice The set price.
protected	stopMsg The stop msg.

protected	storeCash The store cash.
protected	storeData The store data.

Constructor Summary

public	OperationAbstractFactory() Instantiates a new OperationAbstractFactory.

Method Summary

CancelMsg	getCancelMsg() Gets the cancel msg.
DataStore	getDataStore() Gets the data store.
DisplayMenu	getDisplayMenu() Gets the display menu.
GasPumpedMsg	getGasPumpedMsg() Gets the gas pumped msg.
PayMsg	getPayMsg() Gets the pay msg.
PrintReceipt	getPrintReceipt() Gets the prints the receipt.
PumpGasUnit	getPumpGasUnit() Gets the pump gas unit.
ReadyMsg	getReadyMsg() Gets the ready msg.
RejectMsg	getRejectMsg() Gets the reject msg.
ReturnCash	getReturnCash() Gets the return cash.
SetInitialValues	getSetInitialValues() Gets the sets the initial values.
SetPrice	getSetPrice(int g) Gets the sets the price.
StopMsg	getStopMsg() Gets the stop msg.
StoreCash	getStoreCash() Gets the store cash.
StoreData	getStoreData() Gets the store data.

java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

dataStore

protected iit.softwaresystarch.project.model.datastores.DataStore **dataStore**

The data store.

storeData

protected iit.softwaresystarch.project.model.actions.StoreData **storeData**

The store data.

displayMenu

protected iit.softwaresystarch.project.model.actions.DisplayMenu **displayMenu**

The display menu.

payMsg

protected iit.softwaresystarch.project.model.actions.PayMsg **payMsg**

The pay msg.

storeCash

protected iit.softwaresystarch.project.model.actions.StoreCash **storeCash**

The store cash.

cancelMsg

protected iit.softwaresystarch.project.model.actions.CancelMsg **cancelMsg**

The cancel msg.

rejectMsg

protected iit.softwaresystarch.project.model.actions.RejectMsg **rejectMsg**

The reject msg.

setPrice

protected iit.softwaresystarch.project.model.actions.SetPrice **setPrice**

(continued from last page)

The set price.

readyMsg

`protected iit.softwaresystarch.project.model.actions.ReadyMsg` **`readyMsg`**

The ready msg.

setInitialValues

`protected iit.softwaresystarch.project.model.actions.SetInitialValues` **`setInitialValues`**

The set initial values.

pumpGasUnit

`protected iit.softwaresystarch.project.model.actions.PumpGasUnit` **`pumpGasUnit`**

The pump gas unit.

gasPumpedMsg

`protected iit.softwaresystarch.project.model.actions.GasPumpedMsg` **`gasPumpedMsg`**

The gas pumped msg.

stopMsg

`protected iit.softwaresystarch.project.model.actions.StopMsg` **`stopMsg`**

The stop msg.

printReceipt

`protected iit.softwaresystarch.project.model.actions.PrintReceipt` **`printReceipt`**

The print receipt.

returnCash

`protected iit.softwaresystarch.project.model.actions.ReturnCash` **`returnCash`**

The return cash.

Constructors

OperationAbstractFactory

`public` **`OperationAbstractFactory`**`()`

Instantiates a new OperationAbstractFactory.

Methods

getDataStore

`public DataStore` **`getDataStore`**`()`

(continued from last page)

Gets the data store.

Returns:
the data store

getStoreData

`public StoreData getStoreData()`

Gets the store data.

Returns:
the store data

getPayMsg

`public PayMsg getPayMsg()`

Gets the pay msg.

Returns:
the pay msg

getStoreCash

`public StoreCash getStoreCash()`

Gets the store cash.

Returns:
the store cash

getCancelMsg

`public CancelMsg getCancelMsg()`

Gets the cancel msg.

Returns:
the cancel msg

getDisplayMenu

`public DisplayMenu getDisplayMenu()`

Gets the display menu.

Returns:
the display menu

getRejectMsg

`public RejectMsg getRejectMsg()`

Gets the reject msg.

Returns:
the reject msg

getSetPrice

`public SetPrice getSetPrice(int g)`

Gets the sets the price.

Parameters:
g - the g

Returns:
the sets the price

getReadyMsg

`public ReadyMsg getReadyMsg()`

Gets the ready msg.

Returns:
the ready msg

getSetInitialValues

`public SetInitialValues getSetInitialValues()`

Gets the sets the initial values.

Returns:
the sets the initial values

getPumpGasUnit

`public PumpGasUnit getPumpGasUnit()`

Gets the pump gas unit.

Returns:
the pump gas unit

getGasPumpedMsg

`public GasPumpedMsg getGasPumpedMsg()`

Gets the gas pumped msg.

Returns:
the gas pumped msg

getStopMsg

`public StopMsg getStopMsg()`

Gets the stop msg.

Returns:
the stop msg

getPrintReceipt

`public PrintReceipt getPrintReceipt()`

Gets the prints the receipt.

Returns:
the prints the receipt

getReturnCash

`public ReturnCash getReturnCash()`

Gets the return cash.

Returns:
the return cash

toString

`public java.lang.String toString()`

iit.softwaresystarch.project.model.factories
Class OperationAbstractFactory1

```
java.lang.Object
  |-iit.softwaresystarch.project.model.factories.OperationAbstractFactory
    |-iit.softwaresystarch.project.model.factories.OperationAbstractFactory1
```

public class **OperationAbstractFactory1**
extends OperationAbstractFactory

The Class OperationAbstractFactory1.

Fields inherited from class iit.softwaresystarch.project.model.factories.OperationAbstractFactory
cancelMsg, dataStore, displayMenu, gasPumpedMsg, payMsg, printReceipt, pumpGasUnit, readyMsg, rejectMsg, returnCash, setInitialValues, setPrice, stopMsg, storeCash, storeData

Constructor Summary

public	OperationAbstractFactory1() Instantiates a new OperationAbstractFactory1 and Instantiates the requiered attributes.

Method Summary

CancelMsg	getCancelMsg()
DisplayMenu	getDisplayMenu()
GasPumpedMsg	getGasPumpedMsg()
PayMsg	getPayMsg()
PrintReceipt	getPrintReceipt()
PumpGasUnit	getPumpGasUnit()
ReadyMsg	getReadyMsg()
RejectMsg	getRejectMsg()
ReturnCash	getReturnCash()
SetInitialValues	getSetInitialValues()
SetPrice	getSetPrice(int g)
StopMsg	getStopMsg()

StoreCash	getStoreCash()
StoreData	getStoreData()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.factories.OperationAbstractFactory

getCancelMsg, getDataStore, getDisplayMenu, getGasPumpedMsg, getPayMsg, getPrintReceipt, getPumpGasUnit, getReadyMsg, getRejectMsg, getReturnCash, getSetInitialValues, getSetPrice, getStopMsg, getStoreCash, getStoreData, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

OperationAbstractFactory1

public **OperationAbstractFactory1**()

Instantiates a new OperationAbstractFactory1 and Instantiates the requiered attributes.

Methods

getStoreData

public StoreData **getStoreData**()

Gets the store data.

getPayMsg

public PayMsg **getPayMsg**()

Gets the pay msg.

getStoreCash

public StoreCash **getStoreCash**()

Gets the store cash.

getDisplayMenu

public DisplayMenu **getDisplayMenu**()

Gets the display menu.

(continued from last page)

getRejectMsg

`public RejectMsg getRejectMsg()`

Gets the reject msg.

getCancelMsg

`public CancelMsg getCancelMsg()`

Gets the cancel msg.

getSetPrice

`public SetPrice getSetPrice(int g)`

Gets the sets the price.

getReadyMsg

`public ReadyMsg getReadyMsg()`

Gets the ready msg.

getSetInitialValues

`public SetInitialValues getSetInitialValues()`

Gets the sets the initial values.

getPumpGasUnit

`public PumpGasUnit getPumpGasUnit()`

Gets the pump gas unit.

getGasPumpedMsg

`public GasPumpedMsg getGasPumpedMsg()`

Gets the gas pumped msg.

getStopMsg

`public StopMsg getStopMsg()`

Gets the stop msg.

getPrintReceipt

`public PrintReceipt getPrintReceipt()`

Gets the prints the receipt.

getReturnCash

`public ReturnCash getReturnCash()`

(continued from last page)

Gets the return cash.

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.model.factories
Class OperationAbstractFactory2

```
java.lang.Object
  |-iit.softwaresystarch.project.model.factories.OperationAbstractFactory
    |-iit.softwaresystarch.project.model.factories.OperationAbstractFactory2
```

public class **OperationAbstractFactory2**
extends OperationAbstractFactory

The Class OperationAbstractFactory2.

Fields inherited from class iit.softwaresystarch.project.model.factories.OperationAbstractFactory
cancelMsg, dataStore, displayMenu, gasPumpedMsg, payMsg, printReceipt, pumpGasUnit, readyMsg, rejectMsg, returnCash, setInitialValues, setPrice, stopMsg, storeCash, storeData

Constructor Summary

public	OperationAbstractFactory2() Instantiates a new OperationAbstractFactory2 and Instantiates the requiered attributes.

Method Summary

CancelMsg	getCancelMsg()
DisplayMenu	getDisplayMenu()
GasPumpedMsg	getGasPumpedMsg()
PayMsg	getPayMsg()
PrintReceipt	getPrintReceipt()
PumpGasUnit	getPumpGasUnit()
ReadyMsg	getReadyMsg()
RejectMsg	getRejectMsg()
ReturnCash	getReturnCash()
SetInitialValues	getSetInitialValues()
SetPrice	getSetPrice(int g)
StopMsg	getStopMsg()

StoreCash	getStoreCash()
StoreData	getStoreData()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.factories.OperationAbstractFactory

getCancelMsg, getDataStore, getDisplayMenu, getGasPumpedMsg, getPayMsg, getPrintReceipt, getPumpGasUnit, getReadyMsg, getRejectMsg, getReturnCash, getSetInitialValues, getSetPrice, getStopMsg, getStoreCash, getStoreData, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Constructors

OperationAbstractFactory2

public **OperationAbstractFactory2**()

Instantiates a new OperationAbstractFactory2 and Instantiates the requiered attributes.

Methods

getStoreData

public StoreData **getStoreData**()

Gets the store data.

getPayMsg

public PayMsg **getPayMsg**()

Gets the pay msg.

getCancelMsg

public CancelMsg **getCancelMsg**()

Gets the cancel msg.

getStoreCash

public StoreCash **getStoreCash**()

Gets the store cash.

(continued from last page)

getDisplayMenu

`public DisplayMenu getDisplayMenu()`

Gets the display menu.

getRejectMsg

`public RejectMsg getRejectMsg()`

Gets the reject msg.

getSetPrice

`public SetPrice getSetPrice(int g)`

Gets the sets the price.

getReadyMsg

`public ReadyMsg getReadyMsg()`

Gets the ready msg.

getSetInitialValues

`public SetInitialValues getSetInitialValues()`

Gets the sets the initial values.

getPumpGasUnit

`public PumpGasUnit getPumpGasUnit()`

Gets the pump gas unit.

getGasPumpedMsg

`public GasPumpedMsg getGasPumpedMsg()`

Gets the gas pumped msg.

getStopMsg

`public StopMsg getStopMsg()`

Gets the stop msg.

getPrintReceipt

`public PrintReceipt getPrintReceipt()`

Gets the prints the receipt.

getReturnCash

`public ReturnCash getReturnCash()`

(continued from last page)

Gets the return cash.

toString

```
public java.lang.String toString()
```

Package
iit.softwaresystarch.project.model.states

is a package containing the classes mostly related to State design pattern.

iit.softwaresystarch.project.model.states
Class Start

```
java.lang.Object
  +-iit.softwaresystarch.project.model.states.State
    +-iit.softwaresystarch.project.model.states.Start
```

public class **Start**
extends State

The Class Start.

Field Summary

private static final	ID The Constant ID. Value: **-1**

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	Start(OperationAbstractFactory operationAbstractFactory) Instantiates a new Start.

Method Summary

void	activate()
int	getID()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

```
private static final int ID
```

The Constant ID.
Constant value: **-1**

Constructors

Start

```
public Start(OperationAbstractFactory operationAbstractFactory)
```

Instantiates a new Start.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

```
public int getID()
```

Gets the id.

---

activate

```
public void activate()
```

Activate.

---

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.model.states
Class State

```
java.lang.Object
  |
  +-iit.softwaresystarch.project.model.states.State
```

Direct Known Subclasses:
Start, State0, State1, State2, State3, State4, State5, State6

public abstract class **State**
extends java.lang.Object

The abstract Class State.

Field Summary

private static final	ID The Constant ID. Value: **-2**
protected	operation The operation.

Constructor Summary

public	State(OperationAbstractFactory operationAbstractFactory) Instantiates a new State.

Method Summary

void	activate() Activate.
void	approved() Approved.
void	cancel() Cancel.
int	getID() Gets the id.
void	noReceipt() No receipt.
void	payType(int t) Pay type.
void	pump() Pump.
void	receipt() Receipt.

void	reject() Reject.
void	selectGas(int g) Select gas.
void	start() Start.
void	startPump() Start pump.
void	stopPump() Stop pump.
java.lang.String	toString()

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

ID

`private static final int ID`

The Constant ID.
Constant value: **-2**

operation

`protected iit.softwaresystarch.project.model.actions.Operation operation`

The operation.

Constructors

State

`public State(OperationAbstractFactory operationAbstractFactory)`

Instantiates a new State.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

`public int getID()`

Gets the id.

(continued from last page)

Returns:
the id

activate

`public void activate()`

Activate.

start

`public void start()`

Start.

payType

`public void payType(int t)`

Pay type.

Parameters:
`t` - the payment type id

approved

`public void approved()`

Approved.

reject

`public void reject()`

Reject.

cancel

`public void cancel()`

Cancel.

selectGas

`public void selectGas(int g)`

Select gas.

Parameters:
`g` - the gas id

startPump

`public void startPump()`

Start pump.

pump

`public void **pump**()`

Pump.

stopPump

`public void **stopPump**()`

Stop pump.

receipt

`public void **receipt**()`

Receipt.

noReceipt

`public void **noReceipt**()`

No receipt.

toString

`public java.lang.String **toString**()`

iit.softwaresystarch.project.model.states
Class State0

```
java.lang.Object
  |-iit.softwaresystarch.project.model.states.State
    |-iit.softwaresystarch.project.model.states.State0
```

public class **State0**
extends State

The Class State0.

Field Summary

private static final	ID The Constant ID. Value: **0**

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	State0(OperationAbstractFactory operationAbstractFactory) Instantiates a new State0.

Method Summary

int	getID()
void	start()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

`private static final int ID`

The Constant ID.
Constant value: **0**

Constructors

State0

`public State0(OperationAbstractFactory operationAbstractFactory)`

Instantiates a new State0.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

`public int getID()`

Gets the id.

---

start

`public void start()`

Start.

---

toString

`public java.lang.String toString()`

iit.softwaresystarch.project.model.states
Class State1

```
java.lang.Object
  +-iit.softwaresystarch.project.model.states.State
    +-iit.softwaresystarch.project.model.states.State1
```

public class **State1**
extends State

The Class State1.

Field Summary

private static final	ID The Constant ID. Value: **1**

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	State1(OperationAbstractFactory operationAbstractFactory) Instantiates a new State1.

Method Summary

int	getID()
void	payType(int t)
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

```
private static final int ID
```

The Constant ID.
Constant value: **1**

Constructors

State1

```
public State1(OperationAbstractFactory operationAbstractFactory)
```

Instantiates a new State1.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

```
public int getID()
```

Gets the id.

---

payType

```
public void payType(int t)
```

Pay type.

---

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.model.states
Class State2

```
java.lang.Object
  +-iit.softwaresystarch.project.model.states.State
    +-iit.softwaresystarch.project.model.states.State2
```

public class **State2**
extends State

The Class State2.

Field Summary

private static final	ID The Constant ID. Value: **2**

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	State2(OperationAbstractFactory operationAbstractFactory) Instantiates a new State2.

Method Summary

void	approved()
int	getID()
void	reject()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

`private static final int ID`

The Constant ID.
Constant value: **2**

Constructors

State2

`public State2(OperationAbstractFactory operationAbstractFactory)`

Instantiates a new State2.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

`public int getID()`

Gets the id.

---

approved

`public void approved()`

Approved.

---

reject

`public void reject()`

Reject.

---

toString

`public java.lang.String toString()`

iit.softwaresystarch.project.model.states
Class State3

```
java.lang.Object
  +-iit.softwaresystarch.project.model.states.State
    +-iit.softwaresystarch.project.model.states.State3
```

public class **State3**
extends State

The Class State3.

Field Summary

private static final	ID The Constant ID. Value: **3**

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	State3(OperationAbstractFactory operationAbstractFactory) Instantiates a new State3.

Method Summary

void	cancel()
int	getID()
void	selectGas(int g)
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

```
private static final int ID
```

The Constant ID.
Constant value: **3**

Constructors

State3

```
public State3(OperationAbstractFactory operationAbstractFactory)
```

Instantiates a new State3.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

```
public int getID()
```

Gets the id.

---

cancel

```
public void cancel()
```

Cancel.

---

selectGas

```
public void selectGas(int g)
```

Select gas.

---

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.model.states
Class State4

```
java.lang.Object
  |-iit.softwaresystarch.project.model.states.State
    |-iit.softwaresystarch.project.model.states.State4
```

public class **State4**
extends State

The Class State4.

Field Summary

private static final	ID The Constant ID. Value: **4**

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	State4(OperationAbstractFactory operationAbstractFactory) Instantiates a new State4.

Method Summary

int	getID()
void	startPump()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

```
private static final int ID
```

The Constant ID.
Constant value: **4**

Constructors

State4

```
public State4(OperationAbstractFactory operationAbstractFactory)
```

Instantiates a new State4.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

```
public int getID()
```

Gets the id.

---

startPump

```
public void startPump()
```

Start pump.

---

toString

```
public java.lang.String toString()
```

iit.softwaresystarch.project.model.states
Class State5

```
java.lang.Object
  +-iit.softwaresystarch.project.model.states.State
    +-iit.softwaresystarch.project.model.states.State5
```

public class **State5**
extends State

The Class State5.

Field Summary

private static final	ID The Constant ID. Value: **5**

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	State5(OperationAbstractFactory operationAbstractFactory) Instantiates a new State5.

Method Summary

int	getID()
void	pump()
void	stopPump()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

`private static final int ID`

The Constant ID.
Constant value: **5**

Constructors

State5

`public State5(OperationAbstractFactory operationAbstractFactory)`

Instantiates a new State5.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

`public int getID()`

Gets the id.

---

pump

`public void pump()`

Pump.

---

stopPump

`public void stopPump()`

Stop pump.

---

toString

`public java.lang.String toString()`

iit.softwaresystarch.project.model.states
Class State6

```
java.lang.Object
  |-iit.softwaresystarch.project.model.states.State
    |-iit.softwaresystarch.project.model.states.State6
```

public class **State6**
extends State

The Class State6.

Field Summary

private static	ID The id.

Fields inherited from class iit.softwaresystarch.project.model.states.State

ID, operation

Constructor Summary

public	State6(OperationAbstractFactory operationAbstractFactory) Instantiates a new State6.

Method Summary

int	getID()
void	noReceipt()
void	receipt()
java.lang.String	toString()

Methods inherited from class iit.softwaresystarch.project.model.states.State

activate, approved, cancel, getID, noReceipt, payType, pump, receipt, reject, selectGas, start, startPump, stopPump, toString

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Fields

(continued from last page)

ID

`private static int ID`

The id.

Constructors

State6

`public State6(OperationAbstractFactory operationAbstractFactory)`

Instantiates a new State6.

Parameters:
`operationAbstractFactory` - the operation abstract factory

Methods

getID

`public int getID()`

Gets the id.

---

receipt

`public void receipt()`

Receipt.

---

noReceipt

`public void noReceipt()`

No receipt.

---

toString

`public java.lang.String toString()`

Package
iit.softwaresystarch.project.view

is a package containing the classes mostly related to View design pattern.

iit.softwaresystarch.project.view
Class Driver

```
java.lang.Object
   |
   +-iit.softwaresystarch.project.view.Driver
```

public class **Driver**
extends java.lang.Object

The Class Driver.

Field Summary

private	baos The baos.
private	frame The frame.
private	gasPump1 The gas pump 1.
private	gasPump1Panel The gas pump 1 panel.
private	gasPump2 The gas pump 2.
private	gasPump2Panel The gas pump 2 panel.
private	inputPanel The input panel.
private	origOut The orig out.
private	out The out.
private	textPaneOutput The text pane output.

Constructor Summary

public	Driver() Instantiates a new Driver.

Method Summary

void	endPrint() Print the printstream into the textPaneOutput.

void	initialize() Initialize the driver.
static void	main(java.lang.String[] args) The main method to create the view.
void	rebootDriver() Reboot the driver.
void	startPrint() Redirect the system output to a prinstream.

Methods inherited from class `java.lang.Object`
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait`

Fields

frame

`private javax.swing.JFrame` **`frame`**

The frame.

gasPump1Panel

`private iit.softwaresystarch.project.view.GasPump1Panel` **`gasPump1Panel`**

The gas pump 1 panel.

gasPump2Panel

`private iit.softwaresystarch.project.view.GasPump2Panel` **`gasPump2Panel`**

The gas pump 2 panel.

gasPump1

`private iit.softwaresystarch.project.model.efsms.GasPump1` **`gasPump1`**

The gas pump 1.

gasPump2

`private iit.softwaresystarch.project.model.efsms.GasPump2` **`gasPump2`**

The gas pump 2.

textPaneOutput

`private javax.swing.JTextPane` **`textPaneOutput`**

The text pane output.

(continued from last page)

inputPanel

`private iit.softwaresystarch.project.view.InputPanel` **`inputPanel`**

The input panel.

origOut

`private java.io.PrintStream` **`origOut`**

The orig out.

baos

`private java.io.ByteArrayOutputStream` **`baos`**

The baos.

out

`private java.io.PrintStream` **`out`**

The out.

Constructors

Driver

`public` **`Driver`**`()`

Instantiates a new Driver.

Methods

main

`public static void` **`main`**`(java.lang.String[] args)`

The main method to create the view.

Parameters:
`args` - the arguments

initialize

`private void` **`initialize`**`()`

Initialize the driver.

startPrint

`private void` **`startPrint`**`()`

Redirect the system output to a prinstream.

(continued from last page)

endPrint

`private void **endPrint**()`

Print the printstream into the textPaneOutput.

rebootDriver

`private void **rebootDriver**()`

Reboot the driver.

iit.softwaresystarch.project.view
Class GasPump1Panel

```
java.lang.Object
  ┣-java.awt.Component
    ┣-java.awt.Container
      ┣-javax.swing.JComponent
        ┣-javax.swing.JPanel
          ┣-iit.softwaresystarch.project.view.GasPump1Panel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.swing.TransferHandler.HasGetTransferHandler, java.io.Serializable, javax.accessibility.Accessible

---

public class **GasPump1Panel**
extends javax.swing.JPanel

The Class GasPump1Panel.

Field Summary

private final	buttonActivate The button activate.
private final	buttonApproved The button approved.
private final	buttonCancel The button cancel.
private final	buttonPayCredit The button pay credit.
private final	buttonPumpGallon The button pump gallon.
private final	buttonRegular The button regular.
private final	buttonReject The button reject.
private final	buttonStart The button start.
private final	buttonStartPump The button start pump.
private final	buttonStopPump The button stop pump.
private final	buttonSuper The button super.

private final	gasPump1 The gas pump 1.
private final	inputPanel The input panel.
private final	labelGasPump1 The label gas pump 1.
private static final	serialVersionUID The Constant serialVersionUID. Value: **1**
private final	textPaneOutput The text pane output.

Fields inherited from class `javax.swing.JPanel`

`uiClassID`

Fields inherited from class `javax.swing.JComponent`

`aaTextInfo`, `actionMap`, `ACTIONMAP_CREATED`, `alignmentX`, `alignmentY`, `ANCESTOR_INPUTMAP_CREATED`, `ANCESTOR_USING_BUFFER`, `ancestorInputMap`, `autoscrolls`, `AUTOSCROLLS_SET`, `border`, `clientProperties`, `COMPLETELY_OBSCURED`, `componentObtainingGraphicsFrom`, `componentObtainingGraphicsFromLock`, `CREATED_DOUBLE_BUFFER`, `DEBUG_GRAPHICS_LOADED`, `defaultLocale`, `flags`, `FOCUS_INPUTMAP_CREATED`, `FOCUS_TRAVERSAL_KEYS_BACKWARD_SET`, `FOCUS_TRAVERSAL_KEYS_FORWARD_SET`, `focusController`, `focusInputMap`, `INHERITS_POPUP_MENU`, `INPUT_VERIFIER_SOURCE_KEY`, `inputVerifier`, `IS_DOUBLE_BUFFERED`, `IS_OPAQUE`, `IS_PAINTING_TILE`, `IS_PRINTING`, `IS_PRINTING_ALL`, `IS_REPAINTING`, `isAlignmentXSet`, `isAlignmentYSet`, `KEY_EVENTS_ENABLED`, `KEYBOARD_BINDINGS_KEY`, `listenerList`, `managingFocusBackwardTraversalKeys`, `managingFocusForwardTraversalKeys`, `NEXT_FOCUS`, `NOT_OBSCURED`, `OPAQUE_SET`, `paintingChild`, `PARTIALLY_OBSCURED`, `popupMenu`, `readObjectCallbacks`, `REQUEST_FOCUS_DISABLED`, `RESERVED_1`, `RESERVED_2`, `RESERVED_3`, `RESERVED_4`, `RESERVED_5`, `RESERVED_6`, `revalidateRunnableScheduled`, `tempRectangles`, `TOOL_TIP_TEXT_KEY`, `ui`, `uiClassID`, `UNDEFINED_CONDITION`, `verifyInputWhenFocusTarget`, `vetoableChangeSupport`, `WHEN_ANCESTOR_OF_FOCUSED_COMPONENT`, `WHEN_FOCUSED`, `WHEN_IN_FOCUSED_WINDOW`, `WHEN_IN_FOCUSED_WINDOW_BINDINGS`, `WIF_INPUTMAP_CREATED`, `windowInputMap`, `WRITE_OBJ_COUNTER_FIRST`, `WRITE_OBJ_COUNTER_LAST`

Fields inherited from class `java.awt.Container`

`component`, `containerListener`, `containerSerializedDataVersion`, `descendantsCount`, `descendUnconditionallyWhenValidating`, `dispatcher`, `EMPTY_ARRAY`, `eventLog`, `focusCycleRoot`, `focusTraversalPolicy`, `focusTraversalPolicyProvider`, `INCLUDE_SELF`, `isJavaAwtSmartInvalidate`, `layoutMgr`, `listeningBoundsChildren`, `listeningChildren`, `log`, `mixingLog`, `modalAppContext`, `modalComp`, `numOfHWComponents`, `numOfLWComponents`, `preserveBackgroundColor`, `printing`, `printingThreads`, `SEARCH_HEAVYWEIGHTS`, `serialPersistentFields`, `serialVersionUID`

Fields inherited from class `java.awt.Component`

acc, accessibleContext, actionListenerK, adjustmentListenerK, appContext, autoFocusTransferOnDisposal, background, backgroundEraseDisabled, BOTTOM_ALIGNMENT, boundsOp, bufferStrategy, CENTER_ALIGNMENT, changeSupport, coalesceEventsParams, coalesceMap, coalescingEnabled, componentListener, componentListenerK, componentOrientation, componentSerializedDataVersion, compoundShape, containerListenerK, cursor, dropTarget, enabled, eventCache, eventLog, eventMask, FOCUS_TRAVERSABLE_DEFAULT, FOCUS_TRAVERSABLE_SET, FOCUS_TRAVERSABLE_UNKNOWN, focusable, focusListener, focusListenerK, focusLog, focusTraversalKeyPropertyNames, focusTraversalKeys, focusTraversalKeysEnabled, font, foreground, graphicsConfig, height, hierarchyBoundsListener, hierarchyBoundsListenerK, hierarchyListener, hierarchyListenerK, ignoreRepaint, incRate, inputMethodListener, inputMethodListenerK, isAddNotifyComplete, isFocusTraversableOverridden, isInc, isPacked, itemListenerK, keyListener, keyListenerK, LEFT_ALIGNMENT, locale, LOCK, log, maxSize, maxSizeSet, minSize, minSizeSet, mixingCutoutRegion, mixingLog, mouseListener, mouseListenerK, mouseMotionListener, mouseMotionListenerK, mouseWheelListener, mouseWheelListenerK, name, nameExplicitlySet, newEventsOnly, objectLock, ownedWindowK, parent, peer, peerFont, popups, prefSize, prefSizeSet, requestFocusController, RIGHT_ALIGNMENT, serialVersionUID, textListenerK, TOP_ALIGNMENT, valid, visible, width, windowClosingException, windowFocusListenerK, windowListenerK, windowStateListenerK, x, y

Fields inherited from interface `java.awt.image.ImageObserver`

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

public	`GasPump1Panel(GasPump1 gasPump1, javax.swing.JTextPane textPaneOutput, InputPanel inputPanel)` Instantiates a new GasPump1Panel, add the buttons and controllers.

Method Summary

void	`addControllers()` Adds the controllers.
javax.swing.JButton	`getButtonActivate()` Gets the button activate.
javax.swing.JButton	`getButtonApproved()` Gets the button approved.
javax.swing.JButton	`getButtonCancel()` Gets the button cancel.
javax.swing.JButton	`getButtonPayCredit()` Gets the button pay credit.
javax.swing.JButton	`getButtonPumpGallon()` Gets the button pump gallon.
javax.swing.JButton	`getButtonRegular()` Gets the button regular.
javax.swing.JButton	`getButtonReject()` Gets the button reject.
javax.swing.JButton	`getButtonStart()` Gets the button start.

javax.swing.JButton	getButtonStartPump() Gets the button start pump.
javax.swing.JButton	getButtonStopPump() Gets the button stop pump.
javax.swing.JButton	getButtonSuper() Gets the button super.
GasPump1	getGasPump1() Gets the gas pump 1.
javax.swing.JLabel	getLabelGasPump1() Gets the label gas pump 1.

Methods inherited from class `javax.swing.JPanel`

`getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI, writeObject`

Methods inherited from class `javax.swing.JComponent`

_paintImmediately, addAncestorListener, addNotify, addVetoableChangeListener, adjustPaintFlags, alwaysOnTop, checkIfChildObscuredBySibling, clientPropertyChanged, componentInputMapChanged, computeVisibleRect, computeVisibleRect, compWriteObjectNotify, contains, createToolTip, deregisterNextFocusableComponent, disable, dndDone, dropLocationForPoint, enable, fetchRectangle, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAncestorNotifier, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperties, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getCreatedDoubleBuffer, getDebugGraphicsOptions, getDefaultLocale, getFlag, getFontMetrics, getGraphics, getGraphicsInvoked, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getManagingFocusBackwardTraversalKeys, getManagingFocusForwardTraversalKeys, getMaximumSize, getMinimumSize, getNextFocusableComponent, getObscuredState, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getUIClassID, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getWriteObjCounter, getX, getY, grabFocus, hide, isComponentObtainingGraphicsFrom, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPainting, isPaintingForPrint, isPaintingOrigin, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintForceDoubleBuffered, paintImmediately, paintImmediately, paintToOffscreen, paramString, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyBindings, processKeyBindingsForAllComponents, processKeyEvent, processMouseEvent, processMouseMotionEvent, putClientProperty, readObject, rectangleIsObscured, recycleRectangle, registerKeyboardAction, registerKeyboardAction, registerNextFocusableComponent, registerNextFocusableComponent, registerWithKeyboardManager, registerWithKeyboardManager, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, safelyGetGraphics, safelyGetGraphics, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setCreatedDoubleBuffer, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setDropLocation, setEnabled, setFlag, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPaintingChild, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setUIProperty, setVerifyInputWhenFocusTarget, setVisible, setWriteObjCounter, shouldDebugGraphics, superProcessMouseMotionEvent, uninstallUIAndProperties, unregisterKeyboardAction, unregisterWithKeyboardManager, unregisterWithKeyboardManager, update, updateUI, writeObject

Methods inherited from class `java.awt.Container`

add, add, add, add, add, addContainerListener, addDelicately, addImpl, addNotify, addPropertyChangeListener, addPropertyChangeListener, adjustDecendantsOnParent, adjustDescendants, adjustListeningChildren, applyComponentOrientation, areFocusTraversalKeysSet, canContainFocusOwner, checkAdding, checkAddToSelf, checkGD, checkNotAWindow, clearCurrentFocusCycleRootOnHide, clearMostRecentFocusOwnerOnHide, containsFocus, countComponents, countHierarchyMembers, createChildHierarchyEvents, createHierarchyEvents, decreaseComponentCount, deliverEvent, dispatchEventImpl, dispatchEventToSelf, doLayout, eventEnabled, findComponentAt, findComponentAt, findComponentAt, findComponentAtImpl, findTraversalRoot, getAccessibleAt, getAccessibleChild, getAccessibleChildrenCount, getAlignmentX, getAlignmentY, getBottommostComponentIndex, getChildAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents_NoClientCode, getComponents, getComponentsSync, getComponentZOrder, getContainerListeners, getDropTargetEventTarget, getFocusTraversalKeys, getFocusTraversalPolicy, getHeavyweightContainer, getInsets, getLayout, getListeners, getListenersCount, getMaximumSize, getMinimumSize, getMouseEventTarget, getMouseEventTarget, getMouseEventTargetImpl, getMousePosition, getOpaqueShape, getPreferredSize, getTopmostComponentIndex, getTraversalRoot, hasHeavyweightDescendants, hasLightweightDescendants, increaseComponentCount, initializeFocusTraversalKeys, initIDs, insets, invalidate, invalidateParent, invalidateTree, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, isParentOf, isRecursivelyVisibleUpToHeavyweightContainer, isRemoveNotifyNeeded, isSameOrAncestorOf, isValidateRoot, layout, lightweightPaint, lightweightPrint, list, list, locate, minimumSize, mixOnHiding, mixOnReshaping, mixOnShowing, mixOnValidating, mixOnZOrderChanging, numListening, paint, paintComponents, paintHeavyweightComponents, paramString, postProcessKeyEvent, postsOldMouseEvents, preferredSize, preProcessKeyEvent, print, printComponents, printHeavyweightComponents, processContainerEvent, processEvent, proxyEnableEvents, readObject, recursiveApplyCurrentShape, recursiveApplyCurrentShape, recursiveApplyCurrentShape, recursiveHideHeavyweightChildren, recursiveRelocateHeavyweightChildren, recursiveShowHeavyweightChildren, recursiveSubtractAndApplyShape, recursiveSubtractAndApplyShape, recursiveSubtractAndApplyShape, remove, remove, removeAll, removeContainerListener, removeDelicately, removeNotify, reparentChild, reparentTraverse, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, setLayout, startLWModal, stopLWModal, transferFocusDownCycle, update, updateGraphicsData, validate, validateTree, validateUnconditionally, writeObject

Methods inherited from class `java.awt.Component`

```
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener,
addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener,
addMouseMotionListener, addMouseWheelListener, addNotify, addPropertyChangeListener,
addPropertyChangeListener, adjustListeningChildrenOnParent,
applyComponentOrientation, applyCompoundShape, applyCurrentShape,
applyCurrentShapeBelowMe, areBoundsValid, areFocusTraversalKeysSet,
areInputMethodsEnabled, autoProcessMouseWheel, bounds, calculateCurrentShape,
canBeFocusOwner, canBeFocusOwnerRecursively, checkCoalescing, checkGD, checkImage,
checkImage, checkTreeLock, checkWindowClosingException,
clearCurrentFocusCycleRootOnHide, clearMostRecentFocusOwnerOnHide, coalesceEvents,
constructComponentName, contains, contains, containsFocus, countHierarchyMembers,
createBufferStrategy, createBufferStrategy, createHierarchyEvents, createImage,
createImage, createVolatileImage, createVolatileImage, deliverEvent, disable,
disableEvents, dispatchEvent, dispatchEventImpl, dispatchMouseWheelToAncestor,
doLayout, doSwingSerialization, enable, enable, enableEvents, enableInputMethods,
eventEnabled, eventTypeEnabled, findUnderMouseInWindow, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
getAccessControlContext, getAccessibleContext, getAccessibleIndexInParent,
getAccessibleStateSet, getAlignmentX, getAlignmentY, getAppliedShape, getBackBuffer,
getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds,
getBoundsOp, getBufferStrategy, getColorModel, getComponentAt, getComponentAt,
getComponentListeners, getComponentOrientation, getContainer, getContainingWindow,
getCursor_NoClientCode, getCursor, getDropTarget, getFocusCycleRootAncestor,
getFocusListeners, getFocusTraversalKeys_NoIDCheck, getFocusTraversalKeys,
getFocusTraversalKeysEnabled, getFont_NoClientCode, getFont, getFontMetrics,
getForeground, getGraphics_NoClientCode, getGraphics,
getGraphicsConfiguration_NoClientCode, getGraphicsConfiguration, getHeight,
getHierarchyBoundsListeners, getHierarchyListeners, getHWPeerAboveMe,
getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests,
getInsets_NoClientCode, getKeyListeners, getListeners, getLocale, getLocation,
getLocation, getLocationOnScreen_NoTreeLock, getLocationOnScreen,
getLocationOnWindow, getMaximumSize, getMinimumSize, getMouseListeners,
getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName,
getNativeContainer, getNextFocusCandidate, getNormalShape, getObjectLock,
getOpaqueShape, getParent_NoClientCode, getParent, getPeer, getPreferredSize,
getPropertyChangeListeners, getPropertyChangeListeners, getRecursivelyVisibleBounds,
getSiblingIndexAbove, getSiblingIndexBelow, getSize, getSize, getToolkit,
getToolkitImpl, getTraversalRoot, getTreeLock, getWidth, getX, getY, gotFocus,
handleEvent, hasFocus, hide, imageUpdate, initializeFocusTraversalKeys, initIDs,
inside, invalidate, invalidateIfValid, invalidateParent,
isAutoFocusTransferOnDisposal, isBackgroundSet, isCoalesceEventsOverriden,
isCoalescingEnabled, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled,
isEnabledImpl, isFocusable, isFocusCycleRoot, isFocusOwner, isFocusTraversable,
isFocusTraversableOverridden, isFontSet, isForegroundSet, isInstanceOf,
isLightweight, isMaximumSizeSet, isMinimumSizeSet, isMixingNeeded,
isNonOpaqueForMixing, isOpaque, isPreferredSizeSet, isRecursivelyVisible,
isRequestFocusAccepted, isSameOrAncestorOf, isShowing, isValid,
isVisible_NoClientCode, isVisible, keyDown, keyUp, layout, lightweightPaint,
lightweightPrint, list, list, list, list, list, locate, location_NoClientCode,
location, lostFocus, minimumSize, mixOnHiding, mixOnReshaping, mixOnShowing,
mixOnValidating, mixOnZOrderChanging, mouseDown, mouseDrag, mouseEnter, mouseExit,
mouseMove, mouseUp, move, nextFocus, notifyNewBounds, numListening, paint, paintAll,
paintHeavyweightComponents, paramString, pointRelativeToComponent, postEvent,
postsOldMouseEvents, preferredSize, prepareImage, prepareImage, print, printAll,
printHeavyweightComponents, processComponentEvent, processEvent, processFocusEvent,
processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent,
```

processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, readObject, relocateComponent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, repaintParentIfNeeded, requestFocus, requestFocus, requestFocus, requestFocus, requestFocusHelper, requestFocusHelper, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, reshape, reshapeNativePeer, resize, resize, revalidate, revalidateSynchronously, setAutoFocusTransferOnDisposal, setBackground, setBounds, setBounds, setBoundsOp, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeys_NoIDCheck, setFocusTraversalKeys, setFocusTraversalKeysEnabled, setFont, setForeground, setGraphicsConfiguration, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setRequestFocusController, setSize, setSize, setVisible, show, show, size, subtractAndApplyShape, subtractAndApplyShapeBelowMe, toString, transferFocus, transferFocus, transferFocusBackward, transferFocusBackward, transferFocusUpCycle, update, updateCursorImmediately, updateGraphicsData, updateZOrder, validate, writeObject

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface `java.awt.image.ImageObserver`

imageUpdate

Methods inherited from interface `java.awt.MenuContainer`

getFont, postEvent, remove

Methods inherited from interface `javax.swing.TransferHandler.HasGetTransferHandler`

getTransferHandler

Methods inherited from interface `javax.accessibility.Accessible`

getAccessibleContext

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

The Constant serialVersionUID.
Constant value: **1**

gasPump1

`private final iit.softwaresystarch.project.model.efsms.GasPump1` **`gasPump1`**

The gas pump 1.

labelGasPump1

`private final javax.swing.JLabel` **`labelGasPump1`**

The label gas pump 1.

buttonActivate

`private final javax.swing.JButton` **`buttonActivate`**

The button activate.

buttonStart

`private final javax.swing.JButton` **`buttonStart`**

The button start.

buttonPayCredit

`private final javax.swing.JButton` **`buttonPayCredit`**

The button pay credit.

buttonApproved

`private final javax.swing.JButton` **`buttonApproved`**

The button approved.

buttonReject

`private final javax.swing.JButton` **`buttonReject`**

The button reject.

buttonRegular

`private final javax.swing.JButton` **`buttonRegular`**

The button regular.

buttonSuper

`private final javax.swing.JButton` **`buttonSuper`**

The button super.

buttonStartPump

`private final javax.swing.JButton` **`buttonStartPump`**

The button start pump.

buttonPumpGallon

`private final javax.swing.JButton` **`buttonPumpGallon`**

(continued from last page)

The button pump gallon.

buttonStopPump

```
private final javax.swing.JButton buttonStopPump
```

The button stop pump.

buttonCancel

```
private final javax.swing.JButton buttonCancel
```

The button cancel.

textPaneOutput

```
private final javax.swing.JTextPane textPaneOutput
```

The text pane output.

inputPanel

```
private final iit.softwaresystarch.project.view.InputPanel inputPanel
```

The input panel.

Constructors

GasPump1Panel

```
public GasPump1Panel(GasPump1 gasPump1,
                     javax.swing.JTextPane textPaneOutput,
                     InputPanel inputPanel)
```

Instantiates a new GasPump1Panel, add the buttons and controllers.

Parameters:

`gasPump1` - the gas pump 1
`textPaneOutput` - the text pane output
`inputPanel` - the input panel

Methods

addControllers

```
private void addControllers()
```

Adds the controllers.

getGasPump1

```
public GasPump1 getGasPump1()
```

Gets the gas pump 1.

Returns:

the gas pump 1

getLabelGasPump1

`public javax.swing.JLabel getLabelGasPump1()`

Gets the label gas pump 1.

Returns:
the label gas pump 1

getButtonActivate

`public javax.swing.JButton getButtonActivate()`

Gets the button activate.

Returns:
the button activate

getButtonStart

`public javax.swing.JButton getButtonStart()`

Gets the button start.

Returns:
the button start

getButtonPayCredit

`public javax.swing.JButton getButtonPayCredit()`

Gets the button pay credit.

Returns:
the button pay credit

getButtonApproved

`public javax.swing.JButton getButtonApproved()`

Gets the button approved.

Returns:
the button approved

getButtonReject

`public javax.swing.JButton getButtonReject()`

Gets the button reject.

Returns:
the button reject

getButtonRegular

`public javax.swing.JButton getButtonRegular()`

(continued from last page)

Gets the button regular.

Returns:
the button regular

getButtonSuper

`public javax.swing.JButton getButtonSuper()`

Gets the button super.

Returns:
the button super

getButtonStartPump

`public javax.swing.JButton getButtonStartPump()`

Gets the button start pump.

Returns:
the button start pump

getButtonPumpGallon

`public javax.swing.JButton getButtonPumpGallon()`

Gets the button pump gallon.

Returns:
the button pump gallon

getButtonStopPump

`public javax.swing.JButton getButtonStopPump()`

Gets the button stop pump.

Returns:
the button stop pump

getButtonCancel

`public javax.swing.JButton getButtonCancel()`

Gets the button cancel.

Returns:
the button cancel

iit.softwaresystarch.project.view
Class GasPump2Panel

```
java.lang.Object
  +-java.awt.Component
    +-java.awt.Container
      +-javax.swing.JComponent
        +-javax.swing.JPanel
          +-iit.softwaresystarch.project.view.GasPump2Panel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.swing.TransferHandler.HasGetTransferHandler, java.io.Serializable, javax.accessibility.Accessible

---

public class **GasPump2Panel**
extends javax.swing.JPanel

The Class GasPump2Panel.

Field Summary

private final	buttonActivate The button activate.
private final	buttonCancel The button cancel.
private final	buttonNoReceipt The button no receipt.
private final	buttonPayCash The button pay cash.
private final	buttonPremium The button premium.
private final	buttonPumpLiter The button pump liter.
private final	buttonReceipt The button receipt.
private final	buttonRegular The button regular.
private final	buttonStart The button start.
private final	buttonStartPump The button start pump.
private final	buttonStopPump The button stop pump.

private final	buttonSuper The button super.
private final	gasPump2 The gas pump 2.
private final	inputPanel The input panel.
private final	labelGasPump2 The label gas pump 2.
private static final	serialVersionUID The Constant serialVersionUID. Value: **1**
private final	textPaneOutput The text pane output.

Fields inherited from class `javax.swing.JPanel`

`uiClassID`

Fields inherited from class `javax.swing.JComponent`

`aaTextInfo`, `actionMap`, `ACTIONMAP_CREATED`, `alignmentX`, `alignmentY`, `ANCESTOR_INPUTMAP_CREATED`, `ANCESTOR_USING_BUFFER`, `ancestorInputMap`, `autoscrolls`, `AUTOSCROLLS_SET`, `border`, `clientProperties`, `COMPLETELY_OBSCURED`, `componentObtainingGraphicsFrom`, `componentObtainingGraphicsFromLock`, `CREATED_DOUBLE_BUFFER`, `DEBUG_GRAPHICS_LOADED`, `defaultLocale`, `flags`, `FOCUS_INPUTMAP_CREATED`, `FOCUS_TRAVERSAL_KEYS_BACKWARD_SET`, `FOCUS_TRAVERSAL_KEYS_FORWARD_SET`, `focusController`, `focusInputMap`, `INHERITS_POPUP_MENU`, `INPUT_VERIFIER_SOURCE_KEY`, `inputVerifier`, `IS_DOUBLE_BUFFERED`, `IS_OPAQUE`, `IS_PAINTING_TILE`, `IS_PRINTING`, `IS_PRINTING_ALL`, `IS_REPAINTING`, `isAlignmentXSet`, `isAlignmentYSet`, `KEY_EVENTS_ENABLED`, `KEYBOARD_BINDINGS_KEY`, `listenerList`, `managingFocusBackwardTraversalKeys`, `managingFocusForwardTraversalKeys`, `NEXT_FOCUS`, `NOT_OBSCURED`, `OPAQUE_SET`, `paintingChild`, `PARTIALLY_OBSCURED`, `popupMenu`, `readObjectCallbacks`, `REQUEST_FOCUS_DISABLED`, `RESERVED_1`, `RESERVED_2`, `RESERVED_3`, `RESERVED_4`, `RESERVED_5`, `RESERVED_6`, `revalidateRunnableScheduled`, `tempRectangles`, `TOOL_TIP_TEXT_KEY`, `ui`, `uiClassID`, `UNDEFINED_CONDITION`, `verifyInputWhenFocusTarget`, `vetoableChangeSupport`, `WHEN_ANCESTOR_OF_FOCUSED_COMPONENT`, `WHEN_FOCUSED`, `WHEN_IN_FOCUSED_WINDOW`, `WHEN_IN_FOCUSED_WINDOW_BINDINGS`, `WIF_INPUTMAP_CREATED`, `windowInputMap`, `WRITE_OBJ_COUNTER_FIRST`, `WRITE_OBJ_COUNTER_LAST`

Fields inherited from class `java.awt.Container`

`component`, `containerListener`, `containerSerializedDataVersion`, `descendantsCount`, `descendUnconditionallyWhenValidating`, `dispatcher`, `EMPTY_ARRAY`, `eventLog`, `focusCycleRoot`, `focusTraversalPolicy`, `focusTraversalPolicyProvider`, `INCLUDE_SELF`, `isJavaAwtSmartInvalidate`, `layoutMgr`, `listeningBoundsChildren`, `listeningChildren`, `log`, `mixingLog`, `modalAppContext`, `modalComp`, `numOfHWComponents`, `numOfLWComponents`, `preserveBackgroundColor`, `printing`, `printingThreads`, `SEARCH_HEAVYWEIGHTS`, `serialPersistentFields`, `serialVersionUID`

Fields inherited from class `java.awt.Component`

acc, accessibleContext, actionListenerK, adjustmentListenerK, appContext, autoFocusTransferOnDisposal, background, backgroundEraseDisabled, BOTTOM_ALIGNMENT, boundsOp, bufferStrategy, CENTER_ALIGNMENT, changeSupport, coalesceEventsParams, coalesceMap, coalescingEnabled, componentListener, componentListenerK, componentOrientation, componentSerializedDataVersion, compoundShape, containerListenerK, cursor, dropTarget, enabled, eventCache, eventLog, eventMask, FOCUS_TRAVERSABLE_DEFAULT, FOCUS_TRAVERSABLE_SET, FOCUS_TRAVERSABLE_UNKNOWN, focusable, focusListener, focusListenerK, focusLog, focusTraversalKeyPropertyNames, focusTraversalKeys, focusTraversalKeysEnabled, font, foreground, graphicsConfig, height, hierarchyBoundsListener, hierarchyBoundsListenerK, hierarchyListener, hierarchyListenerK, ignoreRepaint, incRate, inputMethodListener, inputMethodListenerK, isAddNotifyComplete, isFocusTraversableOverridden, isInc, isPacked, itemListenerK, keyListener, keyListenerK, LEFT_ALIGNMENT, locale, LOCK, log, maxSize, maxSizeSet, minSize, minSizeSet, mixingCutoutRegion, mixingLog, mouseListener, mouseListenerK, mouseMotionListener, mouseMotionListenerK, mouseWheelListener, mouseWheelListenerK, name, nameExplicitlySet, newEventsOnly, objectLock, ownedWindowK, parent, peer, peerFont, popups, prefSize, prefSizeSet, requestFocusController, RIGHT_ALIGNMENT, serialVersionUID, textListenerK, TOP_ALIGNMENT, valid, visible, width, windowClosingException, windowFocusListenerK, windowListenerK, windowStateListenerK, x, y

Fields inherited from interface `java.awt.image.ImageObserver`

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

public	`GasPump2Panel(GasPump2 gasPump2, javax.swing.JTextPane textPaneOutput, InputPanel inputPanel)` Instantiates a new GasPump2Panel, add the buttons and controllers.

Method Summary

void	`addControllers()` Adds the controllers.
javax.swing.JButton	`getButtonActivate()` Gets the button activate.
javax.swing.JButton	`getButtonCancel()` Gets the button cancel.
javax.swing.JButton	`getButtonNoReceipt()` Gets the button no receipt.
javax.swing.JButton	`getButtonPayCash()` Gets the button pay cash.
javax.swing.JButton	`getButtonPremium()` Gets the button premium.
javax.swing.JButton	`getButtonPumpLiter()` Gets the button pump liter.
javax.swing.JButton	`getButtonReceipt()` Gets the button receipt.
javax.swing.JButton	`getButtonRegular()` Gets the button regular.

javax.swing.JButton	getButtonStart() Gets the button start.
javax.swing.JButton	getButtonStartPump() Gets the button start pump.
javax.swing.JButton	getButtonStopPump() Gets the button stop pump.
javax.swing.JButton	getButtonSuper() Gets the button super.
GasPump2	getGasPump2() Gets the gas pump 2.
javax.swing.JLabel	getLabelGasPump2() Gets the label gas pump 2.

Methods inherited from class `javax.swing.JPanel`
`getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI, writeObject`

Methods inherited from class `javax.swing.JComponent`

_paintImmediately, addAncestorListener, addNotify, addVetoableChangeListener, adjustPaintFlags, alwaysOnTop, checkIfChildObscuredBySibling, clientPropertyChanged, componentInputMapChanged, computeVisibleRect, computeVisibleRect, compWriteObjectNotify, contains, createToolTip, deregisterNextFocusableComponent, disable, dndDone, dropLocationForPoint, enable, fetchRectangle, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAncestorNotifier, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperties, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getCreatedDoubleBuffer, getDebugGraphicsOptions, getDefaultLocale, getFlag, getFontMetrics, getGraphics, getGraphicsInvoked, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getManagingFocusBackwardTraversalKeys, getManagingFocusForwardTraversalKeys, getMaximumSize, getMinimumSize, getNextFocusableComponent, getObscuredState, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getUIClassID, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getWriteObjCounter, getX, getY, grabFocus, hide, isComponentObtainingGraphicsFrom, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPainting, isPaintingForPrint, isPaintingOrigin, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintForceDoubleBuffered, paintImmediately, paintImmediately, paintToOffscreen, paramString, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyBindings, processKeyBindingsForAllComponents, processKeyEvent, processMouseEvent, processMouseMotionEvent, putClientProperty, readObject, rectangleIsObscured, recycleRectangle, registerKeyboardAction, registerKeyboardAction, registerNextFocusableComponent, registerNextFocusableComponent, registerWithKeyboardManager, registerWithKeyboardManager, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, safelyGetGraphics, safelyGetGraphics, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setCreatedDoubleBuffer, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setDropLocation, setEnabled, setFlag, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPaintingChild, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setUIProperty, setVerifyInputWhenFocusTarget, setVisible, setWriteObjCounter, shouldDebugGraphics, superProcessMouseMotionEvent, uninstallUIAndProperties, unregisterKeyboardAction, unregisterWithKeyboardManager, unregisterWithKeyboardManager, update, updateUI, writeObject

Methods inherited from class `java.awt.Container`

add, add, add, add, add, addContainerListener, addDelicately, addImpl, addNotify, addPropertyChangeListener, addPropertyChangeListener, adjustDecendantsOnParent, adjustDescendants, adjustListeningChildren, applyComponentOrientation, areFocusTraversalKeysSet, canContainFocusOwner, checkAdding, checkAddToSelf, checkGD, checkNotAWindow, clearCurrentFocusCycleRootOnHide, clearMostRecentFocusOwnerOnHide, containsFocus, countComponents, countHierarchyMembers, createChildHierarchyEvents, createHierarchyEvents, decreaseComponentCount, deliverEvent, dispatchEventImpl, dispatchEventToSelf, doLayout, eventEnabled, findComponentAt, findComponentAt, findComponentAt, findComponentAtImpl, findTraversalRoot, getAccessibleAt, getAccessibleChild, getAccessibleChildrenCount, getAlignmentX, getAlignmentY, getBottommostComponentIndex, getChildAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents_NoClientCode, getComponents, getComponentsSync, getComponentZOrder, getContainerListeners, getDropTargetEventTarget, getFocusTraversalKeys, getFocusTraversalPolicy, getHeavyweightContainer, getInsets, getLayout, getListeners, getListenersCount, getMaximumSize, getMinimumSize, getMouseEventTarget, getMouseEventTarget, getMouseEventTargetImpl, getMousePosition, getOpaqueShape, getPreferredSize, getTopmostComponentIndex, getTraversalRoot, hasHeavyweightDescendants, hasLightweightDescendants, increaseComponentCount, initializeFocusTraversalKeys, initIDs, insets, invalidate, invalidateParent, invalidateTree, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, isParentOf, isRecursivelyVisibleUpToHeavyweightContainer, isRemoveNotifyNeeded, isSameOrAncestorOf, isValidateRoot, layout, lightweightPaint, lightweightPrint, list, list, locate, minimumSize, mixOnHiding, mixOnReshaping, mixOnShowing, mixOnValidating, mixOnZOrderChanging, numListening, paint, paintComponents, paintHeavyweightComponents, paramString, postProcessKeyEvent, postsOldMouseEvents, preferredSize, preProcessKeyEvent, print, printComponents, printHeavyweightComponents, processContainerEvent, processEvent, proxyEnableEvents, readObject, recursiveApplyCurrentShape, recursiveApplyCurrentShape, recursiveApplyCurrentShape, recursiveHideHeavyweightChildren, recursiveRelocateHeavyweightChildren, recursiveShowHeavyweightChildren, recursiveSubtractAndApplyShape, recursiveSubtractAndApplyShape, recursiveSubtractAndApplyShape, remove, remove, removeAll, removeContainerListener, removeDelicately, removeNotify, reparentChild, reparentTraverse, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, setLayout, startLWModal, stopLWModal, transferFocusDownCycle, update, updateGraphicsData, validate, validateTree, validateUnconditionally, writeObject

Methods inherited from class `java.awt.Component`

```
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener,
addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener,
addMouseMotionListener, addMouseWheelListener, addNotify, addPropertyChangeListener,
addPropertyChangeListener, adjustListeningChildrenOnParent,
applyComponentOrientation, applyCompoundShape, applyCurrentShape,
applyCurrentShapeBelowMe, areBoundsValid, areFocusTraversalKeysSet,
areInputMethodsEnabled, autoProcessMouseWheel, bounds, calculateCurrentShape,
canBeFocusOwner, canBeFocusOwnerRecursively, checkCoalescing, checkGD, checkImage,
checkImage, checkTreeLock, checkWindowClosingException,
clearCurrentFocusCycleRootOnHide, clearMostRecentFocusOwnerOnHide, coalesceEvents,
constructComponentName, contains, contains, containsFocus, countHierarchyMembers,
createBufferStrategy, createBufferStrategy, createHierarchyEvents, createImage,
createImage, createVolatileImage, createVolatileImage, deliverEvent, disable,
disableEvents, dispatchEvent, dispatchEventImpl, dispatchMouseWheelToAncestor,
doLayout, doSwingSerialization, enable, enable, enableEvents, enableInputMethods,
eventEnabled, eventTypeEnabled, findUnderMouseInWindow, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
getAccessControlContext, getAccessibleContext, getAccessibleIndexInParent,
getAccessibleStateSet, getAlignmentX, getAlignmentY, getAppliedShape, getBackBuffer,
getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds,
getBoundsOp, getBufferStrategy, getColorModel, getComponentAt, getComponentAt,
getComponentListeners, getComponentOrientation, getContainer, getContainingWindow,
getCursor_NoClientCode, getCursor, getDropTarget, getFocusCycleRootAncestor,
getFocusListeners, getFocusTraversalKeys_NoIDCheck, getFocusTraversalKeys,
getFocusTraversalKeysEnabled, getFont_NoClientCode, getFont, getFontMetrics,
getForeground, getGraphics_NoClientCode, getGraphics,
getGraphicsConfiguration_NoClientCode, getGraphicsConfiguration, getHeight,
getHierarchyBoundsListeners, getHierarchyListeners, getHWPeerAboveMe,
getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests,
getInsets_NoClientCode, getKeyListeners, getListeners, getLocale, getLocation,
getLocation, getLocationOnScreen_NoTreeLock, getLocationOnScreen,
getLocationOnWindow, getMaximumSize, getMinimumSize, getMouseListeners,
getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName,
getNativeContainer, getNextFocusCandidate, getNormalShape, getObjectLock,
getOpaqueShape, getParent_NoClientCode, getParent, getPeer, getPreferredSize,
getPropertyChangeListeners, getPropertyChangeListeners, getRecursivelyVisibleBounds,
getSiblingIndexAbove, getSiblingIndexBelow, getSize, getSize, getToolkit,
getToolkitImpl, getTraversalRoot, getTreeLock, getWidth, getX, getY, gotFocus,
handleEvent, hasFocus, hide, imageUpdate, initializeFocusTraversalKeys, initIDs,
inside, invalidate, invalidateIfValid, invalidateParent,
isAutoFocusTransferOnDisposal, isBackgroundSet, isCoalesceEventsOverriden,
isCoalescingEnabled, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled,
isEnabledImpl, isFocusable, isFocusCycleRoot, isFocusOwner, isFocusTraversable,
isFocusTraversableOverridden, isFontSet, isForegroundSet, isInstanceOf,
isLightweight, isMaximumSizeSet, isMinimumSizeSet, isMixingNeeded,
isNonOpaqueForMixing, isOpaque, isPreferredSizeSet, isRecursivelyVisible,
isRequestFocusAccepted, isSameOrAncestorOf, isShowing, isValid,
isVisible_NoClientCode, isVisible, keyDown, keyUp, layout, lightweightPaint,
lightweightPrint, list, list, list, list, list, locate, location_NoClientCode,
location, lostFocus, minimumSize, mixOnHiding, mixOnReshaping, mixOnShowing,
mixOnValidating, mixOnZOrderChanging, mouseDown, mouseDrag, mouseEnter, mouseExit,
mouseMove, mouseUp, move, nextFocus, notifyNewBounds, numListening, paint, paintAll,
paintHeavyweightComponents, paramString, pointRelativeToComponent, postEvent,
postsOldMouseEvents, preferredSize, prepareImage, prepareImage, print, printAll,
printHeavyweightComponents, processComponentEvent, processEvent, processFocusEvent,
processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent,
```

processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, readObject, relocateComponent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, repaintParentIfNeeded, requestFocus, requestFocus, requestFocus, requestFocus, requestFocusHelper, requestFocusHelper, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, reshape, reshapeNativePeer, resize, resize, revalidate, revalidateSynchronously, setAutoFocusTransferOnDisposal, setBackground, setBounds, setBounds, setBoundsOp, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeys_NoIDCheck, setFocusTraversalKeys, setFocusTraversalKeysEnabled, setFont, setForeground, setGraphicsConfiguration, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setRequestFocusController, setSize, setSize, setVisible, show, show, size, subtractAndApplyShape, subtractAndApplyShapeBelowMe, toString, transferFocus, transferFocus, transferFocusBackward, transferFocusBackward, transferFocusUpCycle, update, updateCursorImmediately, updateGraphicsData, updateZOrder, validate, writeObject

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface `java.awt.image.ImageObserver`

imageUpdate

Methods inherited from interface `java.awt.MenuContainer`

getFont, postEvent, remove

Methods inherited from interface `javax.swing.TransferHandler.HasGetTransferHandler`

getTransferHandler

Methods inherited from interface `javax.accessibility.Accessible`

getAccessibleContext

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

The Constant serialVersionUID.
Constant value: **1**

gasPump2

`private final iit.softwaresystarch.project.model.efsms.GasPump2` **`gasPump2`**

The gas pump 2.

labelGasPump2

`private final javax.swing.JLabel` **`labelGasPump2`**

The label gas pump 2.

buttonActivate

`private final javax.swing.JButton` **`buttonActivate`**

The button activate.

buttonStart

`private final javax.swing.JButton` **`buttonStart`**

The button start.

buttonPayCash

`private final javax.swing.JButton` **`buttonPayCash`**

The button pay cash.

buttonCancel

`private final javax.swing.JButton` **`buttonCancel`**

The button cancel.

buttonPremium

`private final javax.swing.JButton` **`buttonPremium`**

The button premium.

buttonRegular

`private final javax.swing.JButton` **`buttonRegular`**

The button regular.

buttonSuper

`private final javax.swing.JButton` **`buttonSuper`**

The button super.

buttonStartPump

`private final javax.swing.JButton` **`buttonStartPump`**

The button start pump.

buttonPumpLiter

`private final javax.swing.JButton` **`buttonPumpLiter`**

(continued from last page)

The button pump liter.

buttonStopPump

`private final javax.swing.JButton` **`buttonStopPump`**

The button stop pump.

buttonNoReceipt

`private final javax.swing.JButton` **`buttonNoReceipt`**

The button no receipt.

buttonReceipt

`private final javax.swing.JButton` **`buttonReceipt`**

The button receipt.

textPaneOutput

`private final javax.swing.JTextPane` **`textPaneOutput`**

The text pane output.

inputPanel

`private final iit.softwaresystarch.project.view.InputPanel` **`inputPanel`**

The input panel.

Constructors

GasPump2Panel

```
public GasPump2Panel(GasPump2 gasPump2,
                     javax.swing.JTextPane textPaneOutput,
                     InputPanel inputPanel)
```

Instantiates a new GasPump2Panel, add the buttons and controllers.

Parameters:

`gasPump2` - the gas pump 2
`textPaneOutput` - the text pane output
`inputPanel` - the input panel

Methods

addControllers

`private void` **`addControllers`**`()`

Adds the controllers.

(continued from last page)

getGasPump2

`public GasPump2 getGasPump2()`

Gets the gas pump 2.

Returns:
the gas pump 2

getLabelGasPump2

`public javax.swing.JLabel getLabelGasPump2()`

Gets the label gas pump 2.

Returns:
the label gas pump 2

getButtonActivate

`public javax.swing.JButton getButtonActivate()`

Gets the button activate.

Returns:
the button activate

getButtonStart

`public javax.swing.JButton getButtonStart()`

Gets the button start.

Returns:
the button start

getButtonPayCash

`public javax.swing.JButton getButtonPayCash()`

Gets the button pay cash.

Returns:
the button pay cash

getButtonCancel

`public javax.swing.JButton getButtonCancel()`

Gets the button cancel.

Returns:
the button cancel

getButtonPremium

`public javax.swing.JButton getButtonPremium()`

(continued from last page)

Gets the button premium.

Returns:
the button premium

getButtonRegular

`public javax.swing.JButton getButtonRegular()`

Gets the button regular.

Returns:
the button regular

getButtonSuper

`public javax.swing.JButton getButtonSuper()`

Gets the button super.

Returns:
the button super

getButtonStartPump

`public javax.swing.JButton getButtonStartPump()`

Gets the button start pump.

Returns:
the button start pump

getButtonPumpLiter

`public javax.swing.JButton getButtonPumpLiter()`

Gets the button pump liter.

Returns:
the button pump liter

getButtonStopPump

`public javax.swing.JButton getButtonStopPump()`

Gets the button stop pump.

Returns:
the button stop pump

getButtonNoReceipt

`public javax.swing.JButton getButtonNoReceipt()`

Gets the button no receipt.

Returns:
the button no receipt

getButtonReceipt

```
public javax.swing.JButton getButtonReceipt()
```

Gets the button receipt.

Returns:
the button receipt

iit.softwaresystarch.project.view
Class InputPanel

```
java.lang.Object
  |-java.awt.Component
     |-java.awt.Container
        |-javax.swing.JComponent
           |-javax.swing.JPanel
              |-iit.softwaresystarch.project.view.InputPanel
```

All Implemented Interfaces:
java.io.Serializable, java.awt.MenuContainer, java.awt.image.ImageObserver, javax.swing.TransferHandler.HasGetTransferHandler, java.io.Serializable, javax.accessibility.Accessible

---

public class **InputPanel**
extends javax.swing.JPanel

The Class InputPanel.

Field Summary

private final	aInput The a input.
private final	bInput The b input.
private final	buttonPrintPump1 The button print pump 1.
private final	buttonPrintPump2 The button print pump 2.
private final	buttonReset The button reset.
private final	cashInput The cash input.
private final	cInput The c input.
private final	labelA The label A.
private final	labelB The label B.
private final	labelC The label C.
private final	labelCash The label cash.

private final	labelPrint The label print.
private static final	serialVersionUID The Constant serialVersionUID. Value: **1**

Fields inherited from class `javax.swing.JPanel`

`uiClassID`

Fields inherited from class `javax.swing.JComponent`

`aaTextInfo`, `actionMap`, `ACTIONMAP_CREATED`, `alignmentX`, `alignmentY`, `ANCESTOR_INPUTMAP_CREATED`, `ANCESTOR_USING_BUFFER`, `ancestorInputMap`, `autoscrolls`, `AUTOSCROLLS_SET`, `border`, `clientProperties`, `COMPLETELY_OBSCURED`, `componentObtainingGraphicsFrom`, `componentObtainingGraphicsFromLock`, `CREATED_DOUBLE_BUFFER`, `DEBUG_GRAPHICS_LOADED`, `defaultLocale`, `flags`, `FOCUS_INPUTMAP_CREATED`, `FOCUS_TRAVERSAL_KEYS_BACKWARD_SET`, `FOCUS_TRAVERSAL_KEYS_FORWARD_SET`, `focusController`, `focusInputMap`, `INHERITS_POPUP_MENU`, `INPUT_VERIFIER_SOURCE_KEY`, `inputVerifier`, `IS_DOUBLE_BUFFERED`, `IS_OPAQUE`, `IS_PAINTING_TILE`, `IS_PRINTING`, `IS_PRINTING_ALL`, `IS_REPAINTING`, `isAlignmentXSet`, `isAlignmentYSet`, `KEY_EVENTS_ENABLED`, `KEYBOARD_BINDINGS_KEY`, `listenerList`, `managingFocusBackwardTraversalKeys`, `managingFocusForwardTraversalKeys`, `NEXT_FOCUS`, `NOT_OBSCURED`, `OPAQUE_SET`, `paintingChild`, `PARTIALLY_OBSCURED`, `popupMenu`, `readObjectCallbacks`, `REQUEST_FOCUS_DISABLED`, `RESERVED_1`, `RESERVED_2`, `RESERVED_3`, `RESERVED_4`, `RESERVED_5`, `RESERVED_6`, `revalidateRunnableScheduled`, `tempRectangles`, `TOOL_TIP_TEXT_KEY`, `ui`, `uiClassID`, `UNDEFINED_CONDITION`, `verifyInputWhenFocusTarget`, `vetoableChangeSupport`, `WHEN_ANCESTOR_OF_FOCUSED_COMPONENT`, `WHEN_FOCUSED`, `WHEN_IN_FOCUSED_WINDOW`, `WHEN_IN_FOCUSED_WINDOW_BINDINGS`, `WIF_INPUTMAP_CREATED`, `windowInputMap`, `WRITE_OBJ_COUNTER_FIRST`, `WRITE_OBJ_COUNTER_LAST`

Fields inherited from class `java.awt.Container`

`component`, `containerListener`, `containerSerializedDataVersion`, `descendantsCount`, `descendUnconditionallyWhenValidating`, `dispatcher`, `EMPTY_ARRAY`, `eventLog`, `focusCycleRoot`, `focusTraversalPolicy`, `focusTraversalPolicyProvider`, `INCLUDE_SELF`, `isJavaAwtSmartInvalidate`, `layoutMgr`, `listeningBoundsChildren`, `listeningChildren`, `log`, `mixingLog`, `modalAppContext`, `modalComp`, `numOfHWComponents`, `numOfLWComponents`, `preserveBackgroundColor`, `printing`, `printingThreads`, `SEARCH_HEAVYWEIGHTS`, `serialPersistentFields`, `serialVersionUID`

Fields inherited from class `java.awt.Component`

acc, accessibleContext, actionListenerK, adjustmentListenerK, appContext, autoFocusTransferOnDisposal, background, backgroundEraseDisabled, BOTTOM_ALIGNMENT, boundsOp, bufferStrategy, CENTER_ALIGNMENT, changeSupport, coalesceEventsParams, coalesceMap, coalescingEnabled, componentListener, componentListenerK, componentOrientation, componentSerializedDataVersion, compoundShape, containerListenerK, cursor, dropTarget, enabled, eventCache, eventLog, eventMask, FOCUS_TRAVERSABLE_DEFAULT, FOCUS_TRAVERSABLE_SET, FOCUS_TRAVERSABLE_UNKNOWN, focusable, focusListener, focusListenerK, focusLog, focusTraversalKeyPropertyNames, focusTraversalKeys, focusTraversalKeysEnabled, font, foreground, graphicsConfig, height, hierarchyBoundsListener, hierarchyBoundsListenerK, hierarchyListener, hierarchyListenerK, ignoreRepaint, incRate, inputMethodListener, inputMethodListenerK, isAddNotifyComplete, isFocusTraversableOverridden, isInc, isPacked, itemListenerK, keyListener, keyListenerK, LEFT_ALIGNMENT, locale, LOCK, log, maxSize, maxSizeSet, minSize, minSizeSet, mixingCutoutRegion, mixingLog, mouseListener, mouseListenerK, mouseMotionListener, mouseMotionListenerK, mouseWheelListener, mouseWheelListenerK, name, nameExplicitlySet, newEventsOnly, objectLock, ownedWindowK, parent, peer, peerFont, popups, prefSize, prefSizeSet, requestFocusController, RIGHT_ALIGNMENT, serialVersionUID, textListenerK, TOP_ALIGNMENT, valid, visible, width, windowClosingException, windowFocusListenerK, windowListenerK, windowStateListenerK, x, y

Fields inherited from interface `java.awt.image.ImageObserver`

ABORT, ALLBITS, ERROR, FRAMEBITS, HEIGHT, PROPERTIES, SOMEBITS, WIDTH

Constructor Summary

public	`InputPanel()` Instantiates a new InputPanel with the input fields, labels and controllers.

Method Summary

javax.swing.JFormattedTextField	`getaInput()` Gets the a input.
javax.swing.JFormattedTextField	`getbInput()` Gets the b input.
javax.swing.JButton	`getButtonPrintPump1()` Gets the button print pump 1.
javax.swing.JButton	`getButtonPrintPump2()` Gets the button print pump 2.
javax.swing.JButton	`getButtonReset()` Gets the button reset.
javax.swing.JFormattedTextField	`getCashInput()` Gets the cash input.
javax.swing.JFormattedTextField	`getcInput()` Gets the c input.
javax.swing.JLabel	`getLabelA()` Gets the label A.
javax.swing.JLabel	`getLabelB()` Gets the label B.

javax.swing.JLabel	getLabelC() Gets the label C.
javax.swing.JLabel	getLabelCash() Gets the label cash.
javax.swing.JLabel	getLabelPrint() Gets the label print.

Methods inherited from class javax.swing.JPanel
getAccessibleContext, getUI, getUIClassID, paramString, setUI, updateUI, writeObject

Methods inherited from class javax.swing.JComponent

_paintImmediately, addAncestorListener, addNotify, addVetoableChangeListener, adjustPaintFlags, alwaysOnTop, checkIfChildObscuredBySibling, clientPropertyChanged, componentInputMapChanged, computeVisibleRect, computeVisibleRect, compWriteObjectNotify, contains, createToolTip, deregisterNextFocusableComponent, disable, dndDone, dropLocationForPoint, enable, fetchRectangle, firePropertyChange, firePropertyChange, firePropertyChange, fireVetoableChange, getActionForKeyStroke, getActionMap, getActionMap, getAlignmentX, getAlignmentY, getAncestorListeners, getAncestorNotifier, getAutoscrolls, getBaseline, getBaselineResizeBehavior, getBorder, getBounds, getClientProperties, getClientProperty, getComponentGraphics, getComponentPopupMenu, getConditionForKeyStroke, getCreatedDoubleBuffer, getDebugGraphicsOptions, getDefaultLocale, getFlag, getFontMetrics, getGraphics, getGraphicsInvoked, getHeight, getInheritsPopupMenu, getInputMap, getInputMap, getInputMap, getInputVerifier, getInsets, getInsets, getListeners, getLocation, getManagingFocusBackwardTraversalKeys, getManagingFocusForwardTraversalKeys, getMaximumSize, getMinimumSize, getNextFocusableComponent, getObscuredState, getPopupLocation, getPreferredSize, getRegisteredKeyStrokes, getRootPane, getSize, getToolTipLocation, getToolTipText, getToolTipText, getTopLevelAncestor, getTransferHandler, getUIClassID, getVerifyInputWhenFocusTarget, getVetoableChangeListeners, getVisibleRect, getWidth, getWriteObjCounter, getX, getY, grabFocus, hide, isComponentObtainingGraphicsFrom, isDoubleBuffered, isLightweightComponent, isManagingFocus, isOpaque, isOptimizedDrawingEnabled, isPainting, isPaintingForPrint, isPaintingOrigin, isPaintingTile, isRequestFocusEnabled, isValidateRoot, paint, paintBorder, paintChildren, paintComponent, paintForceDoubleBuffered, paintImmediately, paintImmediately, paintToOffscreen, paramString, print, printAll, printBorder, printChildren, printComponent, processComponentKeyEvent, processKeyBinding, processKeyBindings, processKeyBindingsForAllComponents, processKeyEvent, processMouseEvent, processMouseMotionEvent, putClientProperty, readObject, rectangleIsObscured, recycleRectangle, registerKeyboardAction, registerKeyboardAction, registerNextFocusableComponent, registerNextFocusableComponent, registerWithKeyboardManager, registerWithKeyboardManager, removeAncestorListener, removeNotify, removeVetoableChangeListener, repaint, repaint, requestDefaultFocus, requestFocus, requestFocus, requestFocusInWindow, requestFocusInWindow, resetKeyboardActions, reshape, revalidate, safelyGetGraphics, safelyGetGraphics, scrollRectToVisible, setActionMap, setAlignmentX, setAlignmentY, setAutoscrolls, setBackground, setBorder, setComponentPopupMenu, setCreatedDoubleBuffer, setDebugGraphicsOptions, setDefaultLocale, setDoubleBuffered, setDropLocation, setEnabled, setFlag, setFocusTraversalKeys, setFont, setForeground, setInheritsPopupMenu, setInputMap, setInputVerifier, setMaximumSize, setMinimumSize, setNextFocusableComponent, setOpaque, setPaintingChild, setPreferredSize, setRequestFocusEnabled, setToolTipText, setTransferHandler, setUI, setUIProperty, setVerifyInputWhenFocusTarget, setVisible, setWriteObjCounter, shouldDebugGraphics, superProcessMouseMotionEvent, uninstallUIAndProperties, unregisterKeyboardAction, unregisterWithKeyboardManager, unregisterWithKeyboardManager, update, updateUI, writeObject

Methods inherited from class `java.awt.Container`

add, add, add, add, add, addContainerListener, addDelicately, addImpl, addNotify, addPropertyChangeListener, addPropertyChangeListener, adjustDecendantsOnParent, adjustDescendants, adjustListeningChildren, applyComponentOrientation, areFocusTraversalKeysSet, canContainFocusOwner, checkAdding, checkAddToSelf, checkGD, checkNotAWindow, clearCurrentFocusCycleRootOnHide, clearMostRecentFocusOwnerOnHide, containsFocus, countComponents, countHierarchyMembers, createChildHierarchyEvents, createHierarchyEvents, decreaseComponentCount, deliverEvent, dispatchEventImpl, dispatchEventToSelf, doLayout, eventEnabled, findComponentAt, findComponentAt, findComponentAt, findComponentAtImpl, findTraversalRoot, getAccessibleAt, getAccessibleChild, getAccessibleChildrenCount, getAlignmentX, getAlignmentY, getBottommostComponentIndex, getChildAt, getComponent, getComponentAt, getComponentAt, getComponentCount, getComponents_NoClientCode, getComponents, getComponentsSync, getComponentZOrder, getContainerListeners, getDropTargetEventTarget, getFocusTraversalKeys, getFocusTraversalPolicy, getHeavyweightContainer, getInsets, getLayout, getListeners, getListenersCount, getMaximumSize, getMinimumSize, getMouseEventTarget, getMouseEventTarget, getMouseEventTargetImpl, getMousePosition, getOpaqueShape, getPreferredSize, getTopmostComponentIndex, getTraversalRoot, hasHeavyweightDescendants, hasLightweightDescendants, increaseComponentCount, initializeFocusTraversalKeys, initIDs, insets, invalidate, invalidateParent, invalidateTree, isAncestorOf, isFocusCycleRoot, isFocusCycleRoot, isFocusTraversalPolicyProvider, isFocusTraversalPolicySet, isParentOf, isRecursivelyVisibleUpToHeavyweightContainer, isRemoveNotifyNeeded, isSameOrAncestorOf, isValidateRoot, layout, lightweightPaint, lightweightPrint, list, list, locate, minimumSize, mixOnHiding, mixOnReshaping, mixOnShowing, mixOnValidating, mixOnZOrderChanging, numListening, paint, paintComponents, paintHeavyweightComponents, paramString, postProcessKeyEvent, postsOldMouseEvents, preferredSize, preProcessKeyEvent, print, printComponents, printHeavyweightComponents, processContainerEvent, processEvent, proxyEnableEvents, readObject, recursiveApplyCurrentShape, recursiveApplyCurrentShape, recursiveApplyCurrentShape, recursiveHideHeavyweightChildren, recursiveRelocateHeavyweightChildren, recursiveShowHeavyweightChildren, recursiveSubtractAndApplyShape, recursiveSubtractAndApplyShape, recursiveSubtractAndApplyShape, remove, remove, removeAll, removeContainerListener, removeDelicately, removeNotify, reparentChild, reparentTraverse, setComponentZOrder, setFocusCycleRoot, setFocusTraversalKeys, setFocusTraversalPolicy, setFocusTraversalPolicyProvider, setFont, setLayout, startLWModal, stopLWModal, transferFocusDownCycle, update, updateGraphicsData, validate, validateTree, validateUnconditionally, writeObject

Methods inherited from class `java.awt.Component`

```
action, add, addComponentListener, addFocusListener, addHierarchyBoundsListener,
addHierarchyListener, addInputMethodListener, addKeyListener, addMouseListener,
addMouseMotionListener, addMouseWheelListener, addNotify, addPropertyChangeListener,
addPropertyChangeListener, adjustListeningChildrenOnParent,
applyComponentOrientation, applyCompoundShape, applyCurrentShape,
applyCurrentShapeBelowMe, areBoundsValid, areFocusTraversalKeysSet,
areInputMethodsEnabled, autoProcessMouseWheel, bounds, calculateCurrentShape,
canBeFocusOwner, canBeFocusOwnerRecursively, checkCoalescing, checkGD, checkImage,
checkImage, checkTreeLock, checkWindowClosingException,
clearCurrentFocusCycleRootOnHide, clearMostRecentFocusOwnerOnHide, coalesceEvents,
constructComponentName, contains, contains, containsFocus, countHierarchyMembers,
createBufferStrategy, createBufferStrategy, createHierarchyEvents, createImage,
createImage, createVolatileImage, createVolatileImage, deliverEvent, disable,
disableEvents, dispatchEvent, dispatchEventImpl, dispatchMouseWheelToAncestor,
doLayout, doSwingSerialization, enable, enable, enableEvents, enableInputMethods,
eventEnabled, eventTypeEnabled, findUnderMouseInWindow, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
firePropertyChange, firePropertyChange, firePropertyChange, firePropertyChange,
getAccessControlContext, getAccessibleContext, getAccessibleIndexInParent,
getAccessibleStateSet, getAlignmentX, getAlignmentY, getAppliedShape, getBackBuffer,
getBackground, getBaseline, getBaselineResizeBehavior, getBounds, getBounds,
getBoundsOp, getBufferStrategy, getColorModel, getComponentAt, getComponentAt,
getComponentListeners, getComponentOrientation, getContainer, getContainingWindow,
getCursor_NoClientCode, getCursor, getDropTarget, getFocusCycleRootAncestor,
getFocusListeners, getFocusTraversalKeys_NoIDCheck, getFocusTraversalKeys,
getFocusTraversalKeysEnabled, getFont_NoClientCode, getFont, getFontMetrics,
getForeground, getGraphics_NoClientCode, getGraphics,
getGraphicsConfiguration_NoClientCode, getGraphicsConfiguration, getHeight,
getHierarchyBoundsListeners, getHierarchyListeners, getHWPeerAboveMe,
getIgnoreRepaint, getInputContext, getInputMethodListeners, getInputMethodRequests,
getInsets_NoClientCode, getKeyListeners, getListeners, getLocale, getLocation,
getLocation, getLocationOnScreen_NoTreeLock, getLocationOnScreen,
getLocationOnWindow, getMaximumSize, getMinimumSize, getMouseListeners,
getMouseMotionListeners, getMousePosition, getMouseWheelListeners, getName,
getNativeContainer, getNextFocusCandidate, getNormalShape, getObjectLock,
getOpaqueShape, getParent_NoClientCode, getParent, getPeer, getPreferredSize,
getPropertyChangeListeners, getPropertyChangeListeners, getRecursivelyVisibleBounds,
getSiblingIndexAbove, getSiblingIndexBelow, getSize, getSize, getToolkit,
getToolkitImpl, getTraversalRoot, getTreeLock, getWidth, getX, getY, gotFocus,
handleEvent, hasFocus, hide, imageUpdate, initializeFocusTraversalKeys, initIDs,
inside, invalidate, invalidateIfValid, invalidateParent,
isAutoFocusTransferOnDisposal, isBackgroundSet, isCoalesceEventsOverriden,
isCoalescingEnabled, isCursorSet, isDisplayable, isDoubleBuffered, isEnabled,
isEnabledImpl, isFocusable, isFocusCycleRoot, isFocusOwner, isFocusTraversable,
isFocusTraversableOverridden, isFontSet, isForegroundSet, isInstanceOf,
isLightweight, isMaximumSizeSet, isMinimumSizeSet, isMixingNeeded,
isNonOpaqueForMixing, isOpaque, isPreferredSizeSet, isRecursivelyVisible,
isRequestFocusAccepted, isSameOrAncestorOf, isShowing, isValid,
isVisible_NoClientCode, isVisible, keyDown, keyUp, layout, lightweightPaint,
lightweightPrint, list, list, list, list, list, locate, location_NoClientCode,
location, lostFocus, minimumSize, mixOnHiding, mixOnReshaping, mixOnShowing,
mixOnValidating, mixOnZOrderChanging, mouseDown, mouseDrag, mouseEnter, mouseExit,
mouseMove, mouseUp, move, nextFocus, notifyNewBounds, numListening, paint, paintAll,
paintHeavyweightComponents, paramString, pointRelativeToComponent, postEvent,
postsOldMouseEvents, preferredSize, prepareImage, prepareImage, print, printAll,
printHeavyweightComponents, processComponentEvent, processEvent, processFocusEvent,
processHierarchyBoundsEvent, processHierarchyEvent, processInputMethodEvent,
```

processKeyEvent, processMouseEvent, processMouseMotionEvent, processMouseWheelEvent, readObject, relocateComponent, remove, removeComponentListener, removeFocusListener, removeHierarchyBoundsListener, removeHierarchyListener, removeInputMethodListener, removeKeyListener, removeMouseListener, removeMouseMotionListener, removeMouseWheelListener, removeNotify, removePropertyChangeListener, removePropertyChangeListener, repaint, repaint, repaint, repaint, repaintParentIfNeeded, requestFocus, requestFocus, requestFocus, requestFocus, requestFocusHelper, requestFocusHelper, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, requestFocusInWindow, reshape, reshapeNativePeer, resize, resize, revalidate, revalidateSynchronously, setAutoFocusTransferOnDisposal, setBackground, setBounds, setBounds, setBoundsOp, setComponentOrientation, setCursor, setDropTarget, setEnabled, setFocusable, setFocusTraversalKeys_NoIDCheck, setFocusTraversalKeys, setFocusTraversalKeysEnabled, setFont, setForeground, setGraphicsConfiguration, setIgnoreRepaint, setLocale, setLocation, setLocation, setMaximumSize, setMinimumSize, setName, setPreferredSize, setRequestFocusController, setSize, setSize, setVisible, show, show, size, subtractAndApplyShape, subtractAndApplyShapeBelowMe, toString, transferFocus, transferFocus, transferFocusBackward, transferFocusBackward, transferFocusUpCycle, update, updateCursorImmediately, updateGraphicsData, updateZOrder, validate, writeObject

Methods inherited from class `java.lang.Object`

clone, equals, finalize, getClass, hashCode, notify, notifyAll, registerNatives, toString, wait, wait, wait

Methods inherited from interface `java.awt.image.ImageObserver`

imageUpdate

Methods inherited from interface `java.awt.MenuContainer`

getFont, postEvent, remove

Methods inherited from interface `javax.swing.TransferHandler.HasGetTransferHandler`

getTransferHandler

Methods inherited from interface `javax.accessibility.Accessible`

getAccessibleContext

Fields

serialVersionUID

`private static final long` **`serialVersionUID`**

The Constant serialVersionUID.
Constant value: **1**

labelA

`private final javax.swing.JLabel` **`labelA`**

The label A.

labelB

`private final javax.swing.JLabel` **`labelB`**

The label B.

labelC

`private final javax.swing.JLabel` **`labelC`**

The label C.

labelCash

`private final javax.swing.JLabel` **`labelCash`**

The label cash.

aInput

`private final javax.swing.JFormattedTextField` **`aInput`**

The a input.

bInput

`private final javax.swing.JFormattedTextField` **`bInput`**

The b input.

cInput

`private final javax.swing.JFormattedTextField` **`cInput`**

The c input.

cashInput

`private final javax.swing.JFormattedTextField` **`cashInput`**

The cash input.

buttonPrintPump2

`private final javax.swing.JButton` **`buttonPrintPump2`**

The button print pump 2.

buttonPrintPump1

`private final javax.swing.JButton` **`buttonPrintPump1`**

The button print pump 1.

labelPrint

`private final javax.swing.JLabel` **`labelPrint`**

(continued from last page)

The label print.

buttonReset

`private final javax.swing.JButton buttonReset`

The button reset.

Constructors

InputPanel

`public InputPanel()`

Instantiates a new InputPanel with the input fields, labels and controllers.

Methods

getLabelA

`public javax.swing.JLabel getLabelA()`

Gets the label A.

Returns:
the label A

getLabelB

`public javax.swing.JLabel getLabelB()`

Gets the label B.

Returns:
the label B

getLabelC

`public javax.swing.JLabel getLabelC()`

Gets the label C.

Returns:
the label C

getLabelCash

`public javax.swing.JLabel getLabelCash()`

Gets the label cash.

Returns:
the label cash

getaInput

`public javax.swing.JFormattedTextField getaInput()`

(continued from last page)

Gets the a input.

Returns:
the a input

getbInput

`public javax.swing.JFormattedTextField getbInput()`

Gets the b input.

Returns:
the b input

getcInput

`public javax.swing.JFormattedTextField getcInput()`

Gets the c input.

Returns:
the c input

getCashInput

`public javax.swing.JFormattedTextField getCashInput()`

Gets the cash input.

Returns:
the cash input

getButtonPrintPump2

`public javax.swing.JButton getButtonPrintPump2()`

Gets the button print pump 2.

Returns:
the button print pump 2

getButtonPrintPump1

`public javax.swing.JButton getButtonPrintPump1()`

Gets the button print pump 1.

Returns:
the button print pump 1

getLabelPrint

`public javax.swing.JLabel getLabelPrint()`

Gets the label print.

Returns:
the label print

getButtonReset

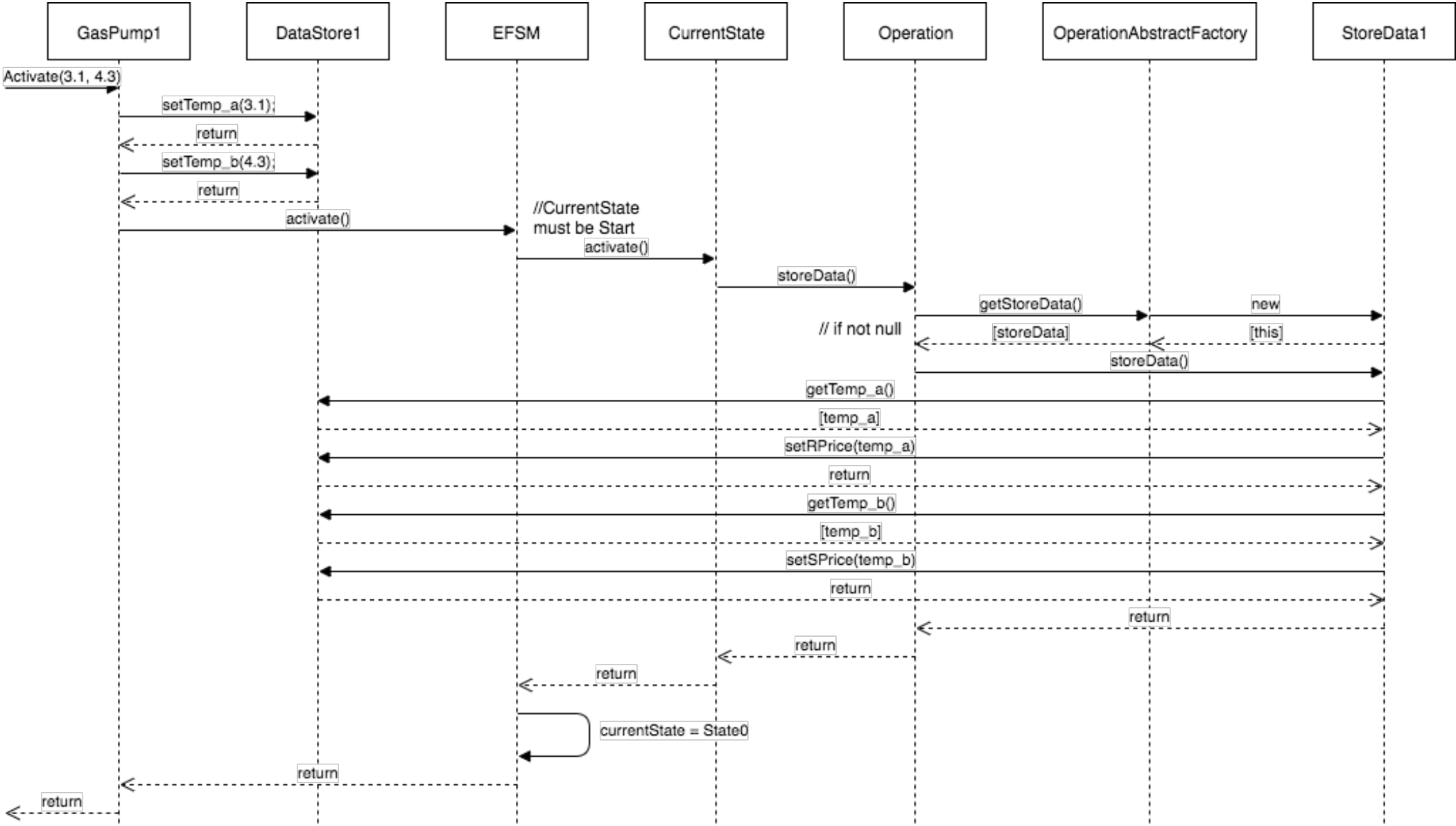
```
public javax.swing.JButton getButtonReset()
```

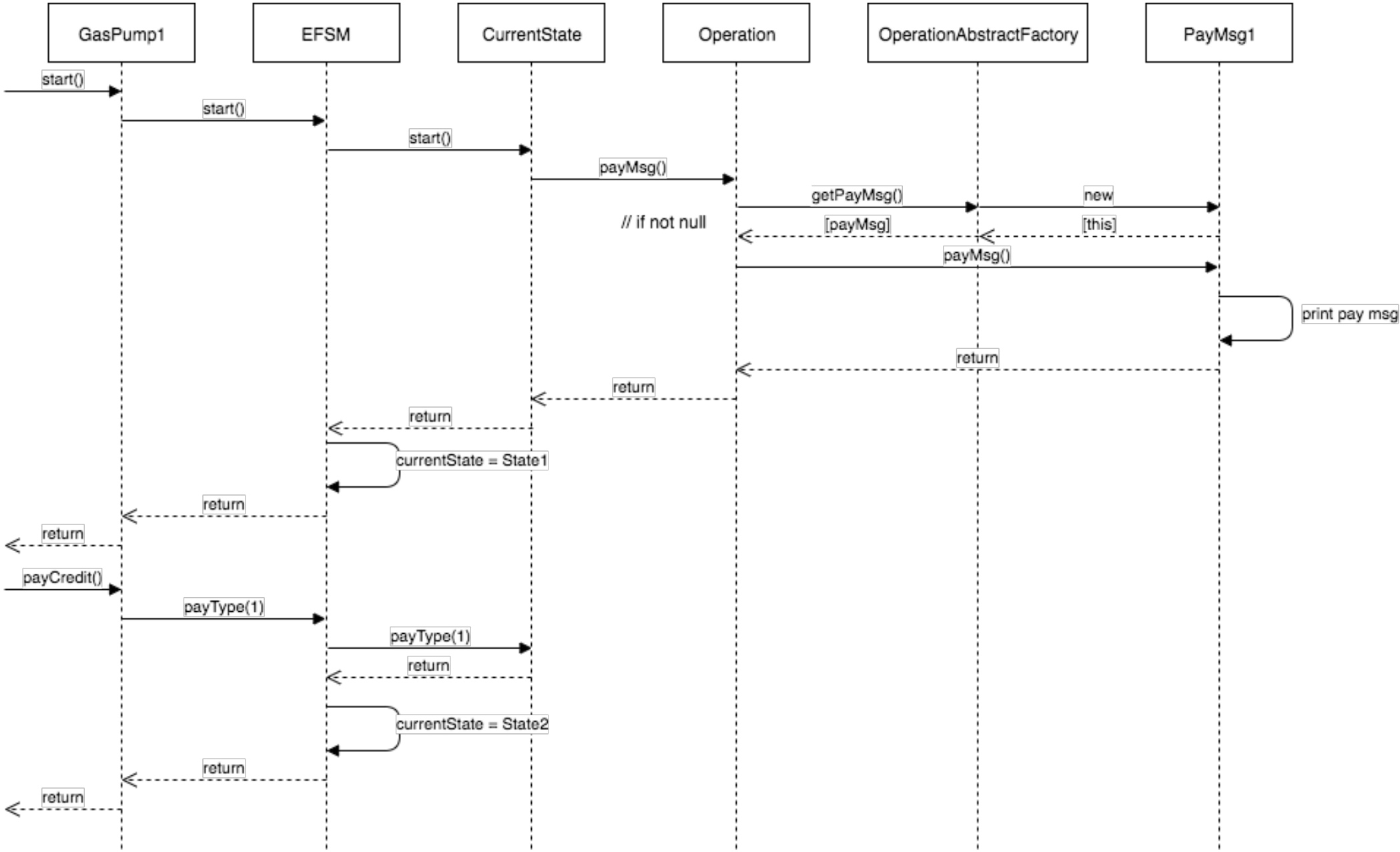
Gets the button reset.

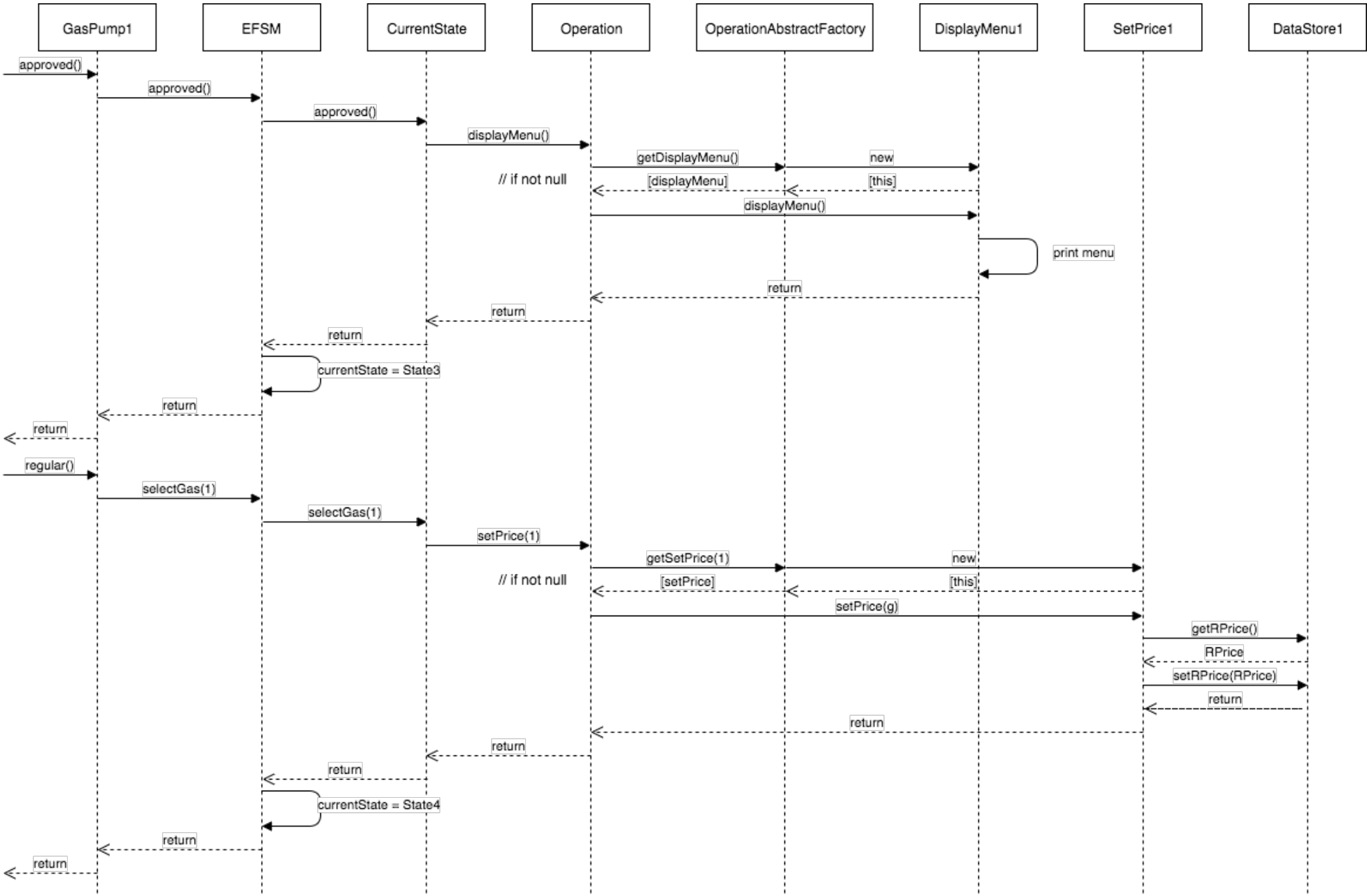
Returns:
the button reset

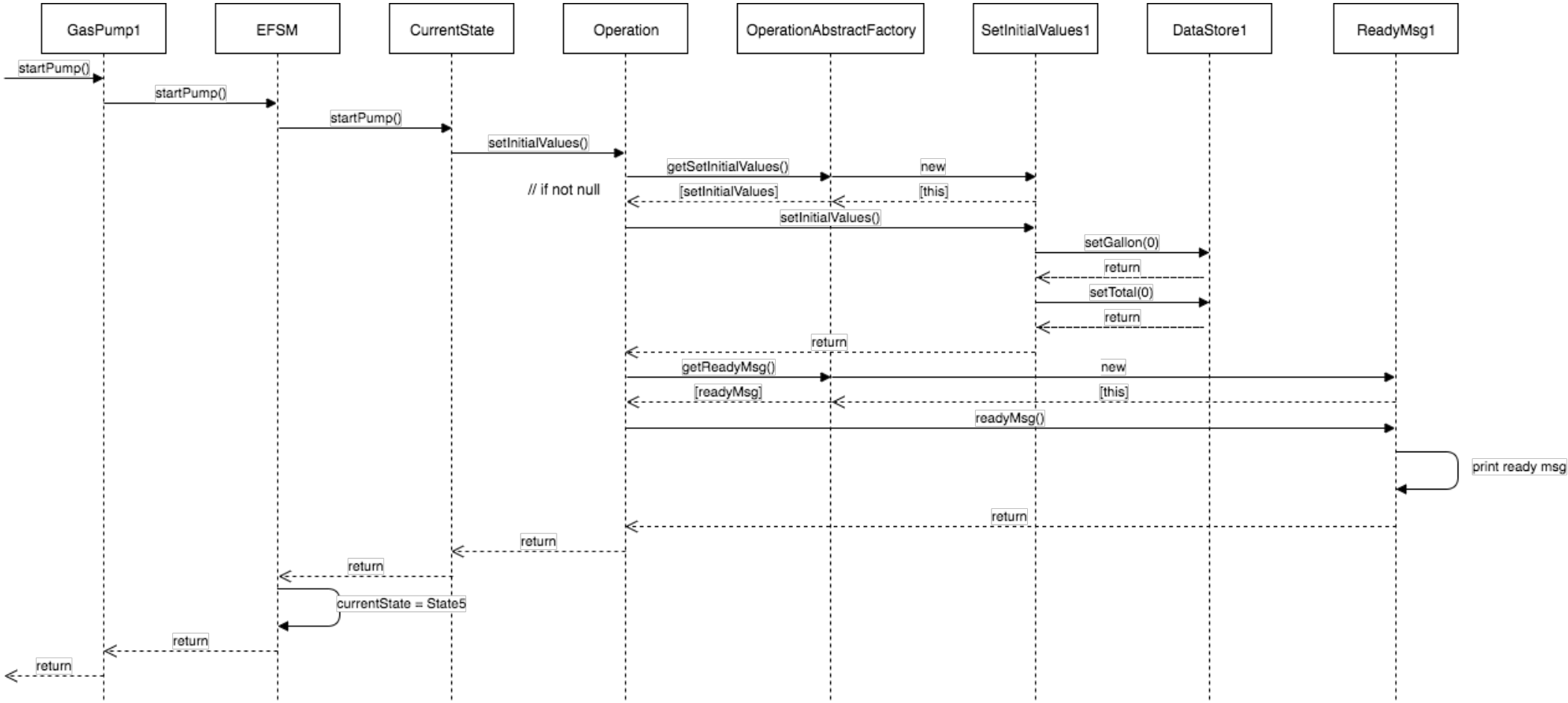
4. Dynamics. Two sequence diagrams for two Scenarios:

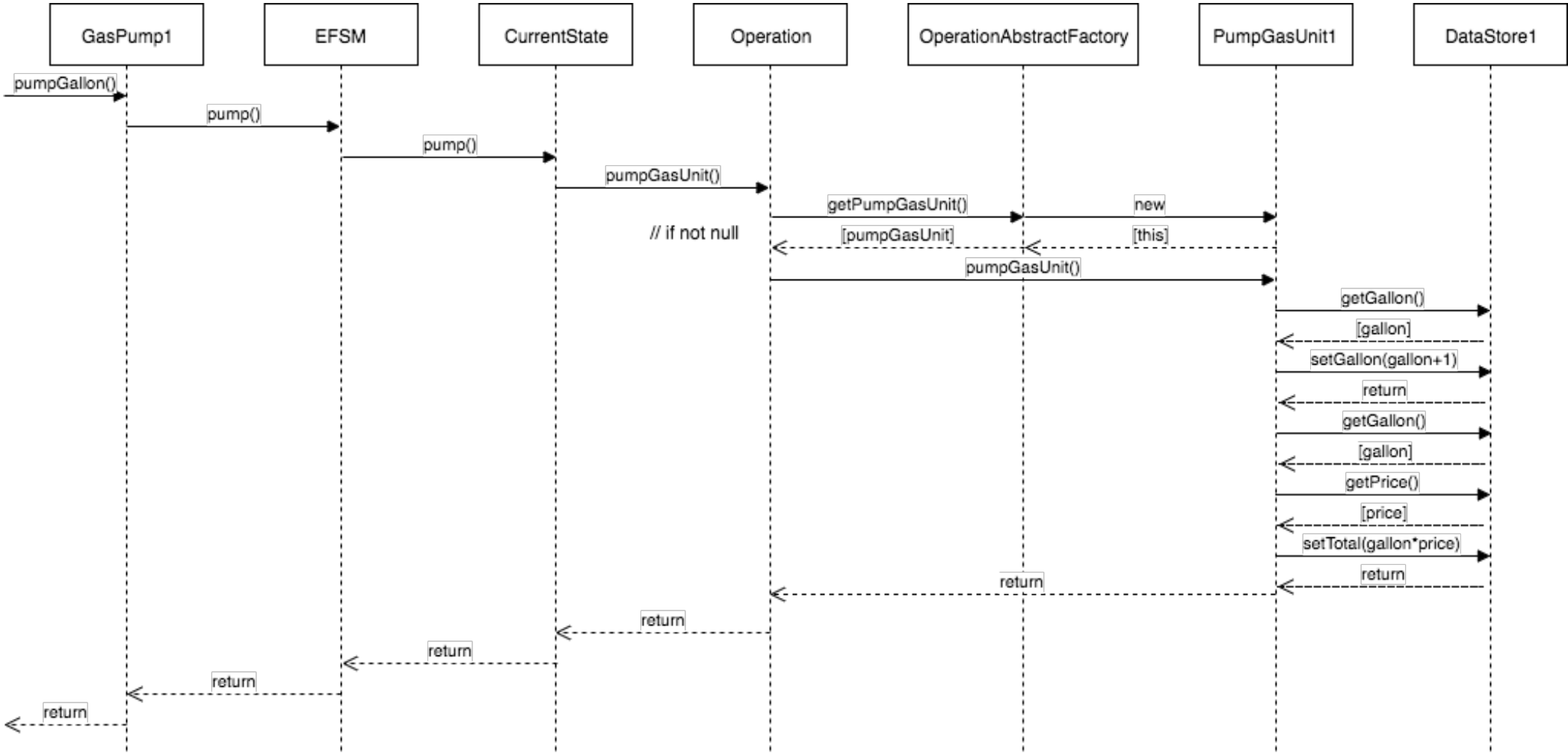
a. Scenario-I show how one gallon of Regular gas is disposed in *GasPump-1*, i.e., the following sequence of operations is issued: *Activate(3.1, 4.3), Start(), PayCredit(), Approved(), Regular(), StartPump(), PumpGallon(), StopPump()*

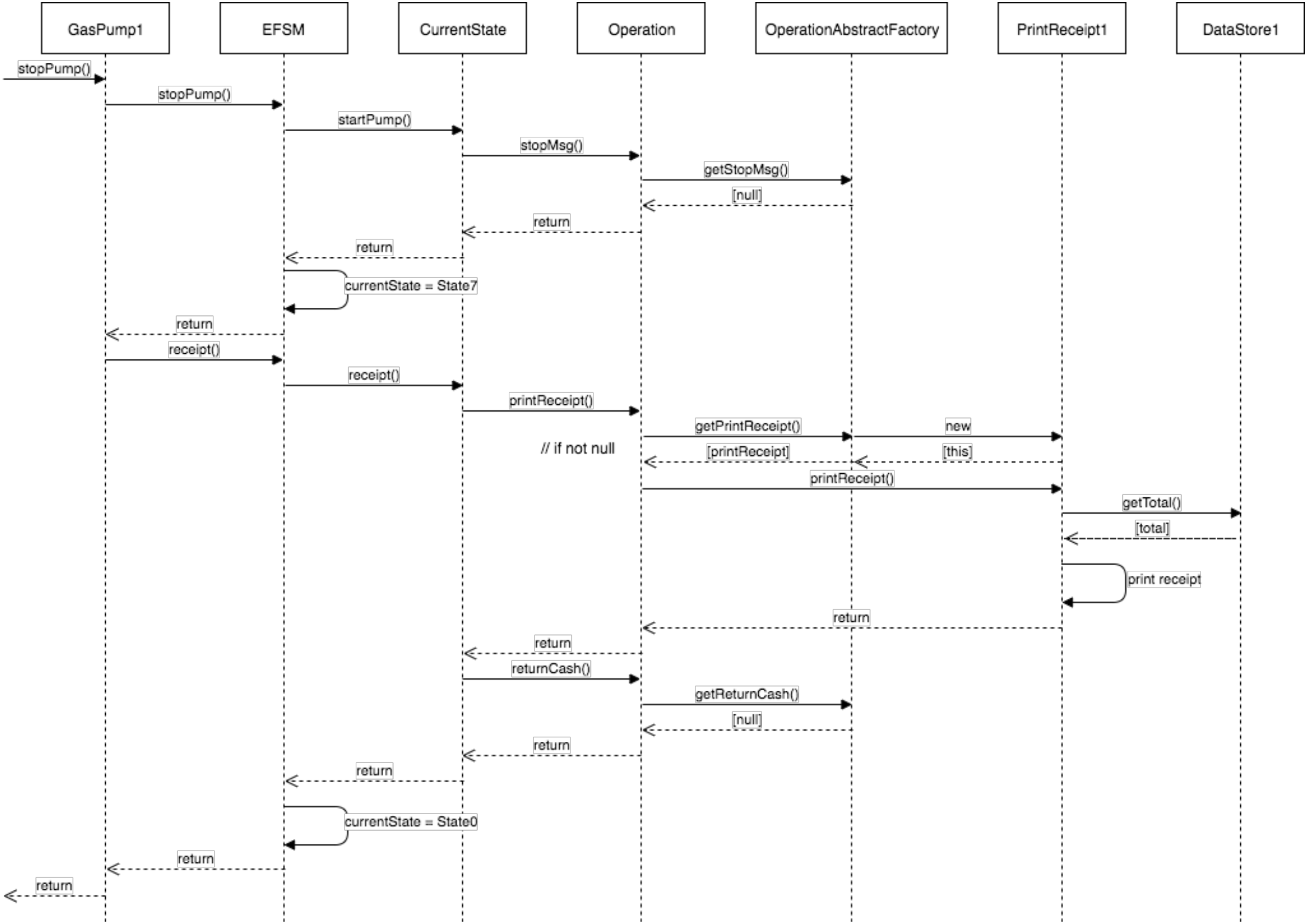

GasPump1
DataStore1
EFSM
CurrentState
Operation
OperationAbstractFactory
StoreData1
Activate(3.1, 4.3)
setTemp_a(3.1);
return
setTemp_b(4.3);
return
activate()
//CurrentState
must be Start
activate()
storeData()
getStoreData()
new
// if not null
[storeData]
[this]
storeData()
getTemp_a()
[temp_a]
setRPrice(temp_a)
return
getTemp_b()
[temp_b]
setSPrice(temp_b)
return
return
return
return
currentState = State0
return
return


GasPump1
EFSM
CurrentState
Operation
OperationAbstractFactory
PayMsg1
start()
start()
start()
payMsg()
getPayMsg()
new
// if not null
[payMsg]
[this]
payMsg()
print pay msg
return
return
return
currentState = State1
return
return
payCredit()
payType(1)
payType(1)
return
currentState = State2
return
return

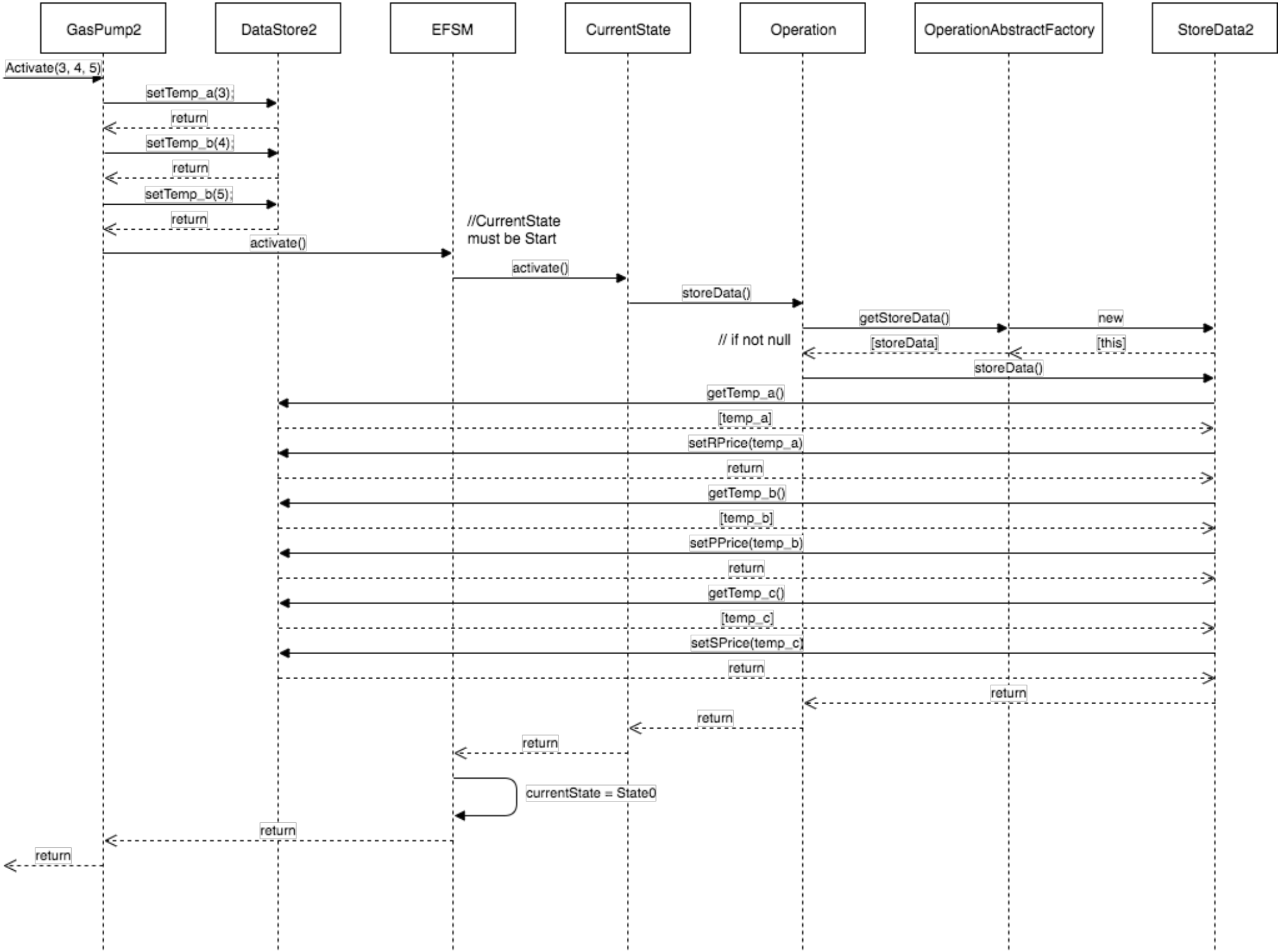

GasPump1
EFSM
CurrentState
Operation
OperationAbstractFactory
DisplayMenu1
SetPrice1
DataStore1
approved()
approved()
approved()
displayMenu()
getDisplayMenu()
new
// if not null
[displayMenu]
[this]
displayMenu()
print menu
return
return
return
currentState = State3
return
return
regular()
selectGas(1)
selectGas(1)
setPrice(1)
getSetPrice(1)
new
// if not null
[setPrice]
[this]
setPrice(g)
getRPrice()
RPrice
setRPrice(RPrice)
return
return
return
return
currentState = State4
return
return

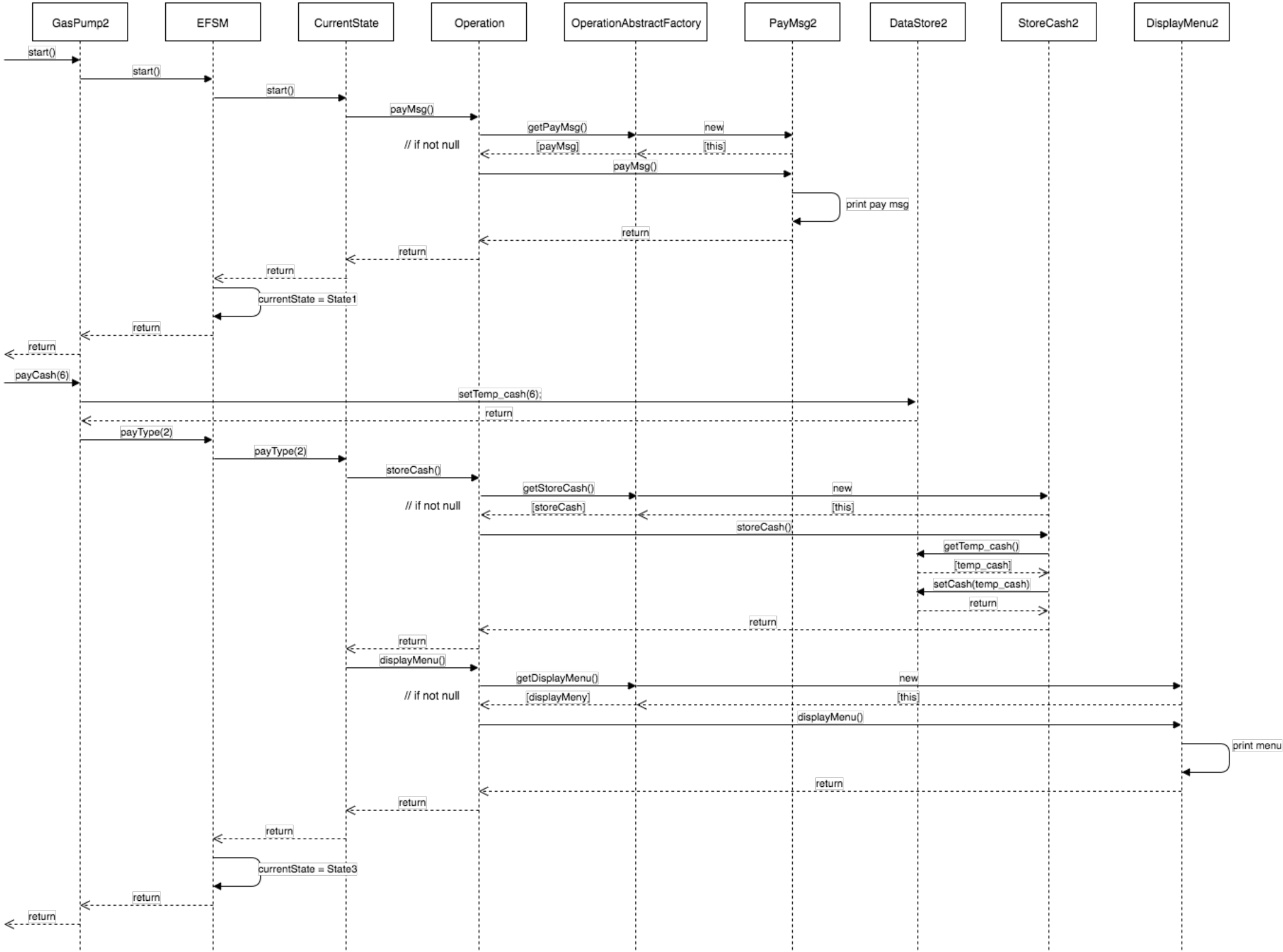

GasPump1
EFSM
CurrentState
Operation
OperationAbstractFactory
SetInitialValues1
DataStore1
ReadyMsg1
startPump()
startPump()
startPump()
setInitialValues()
getSetInitialValues()
new
// if not null
[setInitialValues]
[this]
setInitialValues()
setGallon(0)
return
setTotal(0)
return
return
getReadyMsg()
new
[readyMsg]
[this]
readyMsg()
print ready msg
return
return
return
currentState = State5
return
return

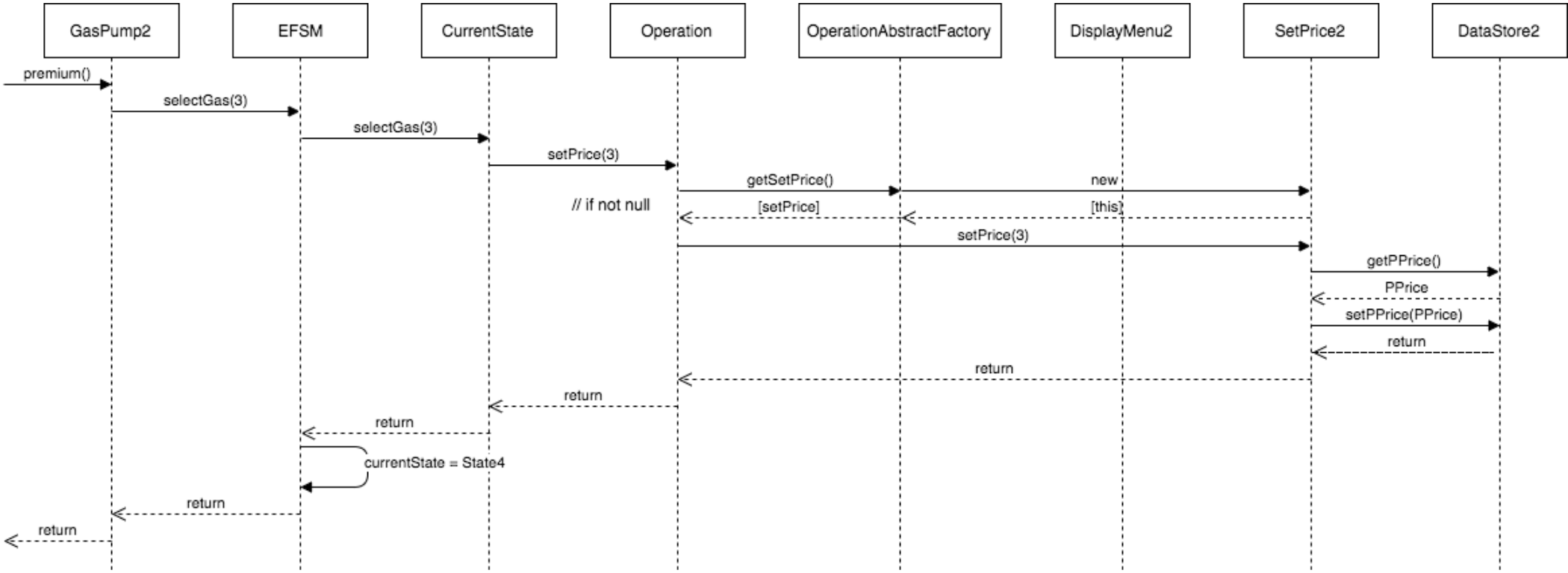

GasPump1
EFSM
CurrentState
Operation
OperationAbstractFactory
PumpGasUnit1
DataStore1
pumpGallon()
pump()
pump()
pumpGasUnit()
getPumpGasUnit()
new
// if not null
[pumpGasUnit]
[this]
pumpGasUnit()
getGallon()
[gallon]
setGallon(gallon+1)
return
getGallon()
[gallon]
getPrice()
[price]
setTotal(gallon*price)
return
return
return
return
return
return

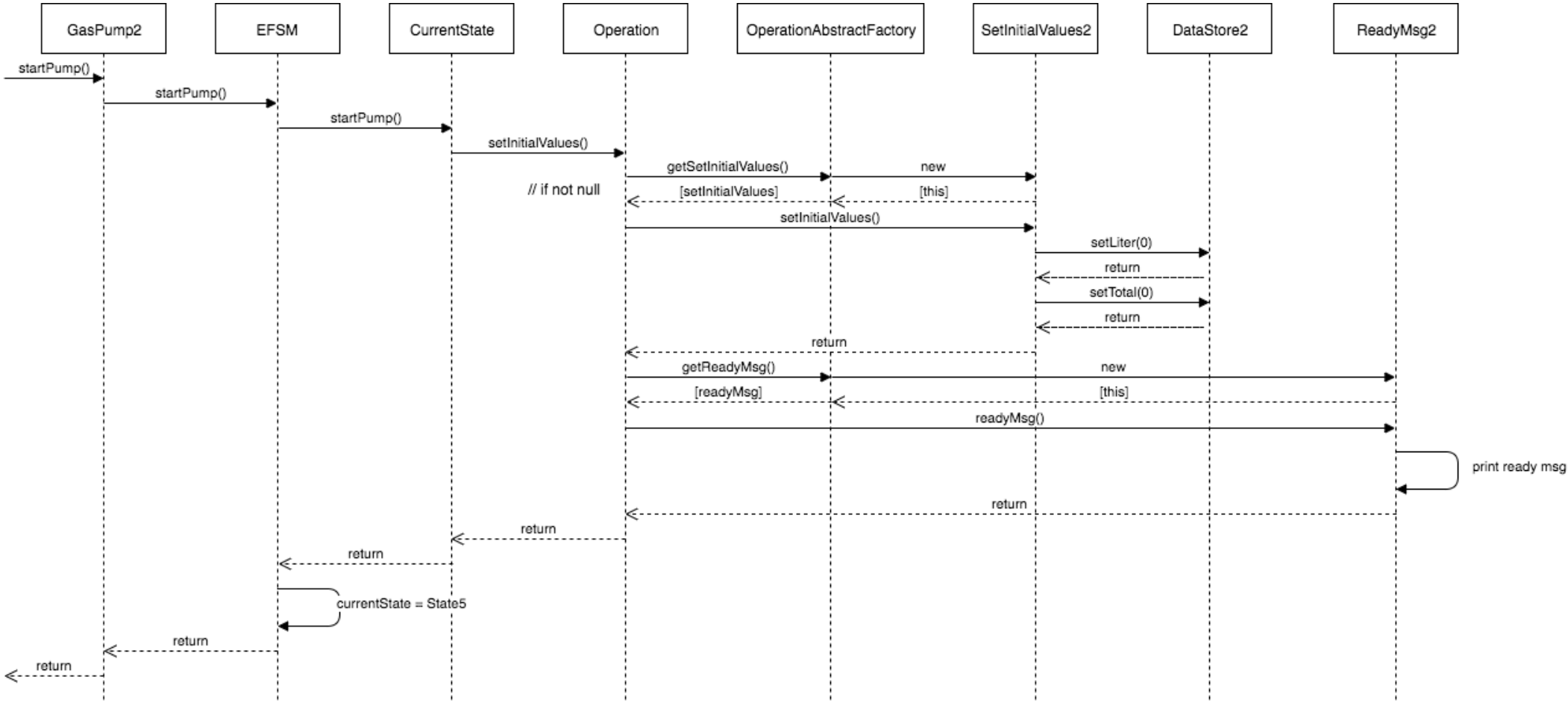

GasPump1
EFSM
CurrentState
Operation
OperationAbstractFactory
PrintReceipt1
DataStore1
stopPump()
stopPump()
startPump()
stopMsg()
getStopMsg()
[null]
return
return
currentState = State7
return
receipt()
receipt()
printReceipt()
getPrintReceipt()
new
// if not null
[printReceipt]
[this]
printReceipt()
getTotal()
[total]
print receipt
return
return
returnCash()
getReturnCash()
[null]
return
return
currentState = State0
return
return

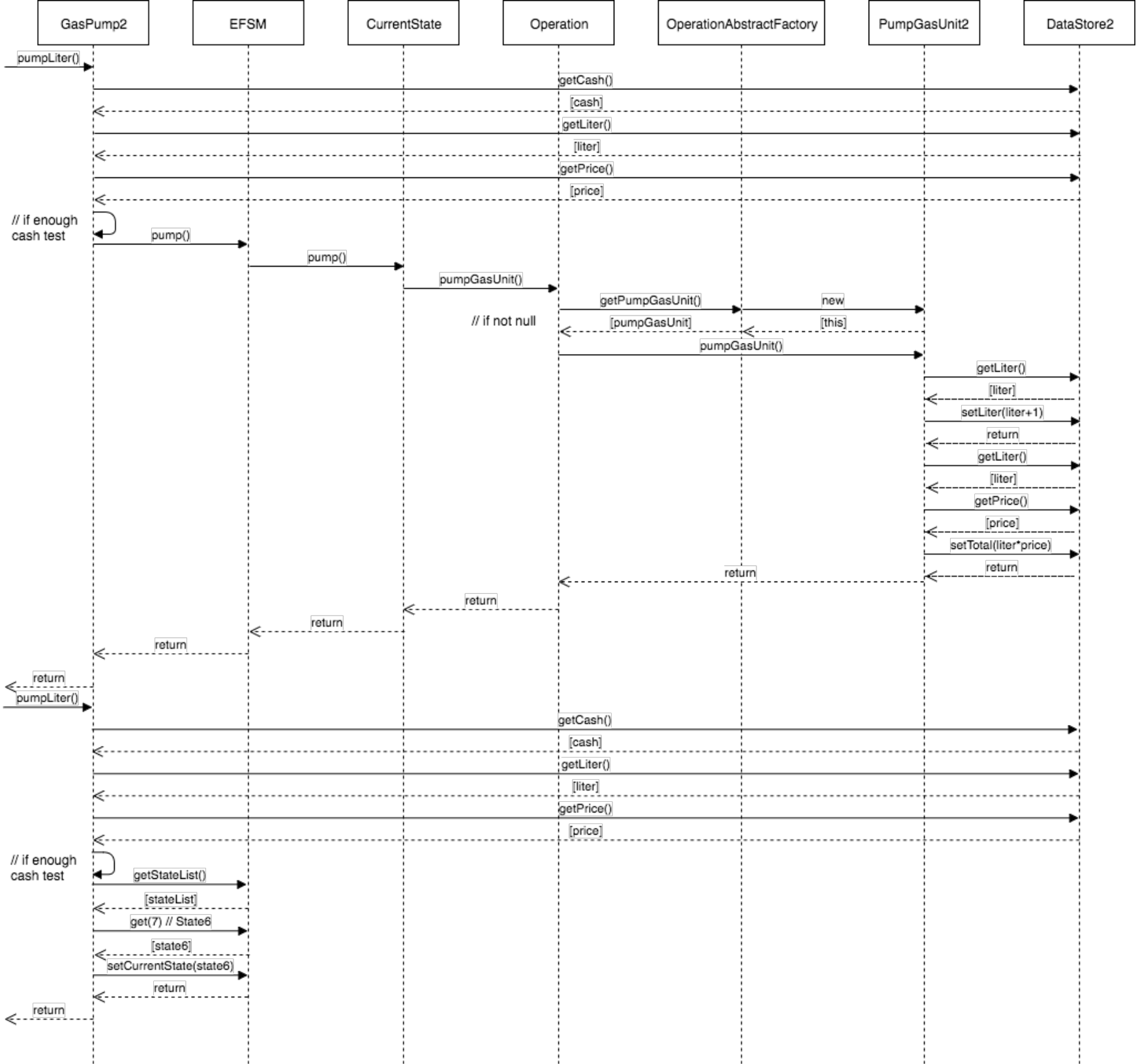
b. *Scenario-II show how one liter of Premium gas is disposed in GasPump-2, i.e., the following sequence of operations is issued: Activate(3, 4, 5), Start(), PayCash(6), Premium(), StartPump(), PumpLiter(), PumpLiter(), NoReceipt()*

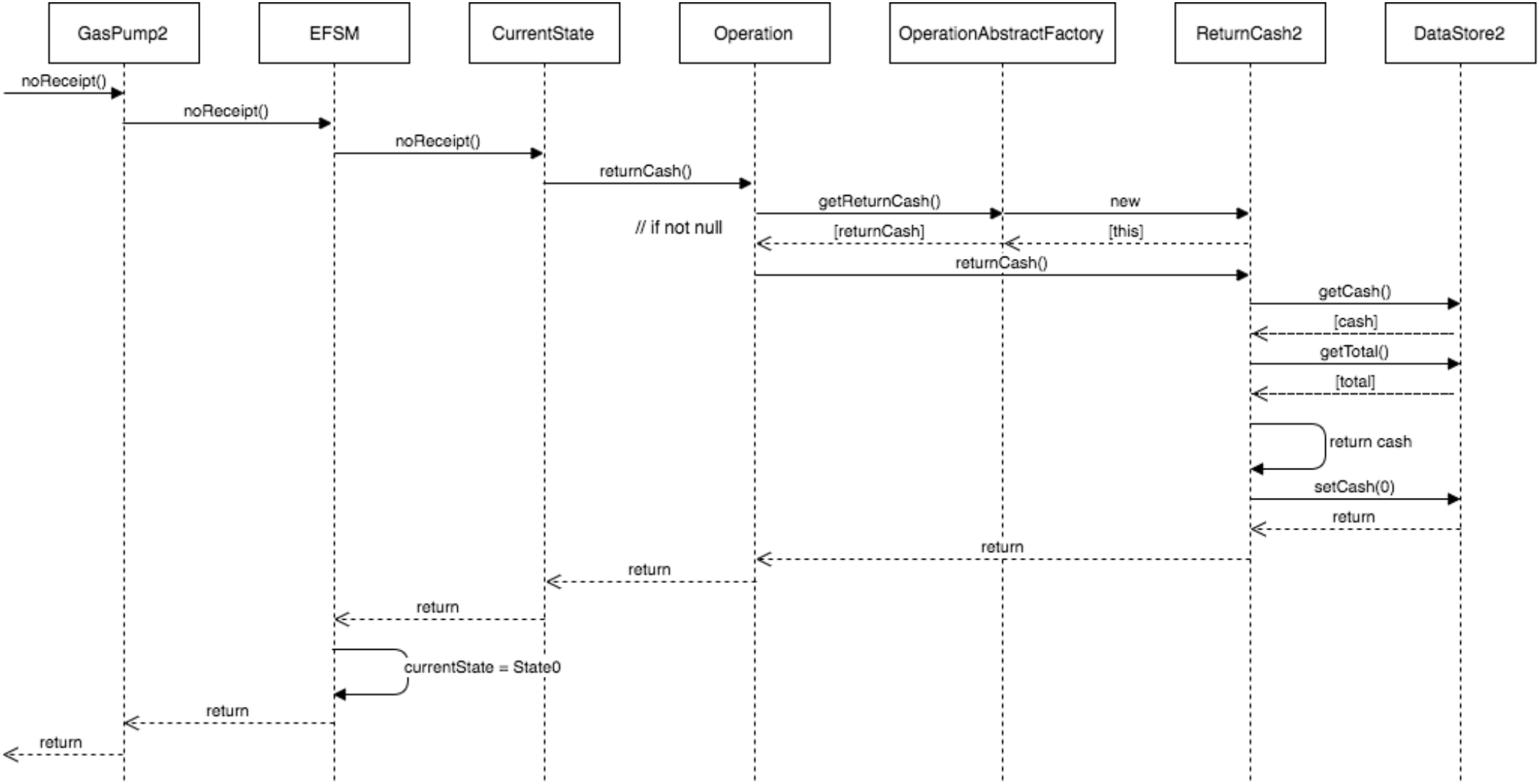

GasPump2
DataStore2
EFSM
CurrentState
Operation
OperationAbstractFactory
StoreData2
Activate(3, 4, 5)
setTemp_a(3);
return
setTemp_b(4);
return
setTemp_b(5);
return
//CurrentState
must be Start
activate()
activate()
storeData()
getStoreData()
new
// if not null
[storeData]
[this]
storeData()
getTemp_a()
[temp_a]
setRPrice(temp_a)
return
getTemp_b()
[temp_b]
setPPrice(temp_b)
return
getTemp_c()
[temp_c]
setSPrice(temp_c)
return
return
return
return
currentState = State0
return
return


GasPump2
EFSM
CurrentState
Operation
OperationAbstractFactory
PayMsg2
DataStore2
StoreCash2
DisplayMenu2
start()
start()
start()
payMsg()
getPayMsg()
new
// if not null
[payMsg]
[this]
payMsg()
print pay msg
return
return
return
currentState = State1
return
return
payCash(6)
setTemp_cash(6);
return
payType(2)
payType(2)
storeCash()
getStoreCash()
new
// if not null
[storeCash]
[this]
storeCash();
getTemp_cash()
[temp_cash]
setCash(temp_cash)
return
return
return
displayMenu()
getDisplayMenu()
new
// if not null
[displayMeny]
[this]
displayMenu()
print menu
return
return
return
currentState = State3
return
return


GasPump2
EFSM
CurrentState
Operation
OperationAbstractFactory
DisplayMenu2
SetPrice2
DataStore2
premium()
selectGas(3)
selectGas(3)
setPrice(3)
getSetPrice()
new
// if not null
[setPrice]
[this]
setPrice(3)
getPPrice()
PPrice
setPPrice(PPrice)
return
return
return
return
currentState = State4
return
return


GasPump2
EFSM
CurrentState
Operation
OperationAbstractFactory
SetInitialValues2
DataStore2
ReadyMsg2
startPump()
startPump()
startPump()
setInitialValues()
getSetInitialValues()
new
// if not null
[setInitialValues]
[this]
setInitialValues()
setLiter(0)
return
setTotal(0)
return
return
getReadyMsg()
new
[readyMsg]
[this]
readyMsg()
print ready msg
return
return
return
currentState = State5
return
return


GasPump2
EFSM
CurrentState
Operation
OperationAbstractFactory
PumpGasUnit2
DataStore2
pumpLiter()
getCash()
[cash]
getLiter()
[liter]
getPrice()
[price]
// if enough cash test
pump()
pump()
pumpGasUnit()
getPumpGasUnit()
new
// if not null
[pumpGasUnit]
[this]
pumpGasUnit()
getLiter()
[liter]
setLiter(liter+1)
return
getLiter()
[liter]
getPrice()
[price]
setTotal(liter*price)
return
return
return
return
return
return
pumpLiter()
getCash()
[cash]
getLiter()
[liter]
getPrice()
[price]
// if enough cash test
getStateList()
[stateList]
get(7) // State6
[state6]
setCurrentState(state6)
return
return


GasPump2
EFSM
CurrentState
Operation
OperationAbstractFactory
ReturnCash2
DataStore2
noReceipt()
noReceipt()
noReceipt()
returnCash()
getReturnCash()
new
// if not null
[returnCash]
[this]
returnCash()
getCash()
[cash]
getTotal()
[total]
return cash
setCash(0)
return
return
return
return
currentState = State0
return
return

5. Source-code and patterns

To make the source code more clear, I also tried to follow the MVC design pattern.

Package iit.softwaresystarch.project.controller is a package containing the classes mostly related to Controller design pattern.

Package iit.softwaresystarch.project.model is a package containing the packages and classes mostly related to Model design pattern.

Package iit.softwaresystarch.project.model.actions is a package containing the classes mostly related to the Strategy design pattern.

Package iit.softwaresystarch.project.model.datastores is a package containing the classes mostly related to both AbstractFactory and Strategy design patterns.

Package iit.softwaresystarch.project.efsms is a package containing the classes mostly related to Model Driven Architecture.

Package iit.softwaresystarch.project.factories is a package containing the classes mostly related to AbstractFactory design pattern.

Package iit.softwaresystarch.project.model.states is a package containing the classes mostly related to State design pattern.

Package iit.softwaresystarch.project.view is a package containing the classes mostly related to View design pattern.

6. Well documented (commented) source code. (Javadoc included)

```
00001: package iit.softwaresystarch.project.controller;
00002:
00003: import java.awt.event.ActionEvent;
00004: import java.awt.event.ActionListener;
00005: import java.io.ByteArrayOutputStream;
00006: import java.io.PrintStream;
00007: import java.nio.charset.StandardCharsets;
00008:
00009: import javax.swing.JTextPane;
00010:
00011: import iit.softwaresystarch.project.model.efsms.GasPump1;
00012: import iit.softwaresystarch.project.view.InputPanel;
00013:
00014: /**
00015:  * The listener interface for receiving buttonActivatePump1Clicked events. The class that is
00016:  * interested in processing a buttonActivatePump1Clicked event implements this interface, and the
00017:  * object created with that class is registered with a component using the component's
00018:  * addButtonActivatePump1ClickedListener method. When the buttonActivatePump1Clicked
00019:  * event occurs, that object's appropriate method is invoked.
00020:  *
00021:  */
00022: public class ButtonActivatePump1ClickedListener implements ActionListener {
00023:
00024:   /** The gas pump 1. */
00025:   private final GasPump1 gasPump1;
00026:
00027:   /** The text pane output. */
00028:   private final JTextPane textPaneOutput;
00029:
00030:   /** The input panel. */
00031:   private final InputPanel inputPanel;
00032:
00033:   /** The orig out. */
```

```
00034:   private PrintStream origOut;
00035:
00036:   /** The baos. */
00037:   private ByteArrayOutputStream baos;
00038:
00039:   /** The out. */
00040:   private PrintStream out;
00041:
00042:   /**
00043:    * Instantiates a new button activate pump 1 clicked listener.
00044:    *
00045:    * @param gasPump1 the gas pump 1
00046:    * @param textPaneOutput the text pane output
00047:    * @param inputPanel the input panel
00048:    */
00049:   public ButtonActivatePump1ClickedListener(GasPump1 gasPump1, JTextPane textPaneOutput,
00050:       InputPanel inputPanel) {
00051:     this.gasPump1 = gasPump1;
00052:     this.textPaneOutput = textPaneOutput;
00053:     this.inputPanel = inputPanel;
00054:   }
00055:
00056:   /*
00057:    * When the button is pressed, the a and b fields are parsed to float and if everything is fine
00058:    * the gas pump 1 is activated with the two floats a and b, else the stack trace is printed and
00059:    * the text field is reset to a default value (1.0).
00060:    *
00061:    * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
00062:    */
00063:   @Override
00064:   public void actionPerformed(ActionEvent e) {
00065:     float a = -1;
00066:     float b = -1;
```

```
00067:     this.startPrint();
00068:     try {
00069:       a = Float.parseFloat(this.inputPanel.getaInput().getText());
00070:       b = Float.parseFloat(this.inputPanel.getbInput().getText());
00071:       this.gasPump1.activate(a, b);
00072:     } catch (final NumberFormatException nfe) {
00073:       if (a == -1) {
00074:         this.inputPanel.getaInput().setText("1.0");
00075:       }
00076:       if (b == -1) {
00077:         this.inputPanel.getbInput().setText("1.0");
00078:       }
00079:       nfe.printStackTrace();
00080:     }
00081:     this.endPrint();
00082:   }
00083:
00084:   /**
00085:    * Redirect the system output to a prinstream.
00086:    */
00087:   private void startPrint() {
00088:     this.origOut = System.out;
00089:     this.baos = new ByteArrayOutputStream();
00090:     this.out = new PrintStream(this.baos);
00091:     System.setOut(this.out);
00092:   }
00093:
00094:   /**
00095:    * Print the printstream into the textPaneOutput.
00096:    */
00097:   private void endPrint() {
00098:     // Update the text field
00099:     this.textPaneOutput.setText(this.textPaneOutput.getText()
```

```
00100:         + new String(this.baos.toByteArray(), StandardCharsets.UTF_8));
00101:     // Get back to the original system output
00102:     System.setOut(this.origOut);
00103:   }
00104:
00105:   /*
00106:    * (non-Javadoc)
00107:    *
00108:    * @see java.lang.Object#toString()
00109:    */
00110:   @Override
00111:   public String toString() {
00112:     return "ButtonActivatePump1ClickedListener [gasPump1=" + this.gasPump1 + ", textPaneOutput="
00113:         + this.textPaneOutput + ", origOut=" + this.origOut + ", baos=" + this.baos + ", out="
00114:         + this.out + "]";
00115:   }
00116:
00117: }
```

```
00001: package iit.softwaresystarch.project.controller;
00002:
00003: import java.awt.event.ActionEvent;
00004: import java.awt.event.ActionListener;
00005: import java.io.ByteArrayOutputStream;
00006: import java.io.PrintStream;
00007: import java.nio.charset.StandardCharsets;
00008:
00009: import javax.swing.JTextPane;
00010:
00011: import iit.softwaresystarch.project.model.efsms.GasPump2;
00012: import iit.softwaresystarch.project.view.InputPanel;
00013:
00014: /**
00015:  * The listener interface for receiving buttonActivatePump2Clicked events. The class that is
00016:  * interested in processing a buttonActivatePump2Clicked event implements this interface, and the
00017:  * object created with that class is registered with a component using the component's
00018:  * addButtonActivatePump2ClickedListener method. When the buttonActivatePump2Clicked
00019:  * event occurs, that object's appropriate method is invoked.
00020:  *
00021:  */
00022: public class ButtonActivatePump2ClickedListener implements ActionListener {
00023:
00024:   /** The gas pump 2. */
00025:   private final GasPump2 gasPump2;
00026:
00027:   /** The text pane output. */
00028:   private final JTextPane textPaneOutput;
00029:
00030:   /** The input panel. */
00031:   private final InputPanel inputPanel;
00032:
00033:   /** The orig out. */
```

```
00034:   private PrintStream origOut;
00035:
00036:   /** The baos. */
00037:   private ByteArrayOutputStream baos;
00038:
00039:   /** The out. */
00040:   private PrintStream out;
00041:
00042:   /**
00043:    * Instantiates a new button activate pump 2 clicked listener.
00044:    *
00045:    * @param gasPump2 the gas pump 2
00046:    * @param textPaneOutput the text pane output
00047:    * @param inputPanel the input panel
00048:    */
00049:   public ButtonActivatePump2ClickedListener(GasPump2 gasPump2, JTextPane textPaneOutput,
00050:       InputPanel inputPanel) {
00051:     this.gasPump2 = gasPump2;
00052:     this.textPaneOutput = textPaneOutput;
00053:     this.inputPanel = inputPanel;
00054:   }
00055:
00056:   /*
00057:    * When the button is pressed, the a and b fields are parsed to float and if everything is fine
00058:    * the gas pump 2 is activated with the three floats a, b and c cast in int else the stack trace
00059:    * is printed and the text field is reset to a default value (1.0).
00060:    *
00061:    * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
00062:    */
00063:   @Override
00064:   public void actionPerformed(ActionEvent e) {
00065:     int a = -1;
00066:     int b = -1;
```

```
00067:     int c = -1;
00068:     this.startPrint();
00069:     try {
00070:       a = (int) Float.parseFloat(this.inputPanel.getaInput().getText());
00071:       b = (int) Float.parseFloat(this.inputPanel.getbInput().getText());
00072:       c = (int) Float.parseFloat(this.inputPanel.getcInput().getText());
00073:       this.gasPump2.activate(a, b, c);
00074:     } catch (final NumberFormatException nfe) {
00075:       if (a == -1) {
00076:         this.inputPanel.getaInput().setText("1.0");
00077:       }
00078:       if (b == -1) {
00079:         this.inputPanel.getbInput().setText("1.0");
00080:       }
00081:       if (c == -1) {
00082:         this.inputPanel.getcInput().setText("1.0");
00083:       }
00084:       nfe.printStackTrace();
00085:     }
00086:     this.endPrint();
00087:   }
00088:
00089:   /**
00090:    * Redirect the system output to a prinstream.
00091:    */
00092:   private void startPrint() {
00093:     this.origOut = System.out;
00094:     this.baos = new ByteArrayOutputStream();
00095:     this.out = new PrintStream(this.baos);
00096:     System.setOut(this.out);
00097:   }
00098:
00099:   /**
```

```
00100:    * Print the printstream into the textPaneOutput.
00101:    */
00102:   private void endPrint() {
00103:     // Update the text field
00104:     this.textPaneOutput.setText(this.textPaneOutput.getText()
00105:         + new String(this.baos.toByteArray(), StandardCharsets.UTF_8));
00106:     // Get back to the original system output
00107:     System.setOut(this.origOut);
00108:   }
00109:
00110:   /*
00111:    * (non-Javadoc)
00112:    *
00113:    * @see java.lang.Object#toString()
00114:    */
00115:   @Override
00116:   public String toString() {
00117:     return "ButtonActivatePump1ClickedListener [gasPump2=" + this.gasPump2 + ", textPaneOutput="
00118:         + this.textPaneOutput + ", origOut=" + this.origOut + ", baos=" + this.baos + ", out="
00119:         + this.out + "]";
00120:   }
00121:
00122: }
```

```
00001: package iit.softwaresystarch.project.controller;
00002:
00003: import java.awt.event.ActionEvent;
00004: import java.awt.event.ActionListener;
00005: import java.io.ByteArrayOutputStream;
00006: import java.io.PrintStream;
00007: import java.nio.charset.StandardCharsets;
00008:
00009: import javax.swing.JTextPane;
00010:
00011: import iit.softwaresystarch.project.model.efsms.GasPump2;
00012: import iit.softwaresystarch.project.view.InputPanel;
00013:
00014: /**
00015:  * The listener interface for receiving buttonCashClicked events. The class that is interested in
00016:  * processing a buttonCashClicked event implements this interface, and the object created with that
00017:  * class is registered with a component using the component's
00018:  * addButtonCashClickedListener method. When the buttonCashClicked event occurs, that
00019:  * object's appropriate method is invoked.
00020:  *
00021:  */
00022: public class ButtonCashClickedListener implements ActionListener {
00023:
00024:   /** The gas pump 2. */
00025:   private final GasPump2 gasPump2;
00026:
00027:   /** The text pane output. */
00028:   private final JTextPane textPaneOutput;
00029:
00030:   /** The input panel. */
00031:   private final InputPanel inputPanel;
00032:
00033:   /** The orig out. */
```

```
00034:   private PrintStream origOut;
00035:
00036:   /** The baos. */
00037:   private ByteArrayOutputStream baos;
00038:
00039:   /** The out. */
00040:   private PrintStream out;
00041:
00042:   /**
00043:    * Instantiates a new button cash clicked listener.
00044:    *
00045:    * @param gasPump2 the gas pump 2
00046:    * @param textPaneOutput the text pane output
00047:    * @param inputPanel the input panel
00048:    */
00049:   public ButtonCashClickedListener(GasPump2 gasPump2, JTextPane textPaneOutput,
00050:       InputPanel inputPanel) {
00051:     this.gasPump2 = gasPump2;
00052:     this.textPaneOutput = textPaneOutput;
00053:     this.inputPanel = inputPanel;
00054:   }
00055:
00056:   /*
00057:    * When the button is pressed, the cash field is parsed to float and if everything is fine the gas
00058:    * pump 2 payCash operation is called with that value else the stack trace is printed and the text
00059:    * field is reset to a default value (1.0).
00060:    *
00061:    * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
00062:    */
00063:   @Override
00064:   public void actionPerformed(ActionEvent e) {
00065:     float cash = -1;
00066:     this.startPrint();
```

```
00067:     try {
00068:       cash = Float.parseFloat(this.inputPanel.getCashInput().getText());
00069:       this.gasPump2.payCash(cash);
00070:     } catch (final NumberFormatException nfe) {
00071:       if (cash == -1) {
00072:         this.inputPanel.getCashInput().setText("1.0");
00073:       }
00074:       nfe.printStackTrace();
00075:     }
00076:     this.endPrint();
00077:   }
00078:
00079:   /**
00080:    * Redirect the system output to a prinstream.
00081:    */
00082:   private void startPrint() {
00083:     this.origOut = System.out;
00084:     this.baos = new ByteArrayOutputStream();
00085:     this.out = new PrintStream(this.baos);
00086:     System.setOut(this.out);
00087:   }
00088:
00089:   /**
00090:    * Print the printstream into the textPaneOutput.
00091:    */
00092:   private void endPrint() {
00093:     // Update the text field
00094:     this.textPaneOutput.setText(this.textPaneOutput.getText()
00095:         + new String(this.baos.toByteArray(), StandardCharsets.UTF_8));
00096:     // Get back to the original system output
00097:     System.setOut(this.origOut);
00098:   }
00099:
```

```
00100:   /*
00101:    * (non-Javadoc)
00102:    *
00103:    * @see java.lang.Object#toString()
00104:    */
00105:   @Override
00106:   public String toString() {
00107:     return "ButtonActivatePump1ClickedListener [gasPump2=" + this.gasPump2 + ", textPaneOutput="
00108:         + this.textPaneOutput + ", origOut=" + this.origOut + ", baos=" + this.baos + ", out="
00109:         + this.out + "]";
00110:   }
00111:
00112: }
```

```
00001: package iit.softwaresystarch.project.controller;
00002:
00003: import java.awt.event.ActionEvent;
00004: import java.awt.event.ActionListener;
00005: import java.io.ByteArrayOutputStream;
00006: import java.io.PrintStream;
00007: import java.lang.reflect.InvocationTargetException;
00008: import java.lang.reflect.Method;
00009: import java.nio.charset.StandardCharsets;
00010:
00011: import javax.swing.JTextPane;
00012:
00013: import iit.softwaresystarch.project.model.efsms.GasPump1;
00014: import iit.softwaresystarch.project.model.efsms.GasPump2;
00015:
00016: /**
00017:  * The listener interface for receiving buttonClicked events. The class that is interested in
00018:  * processing a buttonClicked event implements this interface, and the object created with that
00019:  * class is registered with a component using the component's addButtonClickedListener
00020:  * method. When the buttonClicked event occurs, that object's appropriate method is invoked.
00021:  *
00022:  */
00023: public class ButtonClickedListener implements ActionListener {
00024:
00025:   /** The gas pump 1. */
00026:   private final GasPump1 gasPump1;
00027:
00028:   /** The gas pump 2. */
00029:   private final GasPump2 gasPump2;
00030:
00031:   /** The method string. */
00032:   private final String methodString;
00033:
```

```
00034:   /** The text pane output. */
00035:   private final JTextPane textPaneOutput;
00036:
00037:   /** The orig out. */
00038:   private PrintStream origOut;
00039:
00040:   /** The baos. */
00041:   private ByteArrayOutputStream baos;
00042:
00043:   /** The out. */
00044:   private PrintStream out;
00045:
00046:   /**
00047:    * Instantiates a new button clicked listener.
00048:    *
00049:    * @param object the object which can be either a GasPump1 or a GasPump2
00050:    * @param method the method to be called on click
00051:    * @param textPaneOutput the text pane output
00052:    */
00053:   public ButtonClickedListener(Object object, String method, JTextPane textPaneOutput) {
00054:     if (object instanceof GasPump1) {
00055:       this.gasPump1 = (GasPump1) object;
00056:       this.gasPump2 = null;
00057:     } else if (object instanceof GasPump2) {
00058:       this.gasPump1 = null;
00059:       this.gasPump2 = (GasPump2) object;
00060:     } else {
00061:       this.gasPump1 = null;
00062:       this.gasPump2 = null;
00063:     }
00064:     this.methodString = method;
00065:     this.textPaneOutput = textPaneOutput;
00066:   }
```

```
00067:
00068:   /*
00069:    * When the button is pressed, we find if the button listener was design to handle GasPump1 or
00070:    * GasPump2 input, then we execute the method registred for the listener by using Java Reflection.
00071:    *
00072:    * @see java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
00073:    */
00074:   @Override
00075:   public void actionPerformed(ActionEvent e) {
00076:     if (this.methodString != null) {
00077:       Method method;
00078:       this.startPrint();
00079:       if (this.gasPump1 != null) {
00080:         try {
00081:           method = this.gasPump1.getClass().getMethod(this.methodString);
00082:           method.invoke(this.gasPump1);
00083:         } catch (NoSuchMethodException | SecurityException | IllegalAccessException
00084:             | IllegalArgumentException | InvocationTargetException e1) {
00085:           e1.printStackTrace();
00086:         }
00087:       } else if (this.gasPump2 != null) {
00088:         try {
00089:           method = this.gasPump2.getClass().getMethod(this.methodString);
00090:           method.invoke(this.gasPump2);
00091:         } catch (NoSuchMethodException | SecurityException | IllegalAccessException
00092:             | IllegalArgumentException | InvocationTargetException e1) {
00093:           e1.printStackTrace();
00094:         }
00095:       }
00096:       this.endPrint();
00097:     }
00098:   }
00099:
```

```
00100:   /**
00101:    * Redirect the system output to a prinstream.
00102:    */
00103:   private void startPrint() {
00104:     this.origOut = System.out;
00105:     this.baos = new ByteArrayOutputStream();
00106:     this.out = new PrintStream(this.baos);
00107:     System.setOut(this.out);
00108:   }
00109:
00110:   /**
00111:    * Print the printstream into the textPaneOutput.
00112:    */
00113:   private void endPrint() {
00114:     // Update the text field
00115:     this.textPaneOutput.setText(this.textPaneOutput.getText()
00116:         + new String(this.baos.toByteArray(), StandardCharsets.UTF_8));
00117:     // Get back to the original system output
00118:     System.setOut(this.origOut);
00119:   }
00120:
00121:   /*
00122:    * (non-Javadoc)
00123:    *
00124:    * @see java.lang.Object#toString()
00125:    */
00126:   @Override
00127:   public String toString() {
00128:     return "ButtonClickedListener [gasPump1=" + this.gasPump1 + ", gasPump2=" + this.gasPump2
00129:         + ", methodString=" + this.methodString + ", textPaneOutput=" + this.textPaneOutput
00130:         + ", origOut=" + this.origOut + ", baos=" + this.baos + ", out=" + this.out + "]";
00131:   }
00132:
```

00133: }

```
00001: /**
00002:  * is a package containing the classes mostly related to Controller design pattern.
00003:  */
00004: package iit.softwaresystarch.project.controller;
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The abstract Class CancelMsg.
00005:  * This Class is used as an Interface for the Strategy pattern.
00006:  * The goal is to have multiple implemtation of the method cancelMsg.
00007:  */
00008: public abstract class CancelMsg {
00009:
00010:   /**
00011:    * Instantiates a new CancelMsg.
00012:    */
00013:   public CancelMsg() {}
00014:
00015:   /**
00016:    * Displays a cancel message.
00017:    */
00018:   public void cancelMsg() {}
00019:
00020: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The Class CancelMsg1.
00005:  * This Class is the implementation of the Interface CancelMsg for the GasPump1.
00006:  */
00007: public class CancelMsg1 extends CancelMsg {
00008:
00009:   /**
00010:    * Instantiates a new CancelMsg1.
00011:    */
00012:   public CancelMsg1() {}
00013:
00014:   /*
00015:    * (non-Javadoc)
00016:    *
00017:    * @see iit.softwaresystarch.project.model.actions.CancelMsg#cancelMsg()
00018:    */
00019:   @Override
00020:   public void cancelMsg() {
00021:     System.out.println("Cancel message for pump 1");
00022:   }
00023:
00024: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The Class CancelMsg2.
00005:  * This Class is the implementation of the Interface CancelMsg for the GasPump2.
00006:  */
00007: public class CancelMsg2 extends CancelMsg {
00008:
00009:   /**
00010:    * Instantiates a new CancelMsg2.
00011:    */
00012:   public CancelMsg2() {}
00013:
00014:   /*
00015:    * (non-Javadoc)
00016:    *
00017:    * @see iit.softwaresystarch.project.model.actions.CancelMsg#cancelMsg()
00018:    */
00019:   @Override
00020:   public void cancelMsg() {
00021:     System.out.println("Cancel message for pump 2");
00022:   }
00023:
00024: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The anstract Class DisplayMenu.
00005:  * This Class is used as an Interface for the Strategy pattern.
00006:  * The goal is to have multiple implemtation of the method displayMenu.
00007:  */
00008: public abstract class DisplayMenu {
00009:
00010:   /**
00011:    * Instantiates a new DisplayMenu.
00012:    */
00013:   public DisplayMenu() {}
00014:
00015:   /**
00016:    * Displays a menu.
00017:    */
00018:   public void displayMenu() {}
00019:
00020: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The Class DisplayMenu1.
00005:  * This Class is the implementation of the Interface DisplayMenu for the GasPump1.
00006:  */
00007: public class DisplayMenu1 extends DisplayMenu {
00008:
00009:   /**
00010:    * Instantiates a new DisplayMenu1.
00011:    */
00012:   public DisplayMenu1() {}
00013:
00014:   /*
00015:    * (non-Javadoc)
00016:    *
00017:    * @see iit.softwaresystarch.project.model.actions.DisplayMenu#displayMenu()
00018:    */
00019:   @Override
00020:   public void displayMenu() {
00021:     System.out.println("Display menu 1");
00022:     System.out.println("Please select Regular or Super");
00023:   }
00024:
00025: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The Class DisplayMenu2.
00005:  * This Class is the implementation of the Interface DisplayMenu for the GasPump2.
00006:  */
00007: public class DisplayMenu2 extends DisplayMenu {
00008:
00009:   /**
00010:    * Instantiates a new DisplayMenu2.
00011:    */
00012:   public DisplayMenu2() {}
00013:
00014:   /*
00015:    * (non-Javadoc)
00016:    *
00017:    * @see iit.softwaresystarch.project.model.actions.DisplayMenu#displayMenu()
00018:    */
00019:   @Override
00020:   public void displayMenu() {
00021:     System.out.println("Display menu 2");
00022:     System.out.println("Please select Regular, Super or Premium");
00023:   }
00024:
00025: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The abstract Class GasPumpedMsg.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method gasPumpedMsg.
00009:  */
00010: public abstract class GasPumpedMsg {
00011:
00012:   /** The data store. */
00013:   protected final DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new GasPumpedMsg.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public GasPumpedMsg(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Displays a message when gas is pumped.
00026:    */
00027:   public void gasPumpedMsg() {}
00028:
00029: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore1;
00004:
00005: /**
00006:  * The Class GasPumpedMsg1.
00007:  * This Class is the implementation of the Interface GasPumpedMsg for the GasPump1.
00008:  */
00009: public class GasPumpedMsg1 extends GasPumpedMsg {
00010:
00011:   /**
00012:    * Instantiates a new GasPumpedMsg1.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public GasPumpedMsg1(DataStore1 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Displays the gallon pumped.
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.GasPumpedMsg#gasPumpedMsg()
00024:    */
00025:   @Override
00026:   public void gasPumpedMsg() {
00027:     System.out.println("Gas Pumped Message 1");
00028:     System.out.println("Gallon is:" + ((DataStore1) this.dataStore).getGallon() + ".");
00029:   }
00030:
00031: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class GasPumpedMsg2.
00007:  * This Class is the implementation of the Interface GasPumpedMsg for the GasPump2.
00008:  */
00009: public class GasPumpedMsg2 extends GasPumpedMsg {
00010:
00011:   /**
00012:    * Instantiates a new GasPumpedMsg2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public GasPumpedMsg2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Displays the liter pumped.
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.GasPumpedMsg#gasPumpedMsg()
00024:    */
00025:   @Override
00026:   public void gasPumpedMsg() {
00027:     System.out.println("Gas Pumped Message 2");
00028:     System.out.println("Liter is:" + ((DataStore2) this.dataStore).getLiter() + ".");
00029:   }
00030:
00031: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class Operation responsible for executing the operations build by the factory.
00007:  */
00008: public class Operation {
00009:
00010:   /** The operation abstract factory. */
00011:   private final OperationAbstractFactory operationAbstractFactory;
00012:
00013:   /**
00014:    * Instantiates a new Operation.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public Operation(OperationAbstractFactory operationAbstractFactory) {
00019:     this.operationAbstractFactory = operationAbstractFactory;
00020:   }
00021:
00022:
00023:   /**
00024:    * Stores data if the factory has an implementation.
00025:    */
00026:   public void storeData() {
00027:     final StoreData storeData = this.operationAbstractFactory.getStoreData();
00028:     if (storeData != null) {
00029:       storeData.storeData();
00030:     }
00031:   }
00032:
00033:   /**
```

```
00034:    * Displays the pay message if the factory has an implementation.
00035:    */
00036:   public void payMsg() {
00037:     final PayMsg payMsg = this.operationAbstractFactory.getPayMsg();
00038:     if (payMsg != null) {
00039:       payMsg.payMsg();
00040:     }
00041:   }
00042:
00043:   /**
00044:    * Stores cash if the factory has an implementation.
00045:    */
00046:   public void storeCash() {
00047:     final StoreCash storeCash = this.operationAbstractFactory.getStoreCash();
00048:     if (storeCash != null) {
00049:       storeCash.storeCash();
00050:     }
00051:   }
00052:
00053:   /**
00054:    * Displays a menu if the factory has an implementation.
00055:    */
00056:   public void displayMenu() {
00057:     final DisplayMenu displayMenu = this.operationAbstractFactory.getDisplayMenu();
00058:     if (displayMenu != null) {
00059:       displayMenu.displayMenu();
00060:     }
00061:   }
00062:
00063:   /**
00064:    * Displays a reject message if the factory has an implementation.
00065:    */
00066:   public void rejectMsg() {
```

```
00067:     final RejectMsg rejectMsg = this.operationAbstractFactory.getRejectMsg();
00068:     if (rejectMsg != null) {
00069:       rejectMsg.rejectMsg();
00070:     }
00071:   }
00072:
00073:   /**
00074:    * Displays a cancel message if the factory has an implementation.
00075:    */
00076:   public void cancelMsg() {
00077:     final CancelMsg cancelMsg = this.operationAbstractFactory.getCancelMsg();
00078:     if (cancelMsg != null) {
00079:       cancelMsg.cancelMsg();
00080:     }
00081:   }
00082:
00083:   /**
00084:    * Sets the price if the factory has an implementation.
00085:    *
00086:    * @param g the new price
00087:    */
00088:   public void setPrice(int g) {
00089:     final SetPrice setPrice = this.operationAbstractFactory.getSetPrice(g);
00090:     if (setPrice != null) {
00091:       setPrice.setPrice(g);
00092:     }
00093:   }
00094:
00095:   /**
00096:    * Displays a ready message if the factory has an implementation.
00097:    */
00098:   public void readyMsg() {
00099:     final ReadyMsg readyMsg = this.operationAbstractFactory.getReadyMsg();
```

```
00100:     if (readyMsg != null) {
00101:       readyMsg.readyMsg();
00102:     }
00103:   }
00104:
00105:   /**
00106:    * Sets the initial values if the factory has an implementation.
00107:    */
00108:   public void setInitialValues() {
00109:     final SetInitialValues setInitialValues = this.operationAbstractFactory.getSetInitialValues();
00110:     if (setInitialValues != null) {
00111:       setInitialValues.setInitialValues();
00112:     }
00113:   }
00114:
00115:   /**
00116:    * Pumps a gas unit if the factory has an implementation.
00117:    */
00118:   public void pumpGasUnit() {
00119:     final PumpGasUnit pumpGasUnit = this.operationAbstractFactory.getPumpGasUnit();
00120:     if (pumpGasUnit != null) {
00121:       pumpGasUnit.pumpGasUnit();
00122:     }
00123:   }
00124:
00125:   /**
00126:    * Displays a gas pumped message if the factory has an implementation.
00127:    */
00128:   public void gasPumpedMsg() {
00129:     final GasPumpedMsg gasPumpedMsg = this.operationAbstractFactory.getGasPumpedMsg();
00130:     if (gasPumpedMsg != null) {
00131:       gasPumpedMsg.gasPumpedMsg();
00132:     }
```

```
00133:   }
00134:
00135:   /**
00136:    * Displays a stop message if the factory has an implementation.
00137:    */
00138:   public void stopMsg() {
00139:     final StopMsg stopMsg = this.operationAbstractFactory.getStopMsg();
00140:     if (stopMsg != null) {
00141:       stopMsg.stopMsg();
00142:     }
00143:   }
00144:
00145:   /**
00146:    * Prints the receipt if the factory has an implementation.
00147:    */
00148:   public void printReceipt() {
00149:     final PrintReceipt printReceipt = this.operationAbstractFactory.getPrintReceipt();
00150:     if (printReceipt != null) {
00151:       printReceipt.printReceipt();
00152:     }
00153:   }
00154:
00155:   /**
00156:    * Returns the cash if the factory has an implementation.
00157:    */
00158:   public void returnCash() {
00159:     final ReturnCash returnCash = this.operationAbstractFactory.getReturnCash();
00160:     if (returnCash != null) {
00161:       returnCash.returnCash();
00162:     }
00163:   }
00164:
00165:   /*
```

```
00166:    * (non-Javadoc)
00167:    *
00168:    * @see java.lang.Object#toString()
00169:    */
00170:   @Override
00171:   public String toString() {
00172:     return "Operation [operationAbstractFactory=" + this.operationAbstractFactory + "]";
00173:   }
00174:
00175: }
```

```
00001: /**
00002:  * is a package containing the classes mostly related to the Strategy design pattern.
00003:  */
00004: package iit.softwaresystarch.project.model.actions;
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The abstract Class PayMsg.
00005:  * This Class is used as an Interface for the Strategy pattern.
00006:  * The goal is to have multiple implemtation of the method payMsg.
00007:  */
00008: public abstract class PayMsg {
00009:
00010:   /**
00011:    * Instantiates a new PayMsg.
00012:    */
00013:   public PayMsg() {}
00014:
00015:   /**
00016:    * Displays a pay message.
00017:    */
00018:   public void payMsg() {}
00019:
00020: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The Class PayMsg1.
00005:  * This Class is the implementation of the Interface PayMsg for the GasPump1.
00006:  */
00007: public class PayMsg1 extends PayMsg {
00008:
00009:   /**
00010:    * Instantiates a new PayMsg1.
00011:    */
00012:   public PayMsg1() {}
00013:
00014:   /*
00015:    * (non-Javadoc)
00016:    *
00017:    * @see iit.softwaresystarch.project.model.actions.PayMsg#payMsg()
00018:    */
00019:   @Override
00020:   public void payMsg() {
00021:     System.out.println("Pay Message 1");
00022:     System.out.println("Please select pay credit");
00023:   }
00024:
00025: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The Class PayMsg2.
00005:  * This Class is the implementation of the Interface PayMsg for the GasPump2.
00006:  */
00007: public class PayMsg2 extends PayMsg {
00008:
00009:   /**
00010:    * Instantiates a new PayMsg2.
00011:    */
00012:   public PayMsg2() {}
00013:
00014:   /*
00015:    * (non-Javadoc)
00016:    *
00017:    * @see iit.softwaresystarch.project.model.actions.PayMsg#payMsg()
00018:    */
00019:   @Override
00020:   public void payMsg() {
00021:     System.out.println("Pay Message 2");
00022:     System.out.println("Please select pay cash");
00023:   }
00024:
00025: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The abstract Class PrintReceipt.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method printReceipt.
00009:  */
00010: public abstract class PrintReceipt {
00011:
00012:   /** The data store. */
00013:   protected final DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new PrintReceipt.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public PrintReceipt(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Prints the receipt.
00026:    */
00027:   public void printReceipt() {}
00028:
00029: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore1;
00004:
00005: /**
00006:  * The Class PrintReceipt1.
00007:  * This Class is the implementation of the Interface PrintReceipt for the GasPump1.
00008:  */
00009: public class PrintReceipt1 extends PrintReceipt {
00010:
00011:   /**
00012:    * Instantiates a new PrintReceipt1.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public PrintReceipt1(DataStore1 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * (non-Javadoc)
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.PrintReceipt#printReceipt()
00024:    */
00025:   @Override
00026:   public void printReceipt() {
00027:     System.out.println("Print Receipt 1");
00028:     System.out.println("Price is: " + ((DataStore1) this.dataStore).getTotal() + ".");
00029:   }
00030:
00031: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class PrintReceipt2.
00007:  * This Class is the implementation of the Interface PrintReceipt for the GasPump2.
00008:  */
00009: public class PrintReceipt2 extends PrintReceipt {
00010:
00011:   /**
00012:    * Instantiates a new PrintReceipt2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public PrintReceipt2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * (non-Javadoc)
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.PrintReceipt#printReceipt()
00024:    */
00025:   @Override
00026:   public void printReceipt() {
00027:     System.out.println("Print Receipt 2");
00028:     System.out.println("Price is: " + ((DataStore2) this.dataStore).getTotal() + ", liter is: "
00029:         + ((DataStore2) this.dataStore).getLiter() + ".");
00030:   }
00031:
00032: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The abstract Class PumpGasUnit.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method pumpGasUnit.
00009:  */
00010: public abstract class PumpGasUnit {
00011:
00012:   /** The data store. */
00013:   protected final DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new PumpGasUnit.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public PumpGasUnit(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Pumps a gas unit.
00026:    */
00027:   public void pumpGasUnit() {}
00028:
00029: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore1;
00004:
00005: /**
00006:  * The Class PumpGasUnit1.
00007:  * This Class is the implementation of the Interface PumpGasUnit for the GasPump1.
00008:  */
00009: public class PumpGasUnit1 extends PumpGasUnit {
00010:
00011:   /**
00012:    * Instantiates a new PumpGasUnit1.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public PumpGasUnit1(DataStore1 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Updates the gallon amount and the corresponding total price.
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.PumpGasUnit#pumpGasUnit()
00024:    */
00025:   @Override
00026:   public void pumpGasUnit() {
00027:     System.out.println("Pump Gas Unit 1");
00028:     ((DataStore1) this.dataStore).setGallon(((DataStore1) this.dataStore).getGallon() + 1);
00029:     ((DataStore1) this.dataStore).setTotal(
00030:         ((DataStore1) this.dataStore).getGallon() * (((DataStore1) this.dataStore).getPrice()));
00031:   }
00032: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class PumpGasUnit2.
00007:  * This Class is the implementation of the Interface PumpGasUnit for the GasPump2.
00008:  */
00009: public class PumpGasUnit2 extends PumpGasUnit {
00010:
00011:   /**
00012:    * Instantiates a new PumpGasUnit2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public PumpGasUnit2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Updates the liter amount and the corresponding total price.
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.PumpGasUnit#pumpGasUnit()
00024:    */
00025:   @Override
00026:   public void pumpGasUnit() {
00027:     System.out.println("Pump Gas Unit 2");
00028:     ((DataStore2) this.dataStore).setLiter(((DataStore2) this.dataStore).getLiter() + 1);
00029:     ((DataStore2) this.dataStore).setTotal(
00030:         ((DataStore2) this.dataStore).getLiter() * (((DataStore2) this.dataStore).getPrice()));
00031:   }
00032:
00033: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The abstract Class ReadyMsg.
00005:  * This Class is used as an Interface for the Strategy pattern.
00006:  * The goal is to have multiple implemtation of the method readyMsg.
00007:  */
00008: public abstract class ReadyMsg {
00009:
00010:   /**
00011:    * Instantiates a new ReadyMsg.
00012:    */
00013:   public ReadyMsg() {}
00014:
00015:   /**
00016:    * Displays a ready message.
00017:    */
00018:   public void readyMsg() {}
00019:
00020: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The abstract Class RejectMsg.
00005:  * This Class is used as an Interface for the Strategy pattern.
00006:  * The goal is to have multiple implemtation of the method rejectMsg.
00007:  */
00008: public abstract class RejectMsg {
00009:
00010:   /**
00011:    * Instantiates a new RejectMsg.
00012:    */
00013:   public RejectMsg() {}
00014:
00015:   /**
00016:    * Displays a reject message.
00017:    */
00018:   public void rejectMsg() {}
00019:
00020: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The Class RejectMsg1.
00005:  * This Class is the implementation of the Interface RejectMsg for the GasPump1.
00006:  */
00007: public class RejectMsg1 extends RejectMsg {
00008:
00009:   /**
00010:    * Instantiates a new RejectMsg1.
00011:    */
00012:   public RejectMsg1() {}
00013:
00014:   /*
00015:    * (non-Javadoc)
00016:    *
00017:    * @see iit.softwaresystarch.project.model.actions.RejectMsg#rejectMsg()
00018:    */
00019:   @Override
00020:   public void rejectMsg() {
00021:     System.out.println("Reject Message 1");
00022:     System.out.println("Sorry, you can't pay by credit!");
00023:   }
00024:
00025: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The Class ReturnCash.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method returnCash.
00009:  */
00010: public abstract class ReturnCash {
00011:
00012:   /** The data store. */
00013:   protected final DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new ReturnCash.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public ReturnCash(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Returns the cash.
00026:    */
00027:   public void returnCash() {}
00028:
00029: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class ReturnCash2.
00007:  * This Class is the implementation of the Interface ReturnCash for the GasPump2.
00008:  */
00009: public class ReturnCash2 extends ReturnCash {
00010:
00011:   /**
00012:    * Instantiates a new ReturnCash2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public ReturnCash2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Sets the current cash to 0 and displays the return value
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.ReturnCash#returnCash()
00024:    */
00025:   @Override
00026:   public void returnCash() {
00027:     System.out.println("Return Cash 2");
00028:     System.out.println(
00029:         "Returning :" + (((DataStore2) this.dataStore).getCash() - this.dataStore.getTotal()));
00030:     ((DataStore2) this.dataStore).setCash(0);
00031:   }
00032:
00033: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The abstract Class SetInitialValues.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method setInitialValues.
00009:  */
00010: public abstract class SetInitialValues {
00011:
00012:   /** The data store. */
00013:   protected final DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new SetInitialValues.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public SetInitialValues(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Sets the initial values.
00026:    */
00027:   public void setInitialValues() {}
00028:
00029: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore1;
00004:
00005: /**
00006:  * The Class SetInitialValues1.
00007:  * This Class is the implementation of the Interface SetInitialValues for the GasPump1.
00008:  */
00009: public class SetInitialValues1 extends SetInitialValues {
00010:
00011:   /**
00012:    * Instantiates a new SetInitialValues1.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public SetInitialValues1(DataStore1 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Sets the total to pay to 0 and the pumped gallon to 0
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.SetInitialValues#setInitialValues()
00024:    */
00025:   @Override
00026:   public void setInitialValues() {
00027:     System.out.println("Set Initial Values 1");
00028:     ((DataStore1) this.dataStore).setGallon(0);
00029:     ((DataStore1) this.dataStore).setTotal(0);
00030:   }
00031:
00032: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class SetInitialValues2.
00007:  * This Class is the implementation of the Interface SetInitialValues for the GasPump2.
00008:  */
00009: public class SetInitialValues2 extends SetInitialValues {
00010:
00011:   /**
00012:    * Instantiates a new SetInitialValues2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public SetInitialValues2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Sets the total to pay to 0 and the pumped liter to 0
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.SetInitialValues#setInitialValues()
00024:    */
00025:   @Override
00026:   public void setInitialValues() {
00027:     System.out.println("Set Initial Values 2");
00028:     ((DataStore2) this.dataStore).setLiter(0);
00029:     ((DataStore2) this.dataStore).setTotal(0);
00030:   }
00031:
00032: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The abstract Class SetPrice.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method setPrice(int g).
00009:  */
00010: public abstract class SetPrice {
00011:
00012:   /** The data store. */
00013:   protected final DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new SetPrice.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public SetPrice(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Sets the price.
00026:    *
00027:    * @param g the new price
00028:    */
00029:   public void setPrice(int g) {}
00030:
00031: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore1;
00004:
00005: /**
00006:  * The Class SetPrice1.
00007:  * This Class is the implementation of the Interface SetPrice for the GasPump1.
00008:  */
00009: public class SetPrice1 extends SetPrice {
00010:
00011:   /**
00012:    * Instantiates a new SetPrice1.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public SetPrice1(DataStore1 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * If g == 1, then price = RPrice, if g == 2, then price = SPrice else nothing
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.SetPrice#setPrice(int)
00024:    */
00025:   @Override
00026:   public void setPrice(int g) {
00027:     System.out.println("Set Price 1");
00028:     switch (g) {
00029:       case 1:
00030:         this.dataStore.setPrice(((DataStore1) this.dataStore).getRPrice());
00031:         break;
00032:       case 2:
00033:         this.dataStore.setPrice(((DataStore1) this.dataStore).getSPrice());
```

```
00034:         break;
00035:       default:
00036:         break;
00037:     }
00038:   }
00039:
00040: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class SetPrice2.
00007:  * This Class is the implementation of the Interface SetPrice for the GasPump2.
00008:  */
00009: public class SetPrice2 extends SetPrice {
00010:
00011:   /**
00012:    * Instantiates a new SetPrice2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public SetPrice2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * If g == 1, then price = RPrice, if g == 2, then price = SPrice if g == 3, then price = PPrice
00022:    * else nothing
00023:    *
00024:    * @see iit.softwaresystarch.project.model.actions.SetPrice#setPrice(int)
00025:    */
00026:   @Override
00027:   public void setPrice(int g) {
00028:     System.out.println("Set Price 2");
00029:     switch (g) {
00030:       case 1:
00031:         this.dataStore.setPrice(((DataStore2) this.dataStore).getRPrice());
00032:         break;
00033:       case 2:
```

```
00034:         this.dataStore.setPrice(((DataStore2) this.dataStore).getSPrice());
00035:         break;
00036:       case 3:
00037:         this.dataStore.setPrice(((DataStore2) this.dataStore).getPPrice());
00038:         break;
00039:       default:
00040:         break;
00041:     }
00042:   }
00043:
00044: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: /**
00004:  * The abstract Class StopMsg.
00005:  * This Class is used as an Interface for the Strategy pattern.
00006:  * The goal is to have multiple implemtation of the method stopMsg.
00007:  */
00008: public abstract class StopMsg {
00009:
00010:   /**
00011:    * Instantiates a new StopMsg.
00012:    */
00013:   public StopMsg() {}
00014:
00015:   /**
00016:    * Displays a stop message.
00017:    */
00018:   public void stopMsg() {}
00019:
00020: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The abstract Class StoreCash.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method storeCash.
00009:  */
00010: public abstract class StoreCash {
00011:
00012:   /** The data store. */
00013:   protected final DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new StoreCash.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public StoreCash(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Stores the cash.
00026:    */
00027:   public void storeCash() {}
00028:
00029: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class StoreCash2.
00007:  * This Class is the implementation of the Interface StoreCash for the GasPump2.
00008:  */
00009: public class StoreCash2 extends StoreCash {
00010:
00011:   /**
00012:    * Instantiates a new StoreCash2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public StoreCash2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Sets the cash value to temp_cash.
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.StoreCash#storeCash()
00024:    */
00025:   @Override
00026:   public void storeCash() {
00027:     System.out.println("Store Cash 2");
00028:     ((DataStore2) this.dataStore).setCash(((DataStore2) this.dataStore).getTemp_cash());
00029:   }
00030:
00031: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004:
00005: /**
00006:  * The abstract Class StoreData.
00007:  * This Class is used as an Interface for the Strategy pattern.
00008:  * The goal is to have multiple implemtation of the method storeData.
00009:  */
00010: public abstract class StoreData {
00011:
00012:   /** The data store. */
00013:   protected DataStore dataStore;
00014:
00015:   /**
00016:    * Instantiates a new StoreData.
00017:    *
00018:    * @param dataStore the data store
00019:    */
00020:   public StoreData(DataStore dataStore) {
00021:     this.dataStore = dataStore;
00022:   }
00023:
00024:   /**
00025:    * Stores the data.
00026:    */
00027:   public void storeData() {}
00028:
00029: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore1;
00004:
00005: /**
00006:  * The Class StoreData1.
00007:  * This Class is the implementation of the Interface StoreData for the GasPump1.
00008:  */
00009: public class StoreData1 extends StoreData {
00010:
00011:   /**
00012:    * Instantiates a new StoreData1.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public StoreData1(DataStore1 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Sets the RPrice to temp_a, SPrice to temp_b.
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.StoreData#storeData()
00024:    */
00025:   @Override
00026:   public void storeData() {
00027:     System.out.println("Store Data 1");
00028:     ((DataStore1) this.dataStore).setRPrice(((DataStore1) this.dataStore).getTemp_a());
00029:     ((DataStore1) this.dataStore).setSPrice(((DataStore1) this.dataStore).getTemp_b());
00030:   }
00031:
00032: }
```

```
00001: package iit.softwaresystarch.project.model.actions;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore2;
00004:
00005: /**
00006:  * The Class StoreData2.
00007:  * This Class is the implementation of the Interface StoreData for the GasPump2.
00008:  */
00009: public class StoreData2 extends StoreData {
00010:
00011:   /**
00012:    * Instantiates a new StoreData2.
00013:    *
00014:    * @param dataStore the data store
00015:    */
00016:   public StoreData2(DataStore2 dataStore) {
00017:     super(dataStore);
00018:   }
00019:
00020:   /*
00021:    * Sets the RPrice to temp_a, PPrice to temp_b, SPrice to temp_c.
00022:    *
00023:    * @see iit.softwaresystarch.project.model.actions.StoreData#storeData()
00024:    */
00025:   @Override
00026:   public void storeData() {
00027:     System.out.println("Store Data 2");
00028:     ((DataStore2) this.dataStore).setRPrice(((DataStore2) this.dataStore).getTemp_a());
00029:     ((DataStore2) this.dataStore).setPPrice(((DataStore2) this.dataStore).getTemp_b());
00030:     ((DataStore2) this.dataStore).setSPrice(((DataStore2) this.dataStore).getTemp_c());
00031:   }
00032:
00033: }
```

```
00001: package iit.softwaresystarch.project.model.datastores;
00002:
00003: /**
00004:  * The Class DataStore.
00005:  * This Class is responsible for stocking the main common attributes of Gas Pumps.
00006:  */
00007: public class DataStore {
00008:
00009:
00010:   /** The price. */
00011:   protected float price;
00012:
00013:   /** The total. */
00014:   protected float total;
00015:
00016:   /**
00017:    * Instantiates a new data store.
00018:    */
00019:   public DataStore() {}
00020:
00021:   /**
00022:    * Gets the price.
00023:    *
00024:    * @return the price
00025:    */
00026:   public float getPrice() {
00027:     return this.price;
00028:   }
00029:
00030:   /**
00031:    * Sets the price.
00032:    *
00033:    * @param price the new price
```

```
00034:    */
00035:   public void setPrice(float price) {
00036:     this.price = price;
00037:   }
00038:
00039:   /**
00040:    * Gets the total.
00041:    *
00042:    * @return the total
00043:    */
00044:   public float getTotal() {
00045:     return total;
00046:   }
00047:
00048:   /**
00049:    * Sets the total.
00050:    *
00051:    * @param total the new total
00052:    */
00053:   public void setTotal(float total) {
00054:     this.total = total;
00055:   }
00056:
00057:   /* (non-Javadoc)
00058:    * @see java.lang.Object#toString()
00059:    */
00060:   @Override
00061:   public String toString() {
00062:     return "DataStore [price=" + price + ", total=" + total + "]";
00063:   }
00064:
00065:
00066: }
```

```
00001: package iit.softwaresystarch.project.model.datastores;
00002:
00003: /**
00004:  * The Class DataStore1.
00005:  * This Class is responsible for stocking the specifics attributes of GasPump1.
00006:  */
00007: public class DataStore1 extends DataStore {
00008:
00009:   /**
00010:    * Instantiates a new DataStore1.
00011:    */
00012:   public DataStore1() {}
00013:
00014:   /** The temp a. */
00015:   private float temp_a;
00016:
00017:   /** The temp b. */
00018:   private float temp_b;
00019:
00020:   /** The regular price. */
00021:   private float rPrice;
00022:
00023:   /** The suoer price. */
00024:   private float sPrice;
00025:
00026:   /** The gallon pumped amount. */
00027:   private float gallon;
00028:
00029:   /**
00030:    * Gets the temp a.
00031:    *
00032:    * @return the temp a
00033:    */
```

```
00034:   public float getTemp_a() {
00035:     return temp_a;
00036:   }
00037:
00038:   /**
00039:    * Sets the temp a.
00040:    *
00041:    * @param temp_a the new temp a
00042:    */
00043:   public void setTemp_a(float temp_a) {
00044:     this.temp_a = temp_a;
00045:   }
00046:
00047:   /**
00048:    * Gets the temp b.
00049:    *
00050:    * @return the temp b
00051:    */
00052:   public float getTemp_b() {
00053:     return temp_b;
00054:   }
00055:
00056:   /**
00057:    * Sets the temp b.
00058:    *
00059:    * @param temp_b the new temp b
00060:    */
00061:   public void setTemp_b(float temp_b) {
00062:     this.temp_b = temp_b;
00063:   }
00064:
00065:   /**
00066:    * Gets the regular price.
```

```
00067:    *
00068:    * @return the regular price
00069:    */
00070:   public float getRPrice() {
00071:     return rPrice;
00072:   }
00073:
00074:   /**
00075:    * Sets the reguler price.
00076:    *
00077:    * @param rPrice the new regular price
00078:    */
00079:   public void setRPrice(float rPrice) {
00080:     this.rPrice = rPrice;
00081:   }
00082:
00083:   /**
00084:    * Gets the super price.
00085:    *
00086:    * @return the super price
00087:    */
00088:   public float getSPrice() {
00089:     return sPrice;
00090:   }
00091:
00092:   /**
00093:    * Sets the super price.
00094:    *
00095:    * @param sPrice the new super price
00096:    */
00097:   public void setSPrice(float sPrice) {
00098:     this.sPrice = sPrice;
00099:   }
```

```
00100:
00101:   /**
00102:    * Gets the gallon pumped amount.
00103:    *
00104:    * @return the gallon pumped amount
00105:    */
00106:   public float getGallon() {
00107:     return gallon;
00108:   }
00109:
00110:   /**
00111:    * Sets the gallon pumped amount.
00112:    *
00113:    * @param gallon the new gallon pumped amount
00114:    */
00115:   public void setGallon(float gallon) {
00116:     this.gallon = gallon;
00117:   }
00118:
00119:   /* (non-Javadoc)
00120:    * @see iit.softwaresystarch.project.model.datastores.DataStore#toString()
00121:    */
00122:   @Override
00123:   public String toString() {
00124:     return "DataStore1 [temp_a=" + temp_a + ", temp_b=" + temp_b + ", rPrice=" + rPrice
00125:         + ", sPrice=" + sPrice + ", gallon=" + gallon + ", price=" + price + ", total=" + total + "]";
00126:   }
00127:
00128: }
```

```
00001: package iit.softwaresystarch.project.model.datastores;
00002:
00003: /**
00004:  * The Class DataStore2.
00005:  * This Class is responsible for stocking the specifics attributes of GasPump2.
00006:  */
00007: public class DataStore2 extends DataStore {
00008:
00009:   /**
00010:    * Instantiates a new data store 2.
00011:    */
00012:   public DataStore2() {}
00013:
00014:   /** The temp a. */
00015:   private int temp_a;
00016:
00017:   /** The temp b. */
00018:   private int temp_b;
00019:
00020:   /** The temp c. */
00021:   private int temp_c;
00022:
00023:   /** The temp cash. */
00024:   private float temp_cash;
00025:
00026:   /** The regular price. */
00027:   private int rPrice;
00028:
00029:   /** The super price. */
00030:   private int sPrice;
00031:
00032:   /** The premium price. */
00033:   private int pPrice;
```

```
00034:
00035:   /** The liter pumped amount. */
00036:   private float liter;
00037:
00038:   /** The cash available. */
00039:   private float cash;
00040:
00041:   /**
00042:    * Gets the temp a.
00043:    *
00044:    * @return the temp a
00045:    */
00046:   public int getTemp_a() {
00047:     return temp_a;
00048:   }
00049:
00050:   /**
00051:    * Sets the temp a.
00052:    *
00053:    * @param temp_a the new temp a
00054:    */
00055:   public void setTemp_a(int temp_a) {
00056:     this.temp_a = temp_a;
00057:   }
00058:
00059:   /**
00060:    * Gets the temp b.
00061:    *
00062:    * @return the temp b
00063:    */
00064:   public int getTemp_b() {
00065:     return temp_b;
00066:   }
```

```
00067:
00068:   /**
00069:    * Sets the temp b.
00070:    *
00071:    * @param temp_b the new temp b
00072:    */
00073:   public void setTemp_b(int temp_b) {
00074:     this.temp_b = temp_b;
00075:   }
00076:
00077:   /**
00078:    * Gets the temp c.
00079:    *
00080:    * @return the temp c
00081:    */
00082:   public int getTemp_c() {
00083:     return temp_c;
00084:   }
00085:
00086:   /**
00087:    * Sets the temp c.
00088:    *
00089:    * @param temp_c the new temp c
00090:    */
00091:   public void setTemp_c(int temp_c) {
00092:     this.temp_c = temp_c;
00093:   }
00094:
00095:   /**
00096:    * Gets the temp cash.
00097:    *
00098:    * @return the temp cash
00099:    */
```

```
00100:   public float getTemp_cash() {
00101:     return temp_cash;
00102:   }
00103:
00104:   /**
00105:    * Sets the temp cash.
00106:    *
00107:    * @param temp_cash the new temp cash
00108:    */
00109:   public void setTemp_cash(float temp_cash) {
00110:     this.temp_cash = temp_cash;
00111:   }
00112:
00113:   /**
00114:    * Gets the regular price.
00115:    *
00116:    * @return the regular price
00117:    */
00118:   public int getRPrice() {
00119:     return rPrice;
00120:   }
00121:
00122:   /**
00123:    * Sets the regular price.
00124:    *
00125:    * @param rPrice the new regular price
00126:    */
00127:   public void setRPrice(int rPrice) {
00128:     this.rPrice = rPrice;
00129:   }
00130:
00131:   /**
00132:    * Gets the super price.
```

```
00133:    *
00134:    * @return the super price
00135:    */
00136:   public int getSPrice() {
00137:     return sPrice;
00138:   }
00139:
00140:   /**
00141:    * Sets the super price.
00142:    *
00143:    * @param sPrice the new super price
00144:    */
00145:   public void setSPrice(int sPrice) {
00146:     this.sPrice = sPrice;
00147:   }
00148:
00149:   /**
00150:    * Gets the premium price.
00151:    *
00152:    * @return the premium price
00153:    */
00154:   public int getPPrice() {
00155:     return pPrice;
00156:   }
00157:
00158:   /**
00159:    * Sets the premium price.
00160:    *
00161:    * @param pPrice the new premium price
00162:    */
00163:   public void setPPrice(int pPrice) {
00164:     this.pPrice = pPrice;
00165:   }
```

```
00166:
00167:   /**
00168:    * Gets the liter pumped amount.
00169:    *
00170:    * @return the liter pumped amount
00171:    */
00172:   public float getLiter() {
00173:     return liter;
00174:   }
00175:
00176:   /**
00177:    * Sets the liter pumped amount.
00178:    *
00179:    * @param liter the new liter pumped amount
00180:    */
00181:   public void setLiter(float liter) {
00182:     this.liter = liter;
00183:   }
00184:
00185:   /**
00186:    * Gets the cash available.
00187:    *
00188:    * @return the cash available
00189:    */
00190:   public float getCash() {
00191:     return cash;
00192:   }
00193:
00194:   /**
00195:    * Sets the cash available.
00196:    *
00197:    * @param cash the new cash available
00198:    */
```

```
00199:   public void setCash(float cash) {
00200:     this.cash = cash;
00201:   }
00202:
00203:   /* (non-Javadoc)
00204:    * @see iit.softwaresystarch.project.model.datastores.DataStore#toString()
00205:    */
00206:   @Override
00207:   public String toString() {
00208:     return "DataStore2 [temp_a=" + temp_a + ", temp_b=" + temp_b + ", temp_c=" + temp_c
00209:         + ", temp_cash=" + temp_cash + ", rPrice=" + rPrice + ", sPrice=" + sPrice + ", pPrice="
00210:         + pPrice + ", liter=" + liter + ", cash=" + cash + ", price=" + price + ", total=" + total + "]";
00211:   }
00212:
00213: }
```

```
00001: /**
00002:  * is a package containing the classes mostly related to both AbstractFactory and Strategy design patterns.
00003:  */
00004: package iit.softwaresystarch.project.model.datastores;
```

```
00001: package iit.softwaresystarch.project.model.efsms;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004: import iit.softwaresystarch.project.model.datastores.DataStore1;
00005: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory1;
00006:
00007: /**
00008:  * The Class GasPump1.
00009:  * This Class is the concrete representation of a GasPump1.
00010:  */
00011: public class GasPump1 {
00012:
00013:   /** The data store. */
00014:   private final DataStore dataStore;
00015:
00016:   /** The mda efsm. */
00017:   private final MDAEFSM mdaEfsm;
00018:
00019:   /**
00020:    * Instantiates a new GasPump1.
00021:    */
00022:   public GasPump1() {
00023:     final OperationAbstractFactory1 operationAbstractFactory1 = new OperationAbstractFactory1();
00024:     this.mdaEfsm = new MDAEFSM(operationAbstractFactory1);
00025:     this.dataStore = operationAbstractFactory1.getDataStore();
00026:   }
00027:
00028:   /**
00029:    * Activate.
00030:    *
00031:    * @param a the a
00032:    * @param b the b
00033:    */
```

```
00034:   public void activate(float a, float b) {
00035:     System.out.println("activate");
00036:     if (a > 0 && b > 0) {
00037:       ((DataStore1) this.dataStore).setTemp_a(a);
00038:       ((DataStore1) this.dataStore).setTemp_b(b);
00039:       this.mdaEfsm.activate();
00040:     }
00041:   }
00042:
00043:   /**
00044:    * Start.
00045:    */
00046:   public void start() {
00047:     System.out.println("start");
00048:     this.mdaEfsm.start();
00049:   }
00050:
00051:   /**
00052:    * Pay credit.
00053:    */
00054:   public void payCredit() {
00055:     System.out.println("payCredit");
00056:     this.mdaEfsm.payType(1);
00057:   }
00058:
00059:   /**
00060:    * Approved.
00061:    */
00062:   public void approved() {
00063:     System.out.println("approved");
00064:     this.mdaEfsm.approved();
00065:   }
00066:
```

```
00067:   /**
00068:    * Reject.
00069:    */
00070:   public void reject() {
00071:     System.out.println("reject");
00072:     this.mdaEfsm.reject();
00073:   }
00074:
00075:   /**
00076:    * Cancel.
00077:    */
00078:   public void cancel() {
00079:     System.out.println("cancel");
00080:     this.mdaEfsm.cancel();
00081:   }
00082:
00083:   /**
00084:    * Regular.
00085:    */
00086:   public void regular() {
00087:     System.out.println("regular");
00088:     this.mdaEfsm.selectGas(1);
00089:   }
00090:
00091:   /**
00092:    * Super gas.
00093:    */
00094:   public void superGas() {
00095:     System.out.println("superGas");
00096:     this.mdaEfsm.selectGas(2);
00097:   }
00098:
00099:   /**
```

```
00100:    * Start pump.
00101:    */
00102:   public void startPump() {
00103:     System.out.println("startPump");
00104:     this.mdaEfsm.startPump();
00105:   }
00106:
00107:   /**
00108:    * Pump gallon.
00109:    */
00110:   public void pumpGallon() {
00111:     System.out.println("pumpGallon");
00112:     this.mdaEfsm.pump();
00113:   }
00114:
00115:   /**
00116:    * Stop pump.
00117:    */
00118:   public void stopPump() {
00119:     System.out.println("stopPump");
00120:     this.mdaEfsm.stopPump();
00121:     this.mdaEfsm.receipt();
00122:   }
00123:
00124:   /*
00125:    * (non-Javadoc)
00126:    *
00127:    * @see java.lang.Object#toString()
00128:    */
00129:   @Override
00130:   public String toString() {
00131:     return "GasPump1 [dataStore=" + this.dataStore + ", mdaEfsm=" + this.mdaEfsm + "]";
00132:   }
```

00133:

00134: }

```
00001: package iit.softwaresystarch.project.model.efsms;
00002:
00003: import iit.softwaresystarch.project.model.datastores.DataStore;
00004: import iit.softwaresystarch.project.model.datastores.DataStore2;
00005: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory2;
00006:
00007: /**
00008:  * The Class GasPump2.
00009:  * This Class is the concrete representation of a GasPump2.
00010:  */
00011: public class GasPump2 {
00012:
00013:   /** The data store. */
00014:   private final DataStore dataStore;
00015:
00016:   /** The mda efsm. */
00017:   private final MDAEFSM mdaEfsm;
00018:
00019:   /**
00020:    * Instantiates a new GasPump2.
00021:    */
00022:   public GasPump2() {
00023:     final OperationAbstractFactory2 operationAbstractFactory2 = new OperationAbstractFactory2();
00024:     this.mdaEfsm = new MDAEFSM(operationAbstractFactory2);
00025:     this.dataStore = operationAbstractFactory2.getDataStore();
00026:   }
00027:
00028:   /**
00029:    * Activate.
00030:    *
00031:    * @param a the a
00032:    * @param b the b
00033:    * @param c the c
```

```
00034:    */
00035:   public void activate(int a, int b, int c) {
00036:     System.out.println("activate");
00037:     if (a > 0 && b > 0 && c > 0) {
00038:       ((DataStore2) this.dataStore).setTemp_a(a);
00039:       ((DataStore2) this.dataStore).setTemp_b(b);
00040:       ((DataStore2) this.dataStore).setTemp_c(c);
00041:       this.mdaEfsm.activate();
00042:     }
00043:   }
00044:
00045:   /**
00046:    * Start.
00047:    */
00048:   public void start() {
00049:     System.out.println("start");
00050:     this.mdaEfsm.start();
00051:   }
00052:
00053:   /**
00054:    * Pay cash.
00055:    *
00056:    * @param c the c
00057:    */
00058:   public void payCash(float c) {
00059:     System.out.println("payCash");
00060:     System.out.println("cash is " + c + ".");
00061:     if (c > 0) {
00062:       ((DataStore2) this.dataStore).setTemp_cash(c);
00063:       this.mdaEfsm.payType(2);
00064:     }
00065:   }
00066:
```

```
00067:   /**
00068:    * Cancel.
00069:    */
00070:   public void cancel() {
00071:     System.out.println("cancel");
00072:     this.mdaEfsm.cancel();
00073:   }
00074:
00075:   /**
00076:    * Super gas.
00077:    */
00078:   public void superGas() {
00079:     System.out.println("superGas");
00080:     this.mdaEfsm.selectGas(2);
00081:   }
00082:
00083:   /**
00084:    * Regular.
00085:    */
00086:   public void regular() {
00087:     System.out.println("regular");
00088:     this.mdaEfsm.selectGas(1);
00089:   }
00090:
00091:   /**
00092:    * Premium.
00093:    */
00094:   public void premium() {
00095:     System.out.println("premium");
00096:     this.mdaEfsm.selectGas(3);
00097:   }
00098:
00099:   /**
```

```
00100:    * Start pump.
00101:    */
00102:   public void startPump() {
00103:     System.out.println("startPump");
00104:     this.mdaEfsm.startPump();
00105:   }
00106:
00107:   /**
00108:    * Pump liter.
00109:    */
00110:   public void pumpLiter() {
00111:     System.out.println("pumpLiter");
00112:     if (((DataStore2) this.dataStore).getCash() >= (((DataStore2) this.dataStore).getLiter() + 1)
00113:         * (((DataStore2) this.dataStore).getPrice())) {
00114:       this.mdaEfsm.pump();
00115:     } else {
00116:       this.mdaEfsm.setCurrentState(this.mdaEfsm.getStatesList().get(7));
00117:     }
00118:   }
00119:
00120:   /**
00121:    * Stop.
00122:    */
00123:   public void stop() {
00124:     System.out.println("stop");
00125:     this.mdaEfsm.stopPump();
00126:   }
00127:
00128:   /**
00129:    * Receipt.
00130:    */
00131:   public void receipt() {
00132:     System.out.println("receipt");
```

```
00133:     this.mdaEfsm.receipt();
00134:   }
00135:
00136:   /**
00137:    * No receipt.
00138:    */
00139:   public void noReceipt() {
00140:     System.out.println("noReceipt");
00141:     this.mdaEfsm.noReceipt();
00142:   }
00143:
00144:   /*
00145:    * (non-Javadoc)
00146:    *
00147:    * @see java.lang.Object#toString()
00148:    */
00149:   @Override
00150:   public String toString() {
00151:     return "GasPump2 [dataStore=" + this.dataStore + ", mdaEfsm=" + this.mdaEfsm + "]";
00152:   }
00153:
00154: }
```

```
00001: package iit.softwaresystarch.project.model.efsms;
00002:
00003: import java.util.ArrayList;
00004: import java.util.List;
00005:
00006: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00007: import iit.softwaresystarch.project.model.states.Start;
00008: import iit.softwaresystarch.project.model.states.State;
00009: import iit.softwaresystarch.project.model.states.State0;
00010: import iit.softwaresystarch.project.model.states.State1;
00011: import iit.softwaresystarch.project.model.states.State2;
00012: import iit.softwaresystarch.project.model.states.State3;
00013: import iit.softwaresystarch.project.model.states.State4;
00014: import iit.softwaresystarch.project.model.states.State5;
00015: import iit.softwaresystarch.project.model.states.State6;
00016:
00017: /**
00018:  * The Class MDAEFSM.
00019:  */
00020: public class MDAEFSM {
00021:
00022:   /** The states list. */
00023:   private final List<State> statesList;
00024:
00025:   /** The current state. */
00026:   private State currentState;
00027:
00028:   /** The operation abstract factory. */
00029:   private final OperationAbstractFactory operationAbstractFactory;
00030:
00031:   /**
00032:    * Instantiates a new MDAEFSM and instantiates .
00033:    *
```

```
00034:    * @param operationAbstractFactory the operation abstract factory
00035:    */
00036:   public MDAEFSM(OperationAbstractFactory operationAbstractFactory) {
00037:     this.operationAbstractFactory = operationAbstractFactory;
00038:     this.statesList = new ArrayList<>();
00039:     this.statesList.add(new Start(operationAbstractFactory));
00040:     this.statesList.add(new State0(operationAbstractFactory));
00041:     this.statesList.add(new State1(operationAbstractFactory));
00042:     this.statesList.add(new State2(operationAbstractFactory));
00043:     this.statesList.add(new State3(operationAbstractFactory));
00044:     this.statesList.add(new State4(operationAbstractFactory));
00045:     this.statesList.add(new State5(operationAbstractFactory));
00046:     this.statesList.add(new State6(operationAbstractFactory));
00047:     this.currentState = this.statesList.get(0);
00048:   }
00049:
00050:   /**
00051:    * Activate.
00052:    */
00053:   public void activate() {
00054:     if (this.currentState.getID() == -1) {
00055:       this.currentState.activate();
00056:       this.currentState = this.statesList.get(1);
00057:     }
00058:   }
00059:
00060:   /**
00061:    * Start.
00062:    */
00063:   public void start() {
00064:     if (this.currentState.getID() == 0) {
00065:       this.currentState.start();
00066:       this.currentState = this.statesList.get(2);
```

```
00067:     }
00068:   }
00069:
00070:   /**
00071:    * Pay type.
00072:    *
00073:    * @param t the t
00074:    */
00075:   public void payType(int t) {
00076:     if (this.currentState.getID() == 1) {
00077:       if (t == 1) {
00078:         this.currentState.payType(t);
00079:         this.currentState = this.statesList.get(3);
00080:       } else if (t == 2) {
00081:         this.currentState.payType(t);
00082:         this.currentState = this.statesList.get(4);
00083:       }
00084:     }
00085:   }
00086:
00087:   /**
00088:    * Approved.
00089:    */
00090:   public void approved() {
00091:     if (this.currentState.getID() == 2) {
00092:       this.currentState.approved();
00093:       this.currentState = this.statesList.get(4);
00094:     }
00095:   }
00096:
00097:   /**
00098:    * Reject.
00099:    */
```

```
00100:   public void reject() {
00101:     if (this.currentState.getID() == 2) {
00102:       this.currentState.reject();
00103:       this.currentState = this.statesList.get(1);
00104:     }
00105:   }
00106:
00107:   /**
00108:    * Cancel.
00109:    */
00110:   public void cancel() {
00111:     if (this.currentState.getID() == 3) {
00112:       this.currentState.cancel();
00113:       this.currentState = this.statesList.get(1);
00114:     }
00115:   }
00116:
00117:   /**
00118:    * Select gas.
00119:    *
00120:    * @param g the g
00121:    */
00122:   public void selectGas(int g) {
00123:     if (this.currentState.getID() == 3) {
00124:       this.currentState.selectGas(g);
00125:       this.currentState = this.statesList.get(5);
00126:     }
00127:   }
00128:
00129:   /**
00130:    * Start pump.
00131:    */
00132:   public void startPump() {
```

```
00133:     if (this.currentState.getID() == 4) {
00134:       this.currentState.startPump();
00135:       this.currentState = this.statesList.get(6);
00136:     }
00137:   }
00138:
00139:   /**
00140:    * Pump.
00141:    */
00142:   public void pump() {
00143:     if (this.currentState.getID() == 5) {
00144:       this.currentState.pump();
00145:     }
00146:   }
00147:
00148:   /**
00149:    * Stop pump.
00150:    */
00151:   public void stopPump() {
00152:     if (this.currentState.getID() == 5) {
00153:       this.currentState.stopPump();
00154:       this.currentState = this.statesList.get(7);
00155:     }
00156:   }
00157:
00158:   /**
00159:    * Receipt.
00160:    */
00161:   public void receipt() {
00162:     if (this.currentState.getID() == 6) {
00163:       this.currentState.receipt();
00164:       this.currentState = this.statesList.get(1);
00165:     }
```

```
00166:   }
00167:
00168:   /**
00169:    * No receipt.
00170:    */
00171:   public void noReceipt() {
00172:     if (this.currentState.getID() == 6) {
00173:       this.currentState.noReceipt();
00174:       this.currentState = this.statesList.get(1);
00175:     }
00176:   }
00177:
00178:   /**
00179:    * Gets the states list.
00180:    *
00181:    * @return the states list
00182:    */
00183:   public List<State> getStatesList() {
00184:     return this.statesList;
00185:   }
00186:
00187:   /**
00188:    * Gets the current state.
00189:    *
00190:    * @return the current state
00191:    */
00192:   public State getCurrentState() {
00193:     return this.currentState;
00194:   }
00195:
00196:   /**
00197:    * Sets the current state.
00198:    *
```

```
00199:    * @param currentState the new current state
00200:    */
00201:   public void setCurrentState(State currentState) {
00202:     this.currentState = currentState;
00203:   }
00204:
00205:   /*
00206:    * (non-Javadoc)
00207:    *
00208:    * @see java.lang.Object#toString()
00209:    */
00210:   @Override
00211:   public String toString() {
00212:     return "MDAEFSM [statesList=" + this.statesList + ", currentState=" + this.currentState
00213:         + ", operationAbstractFactory=" + this.operationAbstractFactory + "]";
00214:   }
00215:
00216: }
```

```
00001: /**
00002:  * is a package containing the classes mostly related to Model Driven Architecture.
00003:  */
00004: package iit.softwaresystarch.project.model.efsms;
```

```
00001:
00002: /**
00003:  * iit.softwaresystarch.project.model.factories is a package
00004:  * containing the classes mostly related to AbstractFactory design pattern.
00005:  */
00006: package iit.softwaresystarch.project.model.factories;
00007:
00008: import iit.softwaresystarch.project.model.actions.CancelMsg;
00009: import iit.softwaresystarch.project.model.actions.DisplayMenu;
00010: import iit.softwaresystarch.project.model.actions.GasPumpedMsg;
00011: import iit.softwaresystarch.project.model.actions.PayMsg;
00012: import iit.softwaresystarch.project.model.actions.PrintReceipt;
00013: import iit.softwaresystarch.project.model.actions.PumpGasUnit;
00014: import iit.softwaresystarch.project.model.actions.ReadyMsg;
00015: import iit.softwaresystarch.project.model.actions.RejectMsg;
00016: import iit.softwaresystarch.project.model.actions.ReturnCash;
00017: import iit.softwaresystarch.project.model.actions.SetInitialValues;
00018: import iit.softwaresystarch.project.model.actions.SetPrice;
00019: import iit.softwaresystarch.project.model.actions.StopMsg;
00020: import iit.softwaresystarch.project.model.actions.StoreCash;
00021: import iit.softwaresystarch.project.model.actions.StoreData;
00022: import iit.softwaresystarch.project.model.datastores.DataStore;
00023:
00024: /**
00025:  * A factory for creating Operation objects.
00026:  * This Class is designed as an Interface for the concrete factories that will builds
00027:  * all the differents instances used by the operation class to make actions on the system.
00028:  */
00029: public class OperationAbstractFactory {
00030:
00031:   /** The data store. */
00032:   protected DataStore dataStore;
00033:
```

```
00034:   /** The store data. */
00035:   protected StoreData storeData;
00036:
00037:   /** The display menu. */
00038:   protected DisplayMenu displayMenu;
00039:
00040:   /** The pay msg. */
00041:   protected PayMsg payMsg;
00042:
00043:   /** The store cash. */
00044:   protected StoreCash storeCash;
00045:
00046:   /** The cancel msg. */
00047:   protected CancelMsg cancelMsg;
00048:
00049:   /** The reject msg. */
00050:   protected RejectMsg rejectMsg;
00051:
00052:   /** The set price. */
00053:   protected SetPrice setPrice;
00054:
00055:   /** The ready msg. */
00056:   protected ReadyMsg readyMsg;
00057:
00058:   /** The set initial values. */
00059:   protected SetInitialValues setInitialValues;
00060:
00061:   /** The pump gas unit. */
00062:   protected PumpGasUnit pumpGasUnit;
00063:
00064:   /** The gas pumped msg. */
00065:   protected GasPumpedMsg gasPumpedMsg;
00066:
```

```
00067:   /** The stop msg. */
00068:   protected StopMsg stopMsg;
00069:
00070:   /** The print receipt. */
00071:   protected PrintReceipt printReceipt;
00072:
00073:   /** The return cash. */
00074:   protected ReturnCash returnCash;
00075:
00076:   /**
00077:    * Instantiates a new OperationAbstractFactory.
00078:    */
00079:   public OperationAbstractFactory() {}
00080:
00081:   /**
00082:    * Gets the data store.
00083:    *
00084:    * @return the data store
00085:    */
00086:   public DataStore getDataStore() {
00087:     return this.dataStore;
00088:   }
00089:
00090:   /**
00091:    * Gets the store data.
00092:    *
00093:    * @return the store data
00094:    */
00095:   public StoreData getStoreData() {
00096:     return this.storeData;
00097:   }
00098:
00099:   /**
```

```
00100:    * Gets the pay msg.
00101:    *
00102:    * @return the pay msg
00103:    */
00104:   public PayMsg getPayMsg() {
00105:     return this.payMsg;
00106:   }
00107:
00108:   /**
00109:    * Gets the store cash.
00110:    *
00111:    * @return the store cash
00112:    */
00113:   public StoreCash getStoreCash() {
00114:     return this.storeCash;
00115:   }
00116:
00117:   /**
00118:    * Gets the cancel msg.
00119:    *
00120:    * @return the cancel msg
00121:    */
00122:   public CancelMsg getCancelMsg() {
00123:     return this.cancelMsg;
00124:   }
00125:
00126:   /**
00127:    * Gets the display menu.
00128:    *
00129:    * @return the display menu
00130:    */
00131:   public DisplayMenu getDisplayMenu() {
00132:     return this.displayMenu;
```

```
00133:   }
00134:
00135:   /**
00136:    * Gets the reject msg.
00137:    *
00138:    * @return the reject msg
00139:    */
00140:   public RejectMsg getRejectMsg() {
00141:     return this.rejectMsg;
00142:   }
00143:
00144:   /**
00145:    * Gets the sets the price.
00146:    *
00147:    * @param g the g
00148:    * @return the sets the price
00149:    */
00150:   public SetPrice getSetPrice(int g) {
00151:     return this.setPrice;
00152:   }
00153:
00154:   /**
00155:    * Gets the ready msg.
00156:    *
00157:    * @return the ready msg
00158:    */
00159:   public ReadyMsg getReadyMsg() {
00160:     return this.readyMsg;
00161:   }
00162:
00163:   /**
00164:    * Gets the sets the initial values.
00165:    *
```

```
00166:    * @return the sets the initial values
00167:    */
00168:   public SetInitialValues getSetInitialValues() {
00169:     return this.setInitialValues;
00170:   }
00171:
00172:   /**
00173:    * Gets the pump gas unit.
00174:    *
00175:    * @return the pump gas unit
00176:    */
00177:   public PumpGasUnit getPumpGasUnit() {
00178:     return this.pumpGasUnit;
00179:   }
00180:
00181:   /**
00182:    * Gets the gas pumped msg.
00183:    *
00184:    * @return the gas pumped msg
00185:    */
00186:   public GasPumpedMsg getGasPumpedMsg() {
00187:     return this.gasPumpedMsg;
00188:   }
00189:
00190:   /**
00191:    * Gets the stop msg.
00192:    *
00193:    * @return the stop msg
00194:    */
00195:   public StopMsg getStopMsg() {
00196:     return this.stopMsg;
00197:   }
00198:
```

```
00199:   /**
00200:    * Gets the prints the receipt.
00201:    *
00202:    * @return the prints the receipt
00203:    */
00204:   public PrintReceipt getPrintReceipt() {
00205:     return this.printReceipt;
00206:   }
00207:
00208:   /**
00209:    * Gets the return cash.
00210:    *
00211:    * @return the return cash
00212:    */
00213:   public ReturnCash getReturnCash() {
00214:     return this.returnCash;
00215:   }
00216:
00217:   /*
00218:    * (non-Javadoc)
00219:    *
00220:    * @see java.lang.Object#toString()
00221:    */
00222:   @Override
00223:   public String toString() {
00224:     return "OperationAbstractFactory [dataStore=" + this.dataStore + ", storeData=" + this.storeData
00225:         + ", displayMenu=" + this.displayMenu + ", payMsg=" + this.payMsg + ", storeCash="
00226:         + this.storeCash + ", cancelMsg=" + this.cancelMsg + ", rejectMsg=" + this.rejectMsg
00227:         + ", setPrice=" + this.setPrice + ", readyMsg=" + this.readyMsg + ", setInitialValues="
00228:         + this.setInitialValues + ", pumpGasUnit=" + this.pumpGasUnit + ", gasPumpedMsg="
00229:         + this.gasPumpedMsg + ", stopMsg=" + this.stopMsg + ", printReceipt=" + this.printReceipt
00230:         + ", returnCash=" + this.returnCash + "]";
00231:   }
```

00232:

00233: }

```
00001: package iit.softwaresystarch.project.model.factories;
00002:
00003: import iit.softwaresystarch.project.model.actions.CancelMsg;
00004: import iit.softwaresystarch.project.model.actions.CancelMsg1;
00005: import iit.softwaresystarch.project.model.actions.DisplayMenu;
00006: import iit.softwaresystarch.project.model.actions.DisplayMenu1;
00007: import iit.softwaresystarch.project.model.actions.GasPumpedMsg;
00008: import iit.softwaresystarch.project.model.actions.GasPumpedMsg1;
00009: import iit.softwaresystarch.project.model.actions.PayMsg;
00010: import iit.softwaresystarch.project.model.actions.PayMsg1;
00011: import iit.softwaresystarch.project.model.actions.PrintReceipt;
00012: import iit.softwaresystarch.project.model.actions.PrintReceipt1;
00013: import iit.softwaresystarch.project.model.actions.PumpGasUnit;
00014: import iit.softwaresystarch.project.model.actions.PumpGasUnit1;
00015: import iit.softwaresystarch.project.model.actions.ReadyMsg;
00016: import iit.softwaresystarch.project.model.actions.RejectMsg;
00017: import iit.softwaresystarch.project.model.actions.RejectMsg1;
00018: import iit.softwaresystarch.project.model.actions.ReturnCash;
00019: import iit.softwaresystarch.project.model.actions.SetInitialValues;
00020: import iit.softwaresystarch.project.model.actions.SetInitialValues1;
00021: import iit.softwaresystarch.project.model.actions.SetPrice;
00022: import iit.softwaresystarch.project.model.actions.SetPrice1;
00023: import iit.softwaresystarch.project.model.actions.StopMsg;
00024: import iit.softwaresystarch.project.model.actions.StoreCash;
00025: import iit.softwaresystarch.project.model.actions.StoreData;
00026: import iit.softwaresystarch.project.model.actions.StoreData1;
00027: import iit.softwaresystarch.project.model.datastores.DataStore1;
00028:
00029: /**
00030:  * The Class OperationAbstractFactory1.
00031:  */
00032: public class OperationAbstractFactory1 extends OperationAbstractFactory {
00033:
```

```
00034:   /**
00035:    * Instantiates a new OperationAbstractFactory1 and Instantiates the requiered attributes.
00036:    */
00037:   public OperationAbstractFactory1() {
00038:     this.dataStore = new DataStore1();
00039:     this.payMsg = new PayMsg1();
00040:     this.displayMenu = new DisplayMenu1();
00041:     this.rejectMsg = new RejectMsg1();
00042:     this.cancelMsg = new CancelMsg1();
00043:     this.storeData = new StoreData1((DataStore1) this.dataStore);
00044:     this.setPrice = new SetPrice1((DataStore1) this.dataStore);
00045:     this.setInitialValues = new SetInitialValues1((DataStore1) this.dataStore);
00046:     this.pumpGasUnit = new PumpGasUnit1((DataStore1) this.dataStore);
00047:     this.gasPumpedMsg = new GasPumpedMsg1((DataStore1) this.dataStore);
00048:     this.printReceipt = new PrintReceipt1((DataStore1) this.dataStore);
00049:   }
00050:
00051:   /*
00052:    * (non-Javadoc)
00053:    *
00054:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getStoreData()
00055:    */
00056:   @Override
00057:   public StoreData getStoreData() {
00058:     return this.storeData;
00059:   }
00060:
00061:   /*
00062:    * (non-Javadoc)
00063:    *
00064:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getPayMsg()
00065:    */
00066:   @Override
```

```
00067:   public PayMsg getPayMsg() {
00068:     return this.payMsg;
00069:   }
00070:
00071:   /*
00072:    * (non-Javadoc)
00073:    *
00074:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getStoreCash()
00075:    */
00076:   @Override
00077:   public StoreCash getStoreCash() {
00078:     return this.storeCash;
00079:   }
00080:
00081:   /*
00082:    * (non-Javadoc)
00083:    *
00084:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getDisplayMenu()
00085:    */
00086:   @Override
00087:   public DisplayMenu getDisplayMenu() {
00088:     return this.displayMenu;
00089:   }
00090:
00091:   /*
00092:    * (non-Javadoc)
00093:    *
00094:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getRejectMsg()
00095:    */
00096:   @Override
00097:   public RejectMsg getRejectMsg() {
00098:     return this.rejectMsg;
00099:   }
```

```
00100:
00101:   /*
00102:    * (non-Javadoc)
00103:    *
00104:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getCancelMsg()
00105:    */
00106:   @Override
00107:   public CancelMsg getCancelMsg() {
00108:     return this.cancelMsg;
00109:   }
00110:
00111:   /*
00112:    * (non-Javadoc)
00113:    *
00114:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getSetPrice(int)
00115:    */
00116:   @Override
00117:   public SetPrice getSetPrice(int g) {
00118:     return this.setPrice;
00119:   }
00120:
00121:   /*
00122:    * (non-Javadoc)
00123:    *
00124:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getReadyMsg()
00125:    */
00126:   @Override
00127:   public ReadyMsg getReadyMsg() {
00128:     return this.readyMsg;
00129:   }
00130:
00131:   /*
00132:    * (non-Javadoc)
```

```
00133:    *
00134:    * @see
00135:    * iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getSetInitialValues()
00136:    */
00137:   @Override
00138:   public SetInitialValues getSetInitialValues() {
00139:     return this.setInitialValues;
00140:   }
00141:
00142:   /*
00143:    * (non-Javadoc)
00144:    *
00145:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getPumpGasUnit()
00146:    */
00147:   @Override
00148:   public PumpGasUnit getPumpGasUnit() {
00149:     return this.pumpGasUnit;
00150:   }
00151:
00152:   /*
00153:    * (non-Javadoc)
00154:    *
00155:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getGasPumpedMsg()
00156:    */
00157:   @Override
00158:   public GasPumpedMsg getGasPumpedMsg() {
00159:     return this.gasPumpedMsg;
00160:   }
00161:
00162:   /*
00163:    * (non-Javadoc)
00164:    *
00165:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getStopMsg()
```

```
00166:    */
00167:   @Override
00168:   public StopMsg getStopMsg() {
00169:     return this.stopMsg;
00170:   }
00171:
00172:   /*
00173:    * (non-Javadoc)
00174:    *
00175:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getPrintReceipt()
00176:    */
00177:   @Override
00178:   public PrintReceipt getPrintReceipt() {
00179:     return this.printReceipt;
00180:   }
00181:
00182:   /*
00183:    * (non-Javadoc)
00184:    *
00185:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getReturnCash()
00186:    */
00187:   @Override
00188:   public ReturnCash getReturnCash() {
00189:     return this.returnCash;
00190:   }
00191:
00192:   /*
00193:    * (non-Javadoc)
00194:    *
00195:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#toString()
00196:    */
00197:   @Override
00198:   public String toString() {
```

```
    return "OperationAbstractFactory1 [dataStore=" + this.dataStore + ", storeData="
        + this.storeData + ", displayMenu=" + this.displayMenu + ", payMsg=" + this.payMsg
        + ", storeCash=" + this.storeCash + ", cancelMsg=" + this.cancelMsg + ", rejectMsg="
        + this.rejectMsg + ", setPrice=" + this.setPrice + ", readyMsg=" + this.readyMsg
        + ", setInitialValues=" + this.setInitialValues + ", pumpGasUnit=" + this.pumpGasUnit
        + ", gasPumpedMsg=" + this.gasPumpedMsg + ", stopMsg=" + this.stopMsg + ", printReceipt="
        + this.printReceipt + ", returnCash=" + this.returnCash + "]";
  }

}
```

```
00001: package iit.softwaresystarch.project.model.factories;
00002:
00003: import iit.softwaresystarch.project.model.actions.CancelMsg;
00004: import iit.softwaresystarch.project.model.actions.CancelMsg2;
00005: import iit.softwaresystarch.project.model.actions.DisplayMenu;
00006: import iit.softwaresystarch.project.model.actions.DisplayMenu2;
00007: import iit.softwaresystarch.project.model.actions.GasPumpedMsg;
00008: import iit.softwaresystarch.project.model.actions.GasPumpedMsg2;
00009: import iit.softwaresystarch.project.model.actions.PayMsg;
00010: import iit.softwaresystarch.project.model.actions.PayMsg2;
00011: import iit.softwaresystarch.project.model.actions.PrintReceipt;
00012: import iit.softwaresystarch.project.model.actions.PrintReceipt2;
00013: import iit.softwaresystarch.project.model.actions.PumpGasUnit;
00014: import iit.softwaresystarch.project.model.actions.PumpGasUnit2;
00015: import iit.softwaresystarch.project.model.actions.ReadyMsg;
00016: import iit.softwaresystarch.project.model.actions.RejectMsg;
00017: import iit.softwaresystarch.project.model.actions.ReturnCash;
00018: import iit.softwaresystarch.project.model.actions.ReturnCash2;
00019: import iit.softwaresystarch.project.model.actions.SetInitialValues;
00020: import iit.softwaresystarch.project.model.actions.SetInitialValues2;
00021: import iit.softwaresystarch.project.model.actions.SetPrice;
00022: import iit.softwaresystarch.project.model.actions.SetPrice2;
00023: import iit.softwaresystarch.project.model.actions.StopMsg;
00024: import iit.softwaresystarch.project.model.actions.StoreCash;
00025: import iit.softwaresystarch.project.model.actions.StoreCash2;
00026: import iit.softwaresystarch.project.model.actions.StoreData;
00027: import iit.softwaresystarch.project.model.actions.StoreData2;
00028: import iit.softwaresystarch.project.model.datastores.DataStore2;
00029:
00030: /**
00031:  * The Class OperationAbstractFactory2.
00032:  */
00033: public class OperationAbstractFactory2 extends OperationAbstractFactory {
```

```
00034:
00035:  /**
00036:   * Instantiates a new OperationAbstractFactory2 and Instantiates the requiered attributes.
00037:   */
00038:  public OperationAbstractFactory2() {
00039:    this.dataStore = new DataStore2();
00040:    this.displayMenu = new DisplayMenu2();
00041:    this.payMsg = new PayMsg2();
00042:    this.cancelMsg = new CancelMsg2();
00043:    this.storeData = new StoreData2((DataStore2) this.dataStore);
00044:    this.storeCash = new StoreCash2((DataStore2) this.dataStore);
00045:    this.setPrice = new SetPrice2((DataStore2) this.dataStore);
00046:    this.setInitialValues = new SetInitialValues2((DataStore2) this.dataStore);
00047:    this.pumpGasUnit = new PumpGasUnit2((DataStore2) this.dataStore);
00048:    this.gasPumpedMsg = new GasPumpedMsg2((DataStore2) this.dataStore);
00049:    this.printReceipt = new PrintReceipt2((DataStore2) this.dataStore);
00050:    this.returnCash = new ReturnCash2((DataStore2) this.dataStore);
00051:  }
00052:
00053:  /*
00054:   * (non-Javadoc)
00055:   *
00056:   * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getStoreData()
00057:   */
00058:  @Override
00059:  public StoreData getStoreData() {
00060:    return this.storeData;
00061:  }
00062:
00063:  /*
00064:   * (non-Javadoc)
00065:   *
00066:   * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getPayMsg()
```

```
00067:    */
00068:   @Override
00069:   public PayMsg getPayMsg() {
00070:     return this.payMsg;
00071:   }
00072:
00073:   /*
00074:    * (non-Javadoc)
00075:    *
00076:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getCancelMsg()
00077:    */
00078:   @Override
00079:   public CancelMsg getCancelMsg() {
00080:     return this.cancelMsg;
00081:   }
00082:
00083:   /*
00084:    * (non-Javadoc)
00085:    *
00086:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getStoreCash()
00087:    */
00088:   @Override
00089:   public StoreCash getStoreCash() {
00090:     return this.storeCash;
00091:   }
00092:
00093:   /*
00094:    * (non-Javadoc)
00095:    *
00096:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getDisplayMenu()
00097:    */
00098:   @Override
00099:   public DisplayMenu getDisplayMenu() {
```

```
00100:     return this.displayMenu;
00101:   }
00102:
00103:   /*
00104:    * (non-Javadoc)
00105:    *
00106:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getRejectMsg()
00107:    */
00108:   @Override
00109:   public RejectMsg getRejectMsg() {
00110:     return this.rejectMsg;
00111:   }
00112:
00113:   /*
00114:    * (non-Javadoc)
00115:    *
00116:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getSetPrice(int)
00117:    */
00118:   @Override
00119:   public SetPrice getSetPrice(int g) {
00120:     return this.setPrice;
00121:   }
00122:
00123:   /*
00124:    * (non-Javadoc)
00125:    *
00126:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getReadyMsg()
00127:    */
00128:   @Override
00129:   public ReadyMsg getReadyMsg() {
00130:     return this.readyMsg;
00131:   }
00132:
```

```
00133:   /*
00134:    * (non-Javadoc)
00135:    *
00136:    * @see
00137:    * iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getSetInitialValues()
00138:    */
00139:   @Override
00140:   public SetInitialValues getSetInitialValues() {
00141:     return this.setInitialValues;
00142:   }
00143:
00144:   /*
00145:    * (non-Javadoc)
00146:    *
00147:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getPumpGasUnit()
00148:    */
00149:   @Override
00150:   public PumpGasUnit getPumpGasUnit() {
00151:     return this.pumpGasUnit;
00152:   }
00153:
00154:   /*
00155:    * (non-Javadoc)
00156:    *
00157:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getGasPumpedMsg()
00158:    */
00159:   @Override
00160:   public GasPumpedMsg getGasPumpedMsg() {
00161:     return this.gasPumpedMsg;
00162:   }
00163:
00164:   /*
00165:    * (non-Javadoc)
```

```
00166:    *
00167:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getStopMsg()
00168:    */
00169:   @Override
00170:   public StopMsg getStopMsg() {
00171:     return this.stopMsg;
00172:   }
00173:
00174:   /*
00175:    * (non-Javadoc)
00176:    *
00177:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getPrintReceipt()
00178:    */
00179:   @Override
00180:   public PrintReceipt getPrintReceipt() {
00181:     return this.printReceipt;
00182:   }
00183:
00184:   /*
00185:    * (non-Javadoc)
00186:    *
00187:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#getReturnCash()
00188:    */
00189:   @Override
00190:   public ReturnCash getReturnCash() {
00191:     return this.returnCash;
00192:   }
00193:
00194:   /*
00195:    * (non-Javadoc)
00196:    *
00197:    * @see iit.softwaresystarch.project.model.factories.OperationAbstractFactory#toString()
00198:    */
```

```
00199:   @Override
00200:   public String toString() {
00201:     return "OperationAbstractFactory2 [dataStore=" + this.dataStore + ", storeData="
00202:         + this.storeData + ", displayMenu=" + this.displayMenu + ", payMsg=" + this.payMsg
00203:         + ", storeCash=" + this.storeCash + ", cancelMsg=" + this.cancelMsg + ", rejectMsg="
00204:         + this.rejectMsg + ", setPrice=" + this.setPrice + ", readyMsg=" + this.readyMsg
00205:         + ", setInitialValues=" + this.setInitialValues + ", pumpGasUnit=" + this.pumpGasUnit
00206:         + ", gasPumpedMsg=" + this.gasPumpedMsg + ", stopMsg=" + this.stopMsg + ", printReceipt="
00207:         + this.printReceipt + ", returnCash=" + this.returnCash + "]";
00208:   }
00209:
00210: }
```

```
00001: /**
00002:  * is a package containing the classes mostly related to AbstractFactory design pattern.
00003:  */
00004: package iit.softwaresystarch.project.model.factories;
```

```
00001: /**
00002:  * is a package containing the packages and classes mostly related to Model design pattern.
00003:  */
00004: package iit.softwaresystarch.project.model;
```

```
00001: /**
00002:  * is a package containing the classes mostly related to State design pattern.
00003:  */
00004: package iit.softwaresystarch.project.model.states;
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class Start.
00007:  */
00008: public class Start extends State {
00009:
00010:   /** The Constant ID. */
00011:   private static final int ID = -1;
00012:
00013:   /**
00014:    * Instantiates a new Start.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public Start(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:   /*
00023:    * (non-Javadoc)
00024:    *
00025:    * @see iit.softwaresystarch.project.model.states.State#getID()
00026:    */
00027:   @Override
00028:   public int getID() {
00029:     return Start.ID;
00030:   }
00031:
00032:   /*
00033:    * Implementation of the activate method.
```

```
00034:    *
00035:    * @see iit.softwaresystarch.project.model.states.State#activate()
00036:    */
00037:   @Override
00038:   public void activate() {
00039:     this.operation.storeData();
00040:   }
00041:
00042:   /*
00043:    * (non-Javadoc)
00044:    *
00045:    * @see iit.softwaresystarch.project.model.states.State#toString()
00046:    */
00047:   @Override
00048:   public String toString() {
00049:     return "Start [getID()=" + this.getID() + "]";
00050:   }
00051:
00052:
00053: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.actions.Operation;
00004: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00005:
00006: /**
00007:  * The abstract Class State.
00008:  */
00009: public abstract class State {
00010:
00011:   /** The Constant ID. */
00012:   private static final int ID = -2;
00013:
00014:   /**
00015:    * Instantiates a new State.
00016:    *
00017:    * @param operationAbstractFactory the operation abstract factory
00018:    */
00019:   public State(OperationAbstractFactory operationAbstractFactory) {
00020:     this.operation = new Operation(operationAbstractFactory);
00021:   }
00022:
00023:   /** The operation. */
00024:   protected Operation operation;
00025:
00026:   /**
00027:    * Gets the id.
00028:    *
00029:    * @return the id
00030:    */
00031:   public int getID() {
00032:     return State.ID;
00033:   }
```

```
00034:
00035:   /**
00036:    * Activate.
00037:    */
00038:   public void activate() {}
00039:
00040:   /**
00041:    * Start.
00042:    */
00043:   public void start() {}
00044:
00045:   /**
00046:    * Pay type.
00047:    *
00048:    * @param t the payment type id
00049:    */
00050:   public void payType(int t) {}
00051:
00052:   /**
00053:    * Approved.
00054:    */
00055:   public void approved() {}
00056:
00057:   /**
00058:    * Reject.
00059:    */
00060:   public void reject() {}
00061:
00062:   /**
00063:    * Cancel.
00064:    */
00065:   public void cancel() {}
00066:
```

```
00067:   /**
00068:    * Select gas.
00069:    *
00070:    * @param g the gas id
00071:    */
00072:   public void selectGas(int g) {}
00073:
00074:   /**
00075:    * Start pump.
00076:    */
00077:   public void startPump() {}
00078:
00079:   /**
00080:    * Pump.
00081:    */
00082:   public void pump() {}
00083:
00084:   /**
00085:    * Stop pump.
00086:    */
00087:   public void stopPump() {}
00088:
00089:   /**
00090:    * Receipt.
00091:    */
00092:   public void receipt() {}
00093:
00094:   /**
00095:    * No receipt.
00096:    */
00097:   public void noReceipt() {}
00098:
00099:   /*
```

```
00100:    * (non-Javadoc)
00101:    *
00102:    * @see java.lang.Object#toString()
00103:    */
00104:   @Override
00105:   public String toString() {
00106:     return "State [operation=" + this.operation + ", getID()=" + this.getID() + "]";
00107:   }
00108:
00109: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class State0.
00007:  */
00008: public class State0 extends State {
00009:
00010:   /** The Constant ID. */
00011:   private static final int ID = 0;
00012:
00013:   /**
00014:    * Instantiates a new State0.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public State0(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:   /*
00023:    * (non-Javadoc)
00024:    *
00025:    * @see iit.softwaresystarch.project.model.states.State#getID()
00026:    */
00027:   @Override
00028:   public int getID() {
00029:     return State0.ID;
00030:   }
00031:
00032:   /*
00033:    * Implementation of the start method.
```

```
00034:    *
00035:    * @see iit.softwaresystarch.project.model.states.State#start()
00036:    */
00037:   @Override
00038:   public void start() {
00039:     this.operation.payMsg();
00040:   }
00041:
00042:   /*
00043:    * (non-Javadoc)
00044:    *
00045:    * @see iit.softwaresystarch.project.model.states.State#toString()
00046:    */
00047:   @Override
00048:   public String toString() {
00049:     return "State0 [getID()=" + this.getID() + "]";
00050:   }
00051:
00052: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class State1.
00007:  */
00008: public class State1 extends State {
00009:
00010:   /** The Constant ID. */
00011:   private static final int ID = 1;
00012:
00013:   /**
00014:    * Instantiates a new State1.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public State1(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:   /*
00023:    * (non-Javadoc)
00024:    *
00025:    * @see iit.softwaresystarch.project.model.states.State#getID()
00026:    */
00027:   @Override
00028:   public int getID() {
00029:     return State1.ID;
00030:   }
00031:
00032:   /*
00033:    * Implementation of the payType method.
```

```
00034:    *
00035:    * @see iit.softwaresystarch.project.model.states.State#payType(int)
00036:    */
00037:   @Override
00038:   public void payType(int t) {
00039:     if (t == 2) {
00040:       this.operation.storeCash();
00041:       this.operation.displayMenu();
00042:     }
00043:   }
00044:
00045:   /*
00046:    * (non-Javadoc)
00047:    *
00048:    * @see iit.softwaresystarch.project.model.states.State#toString()
00049:    */
00050:   @Override
00051:   public String toString() {
00052:     return "State1 [getID()=" + this.getID() + "]";
00053:   }
00054:
00055: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class State2.
00007:  */
00008: public class State2 extends State {
00009:
00010:   /** The Constant ID. */
00011:   private static final int ID = 2;
00012:
00013:   /**
00014:    * Instantiates a new State2.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public State2(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:   /*
00023:    * (non-Javadoc)
00024:    *
00025:    * @see iit.softwaresystarch.project.model.states.State#getID()
00026:    */
00027:   @Override
00028:   public int getID() {
00029:     return State2.ID;
00030:   }
00031:
00032:   /*
00033:    * Implementation of the approved method.
```

```
00034:    *
00035:    * @see iit.softwaresystarch.project.model.states.State#approved()
00036:    */
00037:   @Override
00038:   public void approved() {
00039:     this.operation.displayMenu();
00040:   }
00041:
00042:   /*
00043:    * (non-Javadoc)
00044:    *
00045:    * @see iit.softwaresystarch.project.model.states.State#reject()
00046:    */
00047:   @Override
00048:   public void reject() {
00049:     this.operation.rejectMsg();
00050:   }
00051:
00052:   /*
00053:    * (non-Javadoc)
00054:    *
00055:    * @see iit.softwaresystarch.project.model.states.State#toString()
00056:    */
00057:   @Override
00058:   public String toString() {
00059:     return "State2 [getID()=" + this.getID() + "]";
00060:   }
00061:
00062: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class State3.
00007:  */
00008: public class State3 extends State {
00009:
00010:   /** The Constant ID. */
00011:   private static final int ID = 3;
00012:
00013:   /**
00014:    * Instantiates a new State3.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public State3(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:   /*
00023:    * (non-Javadoc)
00024:    *
00025:    * @see iit.softwaresystarch.project.model.states.State#getID()
00026:    */
00027:   @Override
00028:   public int getID() {
00029:     return State3.ID;
00030:   }
00031:
00032:   /*
00033:    * Implementation of the cancel method.
```

```
00034:    *
00035:    * @see iit.softwaresystarch.project.model.states.State#cancel()
00036:    */
00037:   @Override
00038:   public void cancel() {
00039:     this.operation.cancelMsg();
00040:   }
00041:
00042:   /*
00043:    * Implementation of the select gas method.
00044:    *
00045:    * @see iit.softwaresystarch.project.model.states.State#selectGas(int)
00046:    */
00047:   @Override
00048:   public void selectGas(int g) {
00049:     this.operation.setPrice(g);
00050:   }
00051:
00052:   /*
00053:    * (non-Javadoc)
00054:    *
00055:    * @see iit.softwaresystarch.project.model.states.State#toString()
00056:    */
00057:   @Override
00058:   public String toString() {
00059:     return "State3 [getID()=" + this.getID() + "]";
00060:   }
00061:
00062: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class State4.
00007:  */
00008: public class State4 extends State {
00009:
00010:   /** The Constant ID. */
00011:   private static final int ID = 4;
00012:
00013:   /**
00014:    * Instantiates a new State4.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public State4(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:   /*
00023:    * (non-Javadoc)
00024:    *
00025:    * @see iit.softwaresystarch.project.model.states.State#getID()
00026:    */
00027:   @Override
00028:   public int getID() {
00029:     return State4.ID;
00030:   }
00031:
00032:   /*
00033:    * Implementation of the start pump method.
```

```
00034:    *
00035:    * @see iit.softwaresystarch.project.model.states.State#startPump()
00036:    */
00037:   @Override
00038:   public void startPump() {
00039:     this.operation.setInitialValues();
00040:     this.operation.readyMsg();
00041:   }
00042:
00043:   /*
00044:    * (non-Javadoc)
00045:    *
00046:    * @see iit.softwaresystarch.project.model.states.State#toString()
00047:    */
00048:   @Override
00049:   public String toString() {
00050:     return "State4 [getID()=" + this.getID() + "]";
00051:   }
00052:
00053: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class State5.
00007:  */
00008: public class State5 extends State {
00009:
00010:   /** The Constant ID. */
00011:   private static final int ID = 5;
00012:
00013:   /**
00014:    * Instantiates a new State5.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public State5(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:   /*
00023:    * (non-Javadoc)
00024:    *
00025:    * @see iit.softwaresystarch.project.model.states.State#getID()
00026:    */
00027:   @Override
00028:   public int getID() {
00029:     return State5.ID;
00030:   }
00031:
00032:   /*
00033:    * Implementation of the pump method.
```

```
00034:    *
00035:    * @see iit.softwaresystarch.project.model.states.State#pump()
00036:    */
00037:   @Override
00038:   public void pump() {
00039:     this.operation.pumpGasUnit();
00040:   }
00041:
00042:   /*
00043:    * Implementation of the stop pump method.
00044:    *
00045:    * @see iit.softwaresystarch.project.model.states.State#stopPump()
00046:    */
00047:   @Override
00048:   public void stopPump() {
00049:     this.operation.stopMsg();
00050:   }
00051:
00052:   /*
00053:    * (non-Javadoc)
00054:    *
00055:    * @see iit.softwaresystarch.project.model.states.State#toString()
00056:    */
00057:   @Override
00058:   public String toString() {
00059:     return "State5 [getID()=" + this.getID() + "]";
00060:   }
00061:
00062: }
```

```
00001: package iit.softwaresystarch.project.model.states;
00002:
00003: import iit.softwaresystarch.project.model.factories.OperationAbstractFactory;
00004:
00005: /**
00006:  * The Class State6.
00007:  */
00008: public class State6 extends State {
00009:
00010:   /** The id. */
00011:   private static int ID = 6;
00012:
00013:   /**
00014:    * Instantiates a new State6.
00015:    *
00016:    * @param operationAbstractFactory the operation abstract factory
00017:    */
00018:   public State6(OperationAbstractFactory operationAbstractFactory) {
00019:     super(operationAbstractFactory);
00020:   }
00021:
00022:
00023:   /*
00024:    * (non-Javadoc)
00025:    *
00026:    * @see iit.softwaresystarch.project.model.states.State#getID()
00027:    */
00028:   @Override
00029:   public int getID() {
00030:     return State6.ID;
00031:   }
00032:
00033:   /*
```

```
00034:    * Implementation of the receipt method.
00035:    *
00036:    * @see iit.softwaresystarch.project.model.states.State#receipt()
00037:    */
00038:   @Override
00039:   public void receipt() {
00040:     this.operation.printReceipt();
00041:     this.operation.returnCash();
00042:   }
00043:
00044:   /*
00045:    * Implementation of the no receipt method.
00046:    *
00047:    * @see iit.softwaresystarch.project.model.states.State#noReceipt()
00048:    */
00049:   @Override
00050:   public void noReceipt() {
00051:     this.operation.returnCash();
00052:   }
00053:
00054:   /*
00055:    * (non-Javadoc)
00056:    *
00057:    * @see iit.softwaresystarch.project.model.states.State#toString()
00058:    */
00059:   @Override
00060:   public String toString() {
00061:     return "State6 [getID()=" + this.getID() + "]";
00062:   }
00063:
00064: }
```

```
00001: package iit.softwaresystarch.project.view;
00002:
00003: import java.awt.BorderLayout;
00004: import java.awt.Dimension;
00005: import java.awt.EventQueue;
00006: import java.awt.GridLayout;
00007: import java.io.ByteArrayOutputStream;
00008: import java.io.File;
00009: import java.io.IOException;
00010: import java.io.PrintStream;
00011: import java.lang.management.ManagementFactory;
00012: import java.nio.charset.StandardCharsets;
00013:
00014: import javax.swing.JFrame;
00015: import javax.swing.JLabel;
00016: import javax.swing.JScrollPane;
00017: import javax.swing.JTextPane;
00018: import javax.swing.SwingConstants;
00019:
00020: import iit.softwaresystarch.project.model.efsms.GasPump1;
00021: import iit.softwaresystarch.project.model.efsms.GasPump2;
00022:
00023: /**
00024:  * The Class Driver.
00025:  */
00026: public class Driver {
00027:
00028:   /** The frame. */
00029:   private JFrame frame;
00030:
00031:   /** The gas pump 1 panel. */
00032:   private GasPump1Panel gasPump1Panel;
00033:
```

```
00034:   /** The gas pump 2 panel. */
00035:   private GasPump2Panel gasPump2Panel;
00036:
00037:   /** The gas pump 1. */
00038:   private GasPump1 gasPump1;
00039:
00040:   /** The gas pump 2. */
00041:   private GasPump2 gasPump2;
00042:
00043:   /** The text pane output. */
00044:   private JTextPane textPaneOutput;
00045:
00046:   /** The input panel. */
00047:   private InputPanel inputPanel;
00048:
00049:   /** The orig out. */
00050:   private PrintStream origOut;
00051:
00052:   /** The baos. */
00053:   private ByteArrayOutputStream baos;
00054:
00055:   /** The out. */
00056:   private PrintStream out;
00057:
00058:   /**
00059:    * The main method to create the view.
00060:    *
00061:    * @param args the arguments
00062:    */
00063:   public static void main(String[] args) {
00064:     EventQueue.invokeLater(() -> {
00065:       try {
00066:         final Driver window = new Driver();
```

```
00067:         window.frame.setVisible(true);
00068:       } catch (final Exception e) {
00069:         e.printStackTrace();
00070:       }
00071:     });
00072:   }
00073:
00074:   /**
00075:    * Instantiates a new Driver.
00076:    */
00077:   public Driver() {
00078:     this.initialize();
00079:   }
00080:
00081:   /**
00082:    * Initialize the driver.
00083:    */
00084:   private void initialize() {
00085:     this.frame = new JFrame();
00086:     this.frame.setBounds(100, 100, 550, 500);
00087:     this.frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
00088:     this.frame.getContentPane().setLayout(new BorderLayout(0, 0));
00089:     this.frame.setResizable(false);
00090:     this.frame.isAlwaysOnTop();
00091:
00092:     final JLabel labelCsPumpDriver = new JLabel("CS586 Pump Driver");
00093:     labelCsPumpDriver.setHorizontalAlignment(SwingConstants.CENTER);
00094:     this.frame.getContentPane().add(labelCsPumpDriver, BorderLayout.NORTH);
00095:
00096:     this.textPaneOutput = new JTextPane();
00097:     this.textPaneOutput.setToolTipText("The output of the GasPump");
00098:     this.textPaneOutput.setEditable(false);
00099:     this.textPaneOutput.setPreferredSize(new Dimension(550, 100));
```

```
00100:
00101:     final JScrollPane scrollPane = new JScrollPane(this.textPaneOutput);
00102:     this.frame.getContentPane().add(scrollPane, BorderLayout.SOUTH);
00103:
00104:     this.inputPanel = new InputPanel();
00105:     this.inputPanel.getButtonReset().setText("Restart driver");
00106:
00107:     this.inputPanel.getButtonPrintPump1().addActionListener(e -> {
00108:       Driver.this.startPrint();
00109:       System.out.println("########### PRINTING GAS PUMP 1 ###########");
00110:       System.out.println(Driver.this.gasPump1);
00111:       System.out.println("########### GAS PUMP 1 PRINTED ###########");
00112:       Driver.this.endPrint();
00113:     });
00114:
00115:     this.inputPanel.getButtonPrintPump2().addActionListener(e -> {
00116:       Driver.this.startPrint();
00117:       System.out.println("########### PRINTING GAS PUMP 2 ###########");
00118:       System.out.println(Driver.this.gasPump2);
00119:       System.out.println("########### GAS PUMP 2 PRINTED ###########");
00120:       Driver.this.endPrint();
00121:     });
00122:
00123:     this.inputPanel.getButtonReset().addActionListener(e -> Driver.this.rebootDriver());
00124:
00125:     this.frame.getContentPane().add(this.inputPanel, BorderLayout.CENTER);
00126:     this.inputPanel.setLayout(new GridLayout(0, 2, 0, 0));
00127:
00128:     this.gasPump1 = new GasPump1();
00129:     this.gasPump1Panel = new GasPump1Panel(this.gasPump1, this.textPaneOutput, this.inputPanel);
00130:     this.frame.getContentPane().add(this.gasPump1Panel, BorderLayout.WEST);
00131:
00132:     this.gasPump2 = new GasPump2();
```

```
00133:     this.gasPump2Panel = new GasPump2Panel(this.gasPump2, this.textPaneOutput, this.inputPanel);
00134:
00135:     this.frame.getContentPane().add(this.gasPump2Panel, BorderLayout.EAST);
00136:
00137:   }
00138:
00139:   /**
00140:    * Redirect the system output to a prinstream.
00141:    */
00142:   private void startPrint() {
00143:     this.origOut = System.out;
00144:     this.baos = new ByteArrayOutputStream();
00145:     this.out = new PrintStream(this.baos);
00146:     System.setOut(this.out);
00147:   }
00148:
00149:   /**
00150:    * Print the printstream into the textPaneOutput.
00151:    */
00152:   private void endPrint() {
00153:     // Update the text field
00154:     this.textPaneOutput.setText(this.textPaneOutput.getText()
00155:         + new String(this.baos.toByteArray(), StandardCharsets.UTF_8));
00156:     // Get back to the original system output
00157:     System.setOut(this.origOut);
00158:   }
00159:
00160:   /**
00161:    * Reboot the driver.
00162:    */
00163:   private void rebootDriver() {
00164:     final StringBuilder cmd = new StringBuilder();
00165:     cmd.append(System.getProperty("java.home") + File.separator + "bin" + File.separator + "java ");
```

```
00166:     for (final String jvmArg : ManagementFactory.getRuntimeMXBean().getInputArguments()) {
00167:       cmd.append(jvmArg + " ");
00168:     }
00169:     cmd.append("-cp ").append(ManagementFactory.getRuntimeMXBean().getClassPath()).append(" ");
00170:     cmd.append(Driver.class.getName()).append(" ");
00171:
00172:     try {
00173:       Runtime.getRuntime().exec(cmd.toString());
00174:     } catch (final IOException ioe) {
00175:       ioe.printStackTrace();
00176:     }
00177:     System.exit(0);
00178:   }
00179:
00180: }
```

```
00001: package iit.softwaresystarch.project.view;
00002:
00003: import java.awt.GridLayout;
00004:
00005: import javax.swing.JButton;
00006: import javax.swing.JLabel;
00007: import javax.swing.JPanel;
00008: import javax.swing.JTextPane;
00009: import javax.swing.SwingConstants;
00010:
00011: import iit.softwaresystarch.project.controller.ButtonActivatePump1ClickedListener;
00012: import iit.softwaresystarch.project.controller.ButtonClickedListener;
00013: import iit.softwaresystarch.project.model.efsms.GasPump1;
00014:
00015: /**
00016:  * The Class GasPump1Panel.
00017:  */
00018: public class GasPump1Panel extends JPanel {
00019:
00020:   /** The Constant serialVersionUID. */
00021:   private static final long serialVersionUID = 1L;
00022:
00023:   /** The gas pump 1. */
00024:   private final GasPump1 gasPump1;
00025:
00026:   /** The label gas pump 1. */
00027:   private final JLabel labelGasPump1;
00028:
00029:   /** The button activate. */
00030:   private final JButton buttonActivate;
00031:
00032:   /** The button start. */
00033:   private final JButton buttonStart;
```

```
00034:
00035:   /** The button pay credit. */
00036:   private final JButton buttonPayCredit;
00037:
00038:   /** The button approved. */
00039:   private final JButton buttonApproved;
00040:
00041:   /** The button reject. */
00042:   private final JButton buttonReject;
00043:
00044:   /** The button regular. */
00045:   private final JButton buttonRegular;
00046:
00047:   /** The button super. */
00048:   private final JButton buttonSuper;
00049:
00050:   /** The button start pump. */
00051:   private final JButton buttonStartPump;
00052:
00053:   /** The button pump gallon. */
00054:   private final JButton buttonPumpGallon;
00055:
00056:   /** The button stop pump. */
00057:   private final JButton buttonStopPump;
00058:
00059:   /** The button cancel. */
00060:   private final JButton buttonCancel;
00061:
00062:   /** The text pane output. */
00063:   private final JTextPane textPaneOutput;
00064:
00065:   /** The input panel. */
00066:   private final InputPanel inputPanel;
```

```
00067:
00068:   /**
00069:    * Instantiates a new GasPump1Panel, add the buttons and controllers.
00070:    *
00071:    * @param gasPump1 the gas pump 1
00072:    * @param textPaneOutput the text pane output
00073:    * @param inputPanel the input panel
00074:    */
00075:   public GasPump1Panel(GasPump1 gasPump1, JTextPane textPaneOutput, InputPanel inputPanel) {
00076:     this.setLayout(new GridLayout(0, 1, 0, 0));
00077:
00078:     this.labelGasPump1 = new JLabel("GasPump-1");
00079:     this.labelGasPump1.setHorizontalAlignment(SwingConstants.CENTER);
00080:     this.add(this.labelGasPump1);
00081:
00082:     this.buttonActivate = new JButton("Activate");
00083:     this.add(this.buttonActivate);
00084:
00085:     this.buttonStart = new JButton("Start");
00086:     this.add(this.buttonStart);
00087:
00088:     this.buttonPayCredit = new JButton("Pay Credit");
00089:     this.add(this.buttonPayCredit);
00090:
00091:     this.buttonApproved = new JButton("Approved");
00092:     this.add(this.buttonApproved);
00093:
00094:     this.buttonReject = new JButton("Reject");
00095:     this.add(this.buttonReject);
00096:
00097:     this.buttonCancel = new JButton("Cancel");
00098:     this.add(this.buttonCancel);
00099:
```

```
00100:     this.buttonRegular = new JButton("Regular");
00101:     this.add(this.buttonRegular);
00102:
00103:     this.buttonSuper = new JButton("Super");
00104:     this.add(this.buttonSuper);
00105:
00106:     this.buttonStartPump = new JButton("StartPump");
00107:     this.add(this.buttonStartPump);
00108:
00109:     this.buttonPumpGallon = new JButton("PumpGallon");
00110:     this.add(this.buttonPumpGallon);
00111:
00112:     this.buttonStopPump = new JButton("StopPump");
00113:     this.add(this.buttonStopPump);
00114:
00115:     this.gasPump1 = gasPump1;
00116:
00117:     this.textPaneOutput = textPaneOutput;
00118:     this.inputPanel = inputPanel;
00119:
00120:     this.addControllers();
00121:   }
00122:
00123:   /**
00124:    * Adds the controllers.
00125:    */
00126:   private void addControllers() {
00127:     this.buttonActivate.addActionListener(new ButtonActivatePump1ClickedListener(this.gasPump1,
00128:         this.textPaneOutput, this.inputPanel));
00129:
00130:     this.buttonStart
00131:         .addActionListener(new ButtonClickedListener(this.gasPump1, "start", this.textPaneOutput));
00132:
```

```
00133:     this.buttonPayCredit.addActionListener(
00134:         new ButtonClickedListener(this.gasPump1, "payCredit", this.textPaneOutput));
00135:
00136:     this.buttonApproved.addActionListener(
00137:         new ButtonClickedListener(this.gasPump1, "approved", this.textPaneOutput));
00138:
00139:     this.buttonReject
00140:         .addActionListener(new ButtonClickedListener(this.gasPump1, "reject", this.textPaneOutput));
00141:
00142:     this.buttonCancel
00143:         .addActionListener(new ButtonClickedListener(this.gasPump1, "cancel", this.textPaneOutput));
00144:
00145:     this.buttonRegular.addActionListener(
00146:         new ButtonClickedListener(this.gasPump1, "regular", this.textPaneOutput));
00147:
00148:     this.buttonSuper.addActionListener(
00149:         new ButtonClickedListener(this.gasPump1, "superGas", this.textPaneOutput));
00150:
00151:     this.buttonStartPump.addActionListener(
00152:         new ButtonClickedListener(this.gasPump1, "startPump", this.textPaneOutput));
00153:
00154:     this.buttonPumpGallon.addActionListener(
00155:         new ButtonClickedListener(this.gasPump1, "pumpGallon", this.textPaneOutput));
00156:
00157:     this.buttonStopPump.addActionListener(
00158:         new ButtonClickedListener(this.gasPump1, "stopPump", this.textPaneOutput));
00159:   }
00160:
00161:   /**
00162:    * Gets the gas pump 1.
00163:    *
00164:    * @return the gas pump 1
00165:    */
```

```
00166:   public GasPump1 getGasPump1() {
00167:     return this.gasPump1;
00168:   }
00169:
00170:   /**
00171:    * Gets the label gas pump 1.
00172:    *
00173:    * @return the label gas pump 1
00174:    */
00175:   public JLabel getLabelGasPump1() {
00176:     return this.labelGasPump1;
00177:   }
00178:
00179:   /**
00180:    * Gets the button activate.
00181:    *
00182:    * @return the button activate
00183:    */
00184:   public JButton getButtonActivate() {
00185:     return this.buttonActivate;
00186:   }
00187:
00188:   /**
00189:    * Gets the button start.
00190:    *
00191:    * @return the button start
00192:    */
00193:   public JButton getButtonStart() {
00194:     return this.buttonStart;
00195:   }
00196:
00197:   /**
00198:    * Gets the button pay credit.
```

```
00199:    *
00200:    * @return the button pay credit
00201:    */
00202:   public JButton getButtonPayCredit() {
00203:     return this.buttonPayCredit;
00204:   }
00205:
00206:   /**
00207:    * Gets the button approved.
00208:    *
00209:    * @return the button approved
00210:    */
00211:   public JButton getButtonApproved() {
00212:     return this.buttonApproved;
00213:   }
00214:
00215:   /**
00216:    * Gets the button reject.
00217:    *
00218:    * @return the button reject
00219:    */
00220:   public JButton getButtonReject() {
00221:     return this.buttonReject;
00222:   }
00223:
00224:   /**
00225:    * Gets the button regular.
00226:    *
00227:    * @return the button regular
00228:    */
00229:   public JButton getButtonRegular() {
00230:     return this.buttonRegular;
00231:   }
```

```
00232:
00233:   /**
00234:    * Gets the button super.
00235:    *
00236:    * @return the button super
00237:    */
00238:   public JButton getButtonSuper() {
00239:     return this.buttonSuper;
00240:   }
00241:
00242:   /**
00243:    * Gets the button start pump.
00244:    *
00245:    * @return the button start pump
00246:    */
00247:   public JButton getButtonStartPump() {
00248:     return this.buttonStartPump;
00249:   }
00250:
00251:   /**
00252:    * Gets the button pump gallon.
00253:    *
00254:    * @return the button pump gallon
00255:    */
00256:   public JButton getButtonPumpGallon() {
00257:     return this.buttonPumpGallon;
00258:   }
00259:
00260:   /**
00261:    * Gets the button stop pump.
00262:    *
00263:    * @return the button stop pump
00264:    */
```

```
00265:   public JButton getButtonStopPump() {
00266:     return this.buttonStopPump;
00267:   }
00268:
00269:   /**
00270:    * Gets the button cancel.
00271:    *
00272:    * @return the button cancel
00273:    */
00274:   public JButton getButtonCancel() {
00275:     return this.buttonCancel;
00276:   }
00277:
00278: }
```

```
00001: package iit.softwaresystarch.project.view;
00002:
00003: import java.awt.GridLayout;
00004:
00005: import javax.swing.JButton;
00006: import javax.swing.JLabel;
00007: import javax.swing.JPanel;
00008: import javax.swing.JTextPane;
00009: import javax.swing.SwingConstants;
00010:
00011: import iit.softwaresystarch.project.controller.ButtonActivatePump2ClickedListener;
00012: import iit.softwaresystarch.project.controller.ButtonCashClickedListener;
00013: import iit.softwaresystarch.project.controller.ButtonClickedListener;
00014: import iit.softwaresystarch.project.model.efsms.GasPump2;
00015:
00016: /**
00017:  * The Class GasPump2Panel.
00018:  */
00019: public class GasPump2Panel extends JPanel {
00020:
00021:   /** The Constant serialVersionUID. */
00022:   private static final long serialVersionUID = 1L;
00023:
00024:   /** The gas pump 2. */
00025:   private final GasPump2 gasPump2;
00026:
00027:   /** The label gas pump 2. */
00028:   private final JLabel labelGasPump2;
00029:
00030:   /** The button activate. */
00031:   private final JButton buttonActivate;
00032:
00033:   /** The button start. */
```

```
00034:   private final JButton buttonStart;
00035:
00036:   /** The button pay cash. */
00037:   private final JButton buttonPayCash;
00038:
00039:   /** The button cancel. */
00040:   private final JButton buttonCancel;
00041:
00042:   /** The button premium. */
00043:   private final JButton buttonPremium;
00044:
00045:   /** The button regular. */
00046:   private final JButton buttonRegular;
00047:
00048:   /** The button super. */
00049:   private final JButton buttonSuper;
00050:
00051:   /** The button start pump. */
00052:   private final JButton buttonStartPump;
00053:
00054:   /** The button pump liter. */
00055:   private final JButton buttonPumpLiter;
00056:
00057:   /** The button stop pump. */
00058:   private final JButton buttonStopPump;
00059:
00060:   /** The button no receipt. */
00061:   private final JButton buttonNoReceipt;
00062:
00063:   /** The button receipt. */
00064:   private final JButton buttonReceipt;
00065:
00066:   /** The text pane output. */
```

```
00067:   private final JTextPane textPaneOutput;
00068:
00069:   /** The input panel. */
00070:   private final InputPanel inputPanel;
00071:
00072:   /**
00073:    * Instantiates a new GasPump2Panel, add the buttons and controllers.
00074:    *
00075:    * @param gasPump2 the gas pump 2
00076:    * @param textPaneOutput the text pane output
00077:    * @param inputPanel the input panel
00078:    */
00079:   public GasPump2Panel(GasPump2 gasPump2, JTextPane textPaneOutput, InputPanel inputPanel) {
00080:     this.setLayout(new GridLayout(0, 1, 0, 0));
00081:
00082:     this.labelGasPump2 = new JLabel("GasPump-2");
00083:     this.labelGasPump2.setHorizontalAlignment(SwingConstants.CENTER);
00084:     this.add(this.labelGasPump2);
00085:
00086:     this.buttonActivate = new JButton("Activate");
00087:     this.add(this.buttonActivate);
00088:
00089:     this.buttonStart = new JButton("Start");
00090:     this.add(this.buttonStart);
00091:
00092:     this.buttonPayCash = new JButton("Pay Cash");
00093:     this.add(this.buttonPayCash);
00094:
00095:     this.buttonCancel = new JButton("Cancel");
00096:     this.add(this.buttonCancel);
00097:
00098:     this.buttonSuper = new JButton("Super");
00099:     this.add(this.buttonSuper);
```

```
00100:
00101:     this.buttonRegular = new JButton("Regular");
00102:     this.add(this.buttonRegular);
00103:
00104:     this.buttonPremium = new JButton("Premium");
00105:     this.add(this.buttonPremium);
00106:
00107:     this.buttonStartPump = new JButton("StartPump");
00108:     this.add(this.buttonStartPump);
00109:
00110:     this.buttonPumpLiter = new JButton("PumpLiter");
00111:     this.add(this.buttonPumpLiter);
00112:
00113:     this.buttonStopPump = new JButton("Stop");
00114:     this.add(this.buttonStopPump);
00115:
00116:     this.buttonNoReceipt = new JButton("NoReceipt");
00117:     this.add(this.buttonNoReceipt);
00118:
00119:     this.buttonReceipt = new JButton("Receipt");
00120:     this.add(this.buttonReceipt);
00121:
00122:     this.gasPump2 = gasPump2;
00123:
00124:     this.textPaneOutput = textPaneOutput;
00125:     this.inputPanel = inputPanel;
00126:
00127:     this.addControllers();
00128:   }
00129:
00130:   /**
00131:    * Adds the controllers.
00132:    */
```

```
00133:   private void addControllers() {
00134:     this.buttonActivate.addActionListener(new ButtonActivatePump2ClickedListener(this.gasPump2,
00135:         this.textPaneOutput, this.inputPanel));
00136:
00137:     this.buttonStart
00138:         .addActionListener(new ButtonClickedListener(this.gasPump2, "start", this.textPaneOutput));
00139:
00140:     this.buttonPayCash.addActionListener(
00141:         new ButtonCashClickedListener(this.gasPump2, this.textPaneOutput, this.inputPanel));
00142:
00143:     this.buttonCancel
00144:         .addActionListener(new ButtonClickedListener(this.gasPump2, "cancel", this.textPaneOutput));
00145:
00146:     this.buttonSuper.addActionListener(
00147:         new ButtonClickedListener(this.gasPump2, "superGas", this.textPaneOutput));
00148:
00149:     this.buttonRegular.addActionListener(
00150:         new ButtonClickedListener(this.gasPump2, "regular", this.textPaneOutput));
00151:
00152:     this.buttonPremium.addActionListener(
00153:         new ButtonClickedListener(this.gasPump2, "premium", this.textPaneOutput));
00154:
00155:     this.buttonStartPump.addActionListener(
00156:         new ButtonClickedListener(this.gasPump2, "startPump", this.textPaneOutput));
00157:
00158:     this.buttonPumpLiter.addActionListener(
00159:         new ButtonClickedListener(this.gasPump2, "pumpLiter", this.textPaneOutput));
00160:
00161:     this.buttonStopPump
00162:         .addActionListener(new ButtonClickedListener(this.gasPump2, "stop", this.textPaneOutput));
00163:
00164:     this.buttonNoReceipt.addActionListener(
00165:         new ButtonClickedListener(this.gasPump2, "noReceipt", this.textPaneOutput));
```

```
00166:
00167:     this.buttonReceipt.addActionListener(
00168:         new ButtonClickedListener(this.gasPump2, "receipt", this.textPaneOutput));
00169:   }
00170:
00171:   /**
00172:    * Gets the gas pump 2.
00173:    *
00174:    * @return the gas pump 2
00175:    */
00176:   public GasPump2 getGasPump2() {
00177:     return this.gasPump2;
00178:   }
00179:
00180:   /**
00181:    * Gets the label gas pump 2.
00182:    *
00183:    * @return the label gas pump 2
00184:    */
00185:   public JLabel getLabelGasPump2() {
00186:     return this.labelGasPump2;
00187:   }
00188:
00189:   /**
00190:    * Gets the button activate.
00191:    *
00192:    * @return the button activate
00193:    */
00194:   public JButton getButtonActivate() {
00195:     return this.buttonActivate;
00196:   }
00197:
00198:   /**
```

```
00199:    * Gets the button start.
00200:    *
00201:    * @return the button start
00202:    */
00203:   public JButton getButtonStart() {
00204:     return this.buttonStart;
00205:   }
00206:
00207:   /**
00208:    * Gets the button pay cash.
00209:    *
00210:    * @return the button pay cash
00211:    */
00212:   public JButton getButtonPayCash() {
00213:     return this.buttonPayCash;
00214:   }
00215:
00216:   /**
00217:    * Gets the button cancel.
00218:    *
00219:    * @return the button cancel
00220:    */
00221:   public JButton getButtonCancel() {
00222:     return this.buttonCancel;
00223:   }
00224:
00225:   /**
00226:    * Gets the button premium.
00227:    *
00228:    * @return the button premium
00229:    */
00230:   public JButton getButtonPremium() {
00231:     return this.buttonPremium;
```

```
00232:   }
00233:
00234:   /**
00235:    * Gets the button regular.
00236:    *
00237:    * @return the button regular
00238:    */
00239:   public JButton getButtonRegular() {
00240:     return this.buttonRegular;
00241:   }
00242:
00243:   /**
00244:    * Gets the button super.
00245:    *
00246:    * @return the button super
00247:    */
00248:   public JButton getButtonSuper() {
00249:     return this.buttonSuper;
00250:   }
00251:
00252:   /**
00253:    * Gets the button start pump.
00254:    *
00255:    * @return the button start pump
00256:    */
00257:   public JButton getButtonStartPump() {
00258:     return this.buttonStartPump;
00259:   }
00260:
00261:   /**
00262:    * Gets the button pump liter.
00263:    *
00264:    * @return the button pump liter
```

```
00265:    */
00266:   public JButton getButtonPumpLiter() {
00267:     return this.buttonPumpLiter;
00268:   }
00269:
00270:   /**
00271:    * Gets the button stop pump.
00272:    *
00273:    * @return the button stop pump
00274:    */
00275:   public JButton getButtonStopPump() {
00276:     return this.buttonStopPump;
00277:   }
00278:
00279:   /**
00280:    * Gets the button no receipt.
00281:    *
00282:    * @return the button no receipt
00283:    */
00284:   public JButton getButtonNoReceipt() {
00285:     return this.buttonNoReceipt;
00286:   }
00287:
00288:   /**
00289:    * Gets the button receipt.
00290:    *
00291:    * @return the button receipt
00292:    */
00293:   public JButton getButtonReceipt() {
00294:     return this.buttonReceipt;
00295:   }
00296:
00297: }
```

```
00001: package iit.softwaresystarch.project.view;
00002:
00003: import javax.swing.JButton;
00004: import javax.swing.JFormattedTextField;
00005: import javax.swing.JLabel;
00006: import javax.swing.JPanel;
00007: import javax.swing.SwingConstants;
00008:
00009: /**
00010:  * The Class InputPanel.
00011:  */
00012: public class InputPanel extends JPanel {
00013:
00014:   /** The Constant serialVersionUID. */
00015:   private static final long serialVersionUID = 1L;
00016:
00017:   /** The label A. */
00018:   private final JLabel labelA;
00019:
00020:   /** The label B. */
00021:   private final JLabel labelB;
00022:
00023:   /** The label C. */
00024:   private final JLabel labelC;
00025:
00026:   /** The label cash. */
00027:   private final JLabel labelCash;
00028:
00029:   /** The a input. */
00030:   private final JFormattedTextField aInput;
00031:
00032:   /** The b input. */
00033:   private final JFormattedTextField bInput;
```

```
00034:
00035:   /** The c input. */
00036:   private final JFormattedTextField cInput;
00037:
00038:   /** The cash input. */
00039:   private final JFormattedTextField cashInput;
00040:
00041:   /** The button print pump 2. */
00042:   private final JButton buttonPrintPump2;
00043:
00044:   /** The button print pump 1. */
00045:   private final JButton buttonPrintPump1;
00046:
00047:   /** The label print. */
00048:   private final JLabel labelPrint;
00049:
00050:   /** The button reset. */
00051:   private final JButton buttonReset;
00052:
00053:   /**
00054:    * Instantiates a new InputPanel with the input fields, labels and controllers.
00055:    */
00056:   public InputPanel() {
00057:     this.labelA = new JLabel("a: ");
00058:     this.labelA.setHorizontalAlignment(SwingConstants.CENTER);
00059:
00060:     this.labelB = new JLabel("b:");
00061:     this.labelB.setHorizontalAlignment(SwingConstants.CENTER);
00062:
00063:     this.labelC = new JLabel("c");
00064:     this.labelC.setHorizontalAlignment(SwingConstants.CENTER);
00065:
00066:     this.labelCash = new JLabel("cash:");
```

```
00067:     this.labelCash.setHorizontalAlignment(SwingConstants.CENTER);
00068:
00069:     this.aInput = new JFormattedTextField();
00070:     this.aInput.setToolTipText("input for a");
00071:     this.aInput.setText("1.0");
00072:     this.aInput.setHorizontalAlignment(SwingConstants.CENTER);
00073:     this.add(this.labelA);
00074:     this.add(this.aInput);
00075:
00076:
00077:     this.bInput = new JFormattedTextField();
00078:     this.bInput.setToolTipText("input for b");
00079:     this.bInput.setText("1.4");
00080:     this.bInput.setHorizontalAlignment(SwingConstants.CENTER);
00081:     this.add(this.labelB);
00082:     this.add(this.bInput);
00083:
00084:     this.cInput = new JFormattedTextField();
00085:     this.cInput.setToolTipText("input for c");
00086:     this.cInput.setText("2.3");
00087:     this.cInput.setHorizontalAlignment(SwingConstants.CENTER);
00088:     this.add(this.labelC);
00089:     this.add(this.cInput);
00090:
00091:     this.cashInput = new JFormattedTextField();
00092:     this.cashInput.setToolTipText("input for cash");
00093:     this.cashInput.setText("45.2");
00094:     this.cashInput.setHorizontalAlignment(SwingConstants.CENTER);
00095:     this.add(this.labelCash);
00096:     this.add(this.cashInput);
00097:
00098:     this.labelPrint = new JLabel("Print:");
00099:     this.labelPrint.setHorizontalAlignment(SwingConstants.CENTER);
```

```
00100:     this.add(this.labelPrint);
00101:
00102:     this.buttonPrintPump1 = new JButton("Pump1");
00103:     this.add(this.buttonPrintPump1);
00104:
00105:     this.buttonPrintPump2 = new JButton("Pump2");
00106:     this.add(this.buttonPrintPump2);
00107:
00108:     this.buttonReset = new JButton("Reset");
00109:     this.add(this.buttonReset);
00110:   }
00111:
00112:   /**
00113:    * Gets the label A.
00114:    *
00115:    * @return the label A
00116:    */
00117:   public JLabel getLabelA() {
00118:     return this.labelA;
00119:   }
00120:
00121:   /**
00122:    * Gets the label B.
00123:    *
00124:    * @return the label B
00125:    */
00126:   public JLabel getLabelB() {
00127:     return this.labelB;
00128:   }
00129:
00130:   /**
00131:    * Gets the label C.
00132:    *
```

```
00133:    * @return the label C
00134:    */
00135:   public JLabel getLabelC() {
00136:     return this.labelC;
00137:   }
00138:
00139:   /**
00140:    * Gets the label cash.
00141:    *
00142:    * @return the label cash
00143:    */
00144:   public JLabel getLabelCash() {
00145:     return this.labelCash;
00146:   }
00147:
00148:   /**
00149:    * Gets the a input.
00150:    *
00151:    * @return the a input
00152:    */
00153:   public JFormattedTextField getaInput() {
00154:     return this.aInput;
00155:   }
00156:
00157:   /**
00158:    * Gets the b input.
00159:    *
00160:    * @return the b input
00161:    */
00162:   public JFormattedTextField getbInput() {
00163:     return this.bInput;
00164:   }
00165:
```

```
00166:   /**
00167:    * Gets the c input.
00168:    *
00169:    * @return the c input
00170:    */
00171:   public JFormattedTextField getcInput() {
00172:     return this.cInput;
00173:   }
00174:
00175:   /**
00176:    * Gets the cash input.
00177:    *
00178:    * @return the cash input
00179:    */
00180:   public JFormattedTextField getCashInput() {
00181:     return this.cashInput;
00182:   }
00183:
00184:   /**
00185:    * Gets the button print pump 2.
00186:    *
00187:    * @return the button print pump 2
00188:    */
00189:   public JButton getButtonPrintPump2() {
00190:     return this.buttonPrintPump2;
00191:   }
00192:
00193:   /**
00194:    * Gets the button print pump 1.
00195:    *
00196:    * @return the button print pump 1
00197:    */
00198:   public JButton getButtonPrintPump1() {
```

```
00199:     return this.buttonPrintPump1;
00200:   }
00201:
00202:   /**
00203:    * Gets the label print.
00204:    *
00205:    * @return the label print
00206:    */
00207:   public JLabel getLabelPrint() {
00208:     return this.labelPrint;
00209:   }
00210:
00211:   /**
00212:    * Gets the button reset.
00213:    *
00214:    * @return the button reset
00215:    */
00216:   public JButton getButtonReset() {
00217:     return this.buttonReset;
00218:   }
00219:
00220: }
```

```
00001: /**
00002:  * is a package containing the classes mostly related to View design pattern.
00003:  */
00004: package iit.softwaresystarch.project.view;
```