List of classes:

1. **GasPump_1** //providing all gas pump operations for gas pump 1.

void	**activate(double a)** //store gas price and activate pump, initiate same method in MDA-EFSM class without platform specific parameter.
void	**approved()** //approve credit card, initiate same method in MDA-EFSM.
void	**cancel()** //cancel ongoing process, initiate same method in MDA-EFSM.
void	**payCash(double c)** //pay amount of c by cash.
void	**payCredit()** //pay by credit, needs approval, initiate same method in MDA-EFSM.
void	**pumpGallon()** //pump 1 gallon each time, initiate pump() method in MDA-EFSM, if pay by cash, initiate stopPump() in MDA-EFSM when there's not enough balance, initiate printReceipt() in MDA-EFSM after pump stopped.
void	**reject()** //credit card has been rejected.
void	**start()** //show start menu, initiate same method in MDA-EFSM.
void	**startPump()** //show read message to start pump, initiate same method in MDA-EFSM.
void	**stopPump()** //show stop message to stop pump, initiate same method in MDA-EFSM.

2. **GasPump_2** //providing all gas pump operations for gas pump 2.

void	**activate**(double a, double b) //activate with regular and super prices stored in data store.
void	**approved**() //approve credit card, initiate same method in MDA-EFSM.
void	**cancel**() //cancel ongoing process, initiate same method in MDA-EFSM.
void	**payCredit**() //pay by credit, needs approval, initiate same method in MDA-EFSM.
void	**pumpGallon**() //pump 1 gallon each time, initiate pump() method in MDA-EFSM, if pay by cash, initiate stopPump() in MDA-EFSM when there's not enough balance, initiate printReceipt() in MDA-EFSM after pump stopped.
void	**regular**() //select regular gas, initiate same method in MDA-EFSM.
void	**reject**() //credit card has been rejected.
void	**start**() //show start menu, initiate same method in MDA-EFSM.
void	**startPump**() //show read message to start pump, initiate same method in MDA-EFSM.
void	**stopPump**() //show stop message to stop pump, initiate same method in MDA-EFSM.
void	**Super**() //select super gas, initiate same method in MDA-EFSM.

3. **GasPump_3** //providing all gas pump operations for gas pump 3.

void	**activate**(double a, double b) //activate with regular and super prices stored in data store.
void	**cancel**() //cancel ongoing process, initiate same method in MDA-EFSM.
void	**noReceipt**() //finished pump without print receipt, initiate same method in MDA-EFSM.
void	**payCash**(double c) //pay amount of c by cash.
void	**premium**() //select premium gas, initiate same method in MDA-EFSM.
void	**pumpLiter**() //pump 1 liter each time, initiate pump() method in MDA-EFSM, if pay by cash, initiate stopPump() in MDA-EFSM when there's not enough balance, initiate printReceipt() in MDA-EFSM after pump stopped.
void	**receipt**() //finished pump with receipt printed, initiate same method in MDA-EFSM.
void	**regular**() //select regular gas, initiate same method in MDA-EFSM.
void	**start**() //show start menu, initiate same method in MDA-EFSM.
void	**startPump**() //show read message to start pump, initiate same method in MDA-EFSM.
void	**stopPump**() //show stop message to stop pump, initiate same method in MDA-EFSM.

4. **DataStore** //abstract class providing access to subclasses.

5. **DataStore_1** //store data for gas pump 1.

double	**getCash**() //get cash value if pay by cash selected.
int	**getG**() //get number of gallons pumped.
double	**getPrice**() //get selected gas price.
double	**getTemp_A**() //get temporary price for regular gas.
double	**getTemp_cash**() //get temporary cash value if pay by cash selected.
int	**getW**() //get flag value, 1 is pay by credit, 0 is pay by cash.
void	**setCash**(double cash) //set cash value in data store.
void	**setG**(int G) //set number of gallons pumped.
void	**setPrice**(double price) //set selected gas price.
void	**setTemp_A**(double a) //set temporary price for regular gas.
void	**setTemp_cash**(double c) //set temporary cash value if pay by cash selected.
void	**setW**(int w) //set flag value, 1 is pay by credit, 0 is pay by cash.

6. **DataStore_2** //store data for gas pump 2.

double	**getA**() //get price of regular gas.
double	**getB**()

	//get price of super gas.
int	**getG**() //get number of liters pumped.
double	**getPrice**() //get selected gas price.
double	**getTemp_A**() //get temporary price for regular gas.
double	**getTemp_B**() //get temporary price for super gas.
void	**setA**(double a) //set price for regular gas.
void	**setB**(double b) //set price for super gas.
void	**setG**(int G) //set number of gallons pumped.
void	**setPrice**(double price) //set selected gas price.
void	**setTemp_A**(double a) //set temporary price for regular gas.
void	**setTemp_B**(double b) //set temporary price for supergas.

7. **DataStore_3** //store data for gas pump 3.

double	**getA**() //get price of regular gas.
double	**getB**() //get price of super gas.
double	**getCash**() //get cash value if pay by cash selected.

int	**getL**() //get number of liters pumped.
double	**getPrice**() //get selected gas price.
double	**getTemp_A**() //get temporary price for regular gas.
double	**getTemp_B**() //get temporary price for super gas.
double	**getTemp_cash**() //get temporary cash value if pay by cash selected.
void	**setA**(double a) //set price for regular gas.
void	**setB**(double b) //set price for premium gas.
void	**setCash**(double cash) //set cash value in data store.
void	**setL**(int L) //set number of liters pumped.
void	**setPrice**(double price) //set selected gas price.
void	**setTemp_A**(double a) //set temporary price for regular gas.
void	**setTemp_B**(double b) //set temporary price for premium gas.
void	**setTemp_cash**(double temp_cash) //set temporary cash value if pay by cash selected.

8. **MDA_EFSM** //provide events for output processor incorporated with different state classes.

void	**activate**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**approved**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**cancel**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**noReceipt**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**payCash**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**payCredit**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**pump**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**receipt**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**reject**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**selectGas**(int g, **AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**start**(**AbstractFactory** af)

	//instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**startPump**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.
void	**stopPump**(**AbstractFactory** af) //instantiate same events in corresponding state class if current state is correct, then update state flag to that state.

9. **State** //abstract class grouping provide access to subclass representing different state.
10. **Start** //starting point.

void	**activate**(**OP** op, **AbstractFactory** af) //call according actions in OP class.
int	**getID**() //get current state ID flag.

11. **State_0** //1st state.
12.

int	**getID**() //get current state ID flag.
void	**start**(**OP** op, **AbstractFactory** af) //call according actions in OP class.

13. **State_1** //2nd state.
14.

int	**getID**() //get current state ID flag.
void	**payCash**(**OP** op, **AbstractFactory** af) //call according actions in OP class.
void	**payCredit**(**OP** op, **AbstractFactory** af) //call according actions in OP class.

15. **State_2** //3rd state.

void	**approved**(**OP** op, **AbstractFactory** af) //call according actions in OP class.
int	**getID**() //get current state ID flag.
void	**reject**(**OP** op, **AbstractFactory** af) //call according actions in OP class.

16. **State_3** //4th state.

void	**cancel**(**OP** op, **AbstractFactory** af) //call according actions in OP class.
int	**getID**() //get current state ID flag.
void	**selectGas**(int g, **OP** op, **AbstractFactory** af) //call according actions in OP class.

17. **State_4** //5th state.

int	**getID**() //get current state ID flag.
void	**startPump**(**OP** op, **AbstractFactory** af) //call according actions in OP class.

18. **State_5** //6th state.

int	**getID**() //get current state ID flag.
void	**pumpGas**(**OP** op, **AbstractFactory** af) //call according actions in OP class.
void	**stopPump**(**OP** op, **AbstractFactory** af) //call according actions in OP class.

19. **State_6** //7th state.

int	**getID**() //get current state ID flag.
void	**noReceipt**(**OP** op, **AbstractFactory** af) //call according actions in OP class.
void	**receipt**(**OP** op, **AbstractFactory** af) //call according actions in OP class.

20. **OP** // output processor handle events from MDA-EFSM and execute actions accordingly.

void	**cancelMsg**(**AbstractFactory** af) // call according actions in abstract factory.
void	**displayMenu**(**AbstractFactory** af) // call according actions in abstract factory.
void	**gasPumpedMsg**(**AbstractFactory** af) // call according actions in abstract factory.
void	**payMsg**(**AbstractFactory** af) // call according actions in abstract factory.
void	**printReceipt**(**AbstractFactory** af) // call according actions in abstract factory.
void	**pumpGasUnit**(**AbstractFactory** af) // call according actions in abstract factory.
void	**readyMsg**(**AbstractFactory** af) // call according actions in abstract factory.
void	**rejectMsg**(**AbstractFactory** af) // call according actions in abstract factory.
void	**setInitialValues**(**AbstractFactory** af) // call according actions in abstract factory.
void	**setPrice**(int g, **AbstractFactory** af) // call according actions in abstract factory.
void	**setW**(int k, **AbstractFactory** af)

// call according actions in abstract factory.

void **stopMsg**(**AbstractFactory** af)
// call according actions in abstract factory.

void **storeCash**(**AbstractFactory** af)
// call according actions in abstract factory.

void **storeData**(**AbstractFactory** af)
// call according actions in abstract factory.

21. **Abstract_Factory** //abstract classes grouping factory classes and provide access to them.
22. **AbstractFactory_1** //factory class for gas pump 1.

void **cancelMsg**()
// call according action class.

void **displayMenu**()
// call according action class.

void **gaspumpedMsg**()
// call according action class.

void **payMsg**()
// call according action class.

void **printReceipt**()
// call according action class.

void **pumpGasUnit**()
// call according action class.

void **readMsg**()
// call according action class.

void **rejectMsg**()
// call according action class.

void **setInitialValues**()
// call according action class.

void **setW**(int k)

	// call according action class.
void	**stopMsg**() // call according action class.
void	**storeCash**() // call according action class.
void	**storeData**() // call according action class.

23. **AbstractFactory_2** //factory class for gas pump 2.

void	**cancelMsg**() // call according action class.
void	**displayMenu**() // call according action class.
void	**gaspumpedMsg**() // call according action class.
void	**payMsg**() // call according action class.
void	**printReceipt**() // call according action class.
void	**pumpGasUnit**() // call according action class.
void	**rejectMsg**() // call according action class.
void	**setInitialValues**() // call according action class.
void	**setPrice**(int g) // call according action class.
void	**stopMsg**() // call according action class.

void	**storeData**() // call according action class.

24. **AbstractFactory_3** //factory class for gas pump 3.

void	**cancelMsg**() // call according action class.
void	**displayMenu**() // call according action class.
void	**gaspumpedMsg**() // call according action class.
void	**payMsg**() // call according action class.
void	**printReceipt**() // call according action class.
void	**pumpGasUnit**() // call according action class.
void	**setInitialValues**() // call according action class.
void	**setPrice**(int g) // call according action class.
void	**stopMsg**() // call according action class.
void	**storeCash**() // call according action class.
void	**storeData**() // call according action class.

25. **StoreData** //abstract class grouping subclasses and providing access.
26. **StoreData_1**

void	**storeData**(**DataStore** d)

//store gas price from temporary price in data store.

27. **StoreData_2**

void	**storeData**(**DataStore** d) //store two gas prices from temporary price in data store.

28. **PayMsg** //abstract class grouping subclasses and providing access.
29. **PayMsg_1**

void	**display**() //display payment options.

30. **PayMsg_2**

void	**display**() //display payment options.

31. **PayMsg_3**

void	**display**() //display payment options.

32. **StoreCash** //action to store cash in data store.
33. **DisplayMenu** //abstract class grouping subclasses and providing access.
34. **DisplayMenu_1**

void	**display**() //display gas selection.

35. **DisplayMenu_2**

void	**display**() //display gas selection.

36. **DisplayMenu_3**

void	**display**() //display gas selection.

37. **RejectMsg**

void	**display**()

	//display reject message.

38. **SetW**

void	**setW(DataStore d, int k)** //set flag in datastore, 0 is cash payment, 1 is credit payment.

39. **SetPrice** //abstract class grouping subclasses and providing access.
40. **SetPrice_1**

void	**setPrice(DataStore d, int g)** //set price for selected gas.

41. **SetPrice_2**

void	**setPrice(DataStore d, int g)** //set price for selected gas.

42. **ReadyMsg**.

void	**display**() //display ready message.

43. **SetInitialValues** //abstract class grouping subclasses and providing access.
44. **SetInitialValues_1**

void	**setInitialValues(DataStore d)** //set number of gallons to 0.

45. **SetInitialValues_2**

void	**setInitialValues(DataStore d)** //set number of liters to 0.

46. **PumpGasUnit** //abstract class grouping subclasses and providing access.
47. **PumpGasUnit_1**

void	**pumpGasUnit(DataStore d)** //add number of gallons by 1 each time.

48. **PumpGasUnit_2**

void	**pumpGasUnit(DataStore d)** //add number of liters by 1 each time.

49. **GasPumpedMsg** //abstract class grouping subclasses and providing access.
50. **GasPumpedMsg_1**

void	**display**() //display pumped message.

51. **GasPumpedMsg_2**

void	**display**() //display pumped message.

52. **StopMsg** //abstract class grouping subclasses and providing access.
53. **StopMsg_1**

void	**display**() //display stop message.

54. **StopMsg_2**

void	**display**() //display stop message.

55. **PrintReceipt** //abstract class grouping subclasses and providing access.
56. **PrintReceipt_1**

void	**display**() //print receipt.

57. **PrintReceipt_2**

void	**display**() //print receipt.

58. **CancelMsg**

void	**display**() //display cancel message.