Lab 3

Object-Oriented Design, IV1350

Evan Saboo
saboo@kth.se
2015-05-06

Innehåll

1 Introduktion

Syftet med den tredje labb uppgiften var att skapa ett java program från klass diagrammet som vi utförde i andra labben. Syftet var också att lära sig att skriva Unit tester som är kopplad med programmet man utförde.

2 Metod

Jag arbetade med Emil Nordin på programmeringen av inspektionsprogrammet för att vi skulle bli klara snabbare och utföra koden på ett effektivt sätt. Eftersom vi hade två nästan likadana designer kunde vi gå igenom dem och implementera de bästa metoderna i programmet. Med hjälp klass diagrammen fick vi en stor bild på hur programmet skulle se ut och hur metoderna skulle fungera. Vi fokuserade mest på inspektionsprogrammet eftersom vi ville att alla metoder och klasser skulle funka utan att programmet kraschade på grund av t.ex. fel inmatning. I föreläsningar och handledningen fick vi veta att några klasser, som ”Garage” och ”Payment” skulle inte användas eftersom vi alla metoder skulle användas i ”View” klassen och metoderna skulle gå igenom ”Controller” klassen innan den kunde användas i View.

3 Resultat

I resultat finns det bilder på hela programmet med Unit testet.

```
1  package startup;
2
3  import controller.Controller;
4  /**
5   * Startup of the entire program.
6   *
7   */
8  public class StartUp {
9
10     /**
11      * Starts the program.
12      * @param args  This takes nothing.
13      */
14     public static void main(String[] args) {
15         new Controller().controller();
16
17     }
18 }
19
```

Figur 1. En bild på klassen StartUp.

vehicleInspection/src/controller/Controller.java

```java
package view;

import java.util.Scanner;
import controller.Controller;

/**
 * <code>View</code> handles all the inputs and calls on the controller to use them.
 * Also prints to user.
 */
public class View {

    public  String licenseNumber;
    public  String licenseNumberValid;
    public  boolean licensevalidation;
    private  boolean isReady;
    private  String ready;
    private  String doorCommand;
    private  boolean doorCommandCheck;
    private  String doorState;
    private  boolean correctinput = false;
    private  boolean continueprogram = false;
    public  int queuenumber = 0;
    private  String payment;
    public  int payCash;
    public  int totalChange;
    private boolean updateCashRegister;
    private  boolean creditCardValidation;
    public  String creditCardInfo;
    private  String creditCardPin;
    public String [] inspectionPart = new String [7];
    public  String [] inspect = new String [7];
    public  boolean [] inspectionResults = new boolean[inspect.length];
     Scanner s = new Scanner(System.in);

    /**
     * Runs entire program.
     * Prints out to user and takes inputs to then sent inputs to controller.
     */
    public void view()
    {

        while(continueprogram == false)
        {

            while(correctinput == false)
            {
                System.out.println("Start new inspection? (y/n)");

                ready = s.nextLine();

                if(ready.equals("y") || ready.equals("n"))
                    correctinput = true;

                else
                {
                    correctinput = false;
                    System.out.println("Invalid input");
                }
            }

            if(ready.equals("y"))
            {   isReady = true;
                continueprogram = true;}

            else
            {   isReady = false;
                continueprogram = false;
                correctinput = false;
            }

            queuenumber =  new Controller().startNewInspection(isReady);

        }

```

```
75          System.out.println("Current number is " + queuenumber);
76
77          continueprogram = false;
78
79          while(continueprogram == false)
80          {
81              correctinput = false;
82              while(correctinput == false)
83              {
84                  System.out.println ("Press 'o' to open door");
85                  doorCommand = s.nextLine();
86                  if(doorCommand.equals("o") || doorCommand.equals("c"))
87                          correctinput = true;
88                  else
89                  {
90                      correctinput = false;
91                      System.out.println("Invalid input");
92                  }
93              }
94              doorCommandCheck = new Controller().isOpen(doorCommand);
95
96              if(doorCommandCheck == true)
97              {
98                  doorState = "open";
99                  continueprogram = true;
100             }
101             else
102             {
103                 doorState = "closed";
104                 continueprogram = false;
105                 System.out.println("Open the door to start inspection");
106
107             }
108
109             System.out.println ("The door is " + doorState);
110
111         }
112
113         continueprogram = false;
114         while(continueprogram == false)
115         {
116             correctinput = false;
117             while(correctinput == false)
```

```
            {
                System.out.println ("Press 'c' to close door");
                doorCommand = s.nextLine();
                if(doorCommand.equals("o") || doorCommand.equals("c"))
                    correctinput = true;
                else
                {
                    correctinput = false;
                    System.out.println("Invalid input");
                }
            }
            doorCommandCheck = new Controller().isOpen(doorCommand);

            if(doorCommandCheck == true)
            {
                doorState = "open";
                continueprogram = false;
                System.out.println("Close the door to start inspection");
            }
            else
            {
                doorState = "closed";
                continueprogram = true;
            }


            System.out.println ("The door is " + doorState);
        }


        continueprogram = false;
        while(continueprogram == false)
        {
            System.out.println ("Enter vehicle license number:");
            licenseNumber = s.nextLine();
            licensevalidation = new Controller().enterVehicleInfo(licenseNumber);
            if(licensevalidation == true)
            {
                licenseNumberValid = "valid";
                continueprogram = true;
            }
          else
          {
              licenseNumberValid = "invalid";
              continueprogram = false;
          }

          System.out.println ("The licenseNumber is " + licenseNumberValid);
       }

       continueprogram = false;

       while(continueprogram == false)
       {
           System.out.println("Your inspection payment is " + new Controller().cashToPay + " dollars");

           System.out.println ("Do you want pay with cash or credit? (cash/credit)");
           payment = s.nextLine();

           if(payment.equals("cash") || payment.equals("Cash"))
           {

               while(continueprogram == false)
               {

                   System.out.println("How much do you want to pay? (in dollar bills)");

                   continueprogram = false;
                   while(continueprogram == false)
                   {
                   boolean isNumber;
                   do {
                       if (s.hasNextInt())
                       {
                           payCash = s.nextInt();
                           isNumber = true;

                           updateCashRegister = new Controller().canCashRegisterPayChangeCheck(payCash);
                           totalChange = new Controller().payWithCash(payCash);

                           if(updateCashRegister == true)
                           {
                               continueprogram = true;
```

```
                        }
                        else
                        {
                            System.out.println("Our cash register doesn't have enought cash for change, please pay with a
                            continueprogram = false;
                        }
                    }

                    else
                    {
                        System.out.println("Your payment is invalid or our cash register doesn't have enought cash for cha
                        isNumber = false;
                        s.next();

                    }

                }while(!(isNumber));
                }
                s.nextLine();

                if (totalChange >= 0)
                {
                    System.out.println("Payemt Complete\n"
                            + "");
                     new Controller().cashReceipt(licenseNumber, new Controller().cashToPay, payCash, totalChange);
                    continueprogram = true;
                    System.out.println("----------------------------------------");
                    System.out.println("Total amount cash in CashRegister is: " + new Controller().cashLeftInRegister);


                }

                else
                {
                     System.out.println("You have not paid enough money, please try again");
                     continueprogram = false;
                }
            }

        }
```

Figur 2. Bilder på klassen View.

```
241
242             else if(payment.equals("credit") || payment.equals("Credit"))
243             {
244                 while(continueprogram == false)
245                 {
246                  System.out.println("Enter your credit card info:");
247
248
249                  correctinput = false;
250                  while (correctinput == false)
251                  {
252                   creditCardInfo = s.nextLine();
253
254                   if(creditCardInfo.length() == 16)
255                    correctinput = true;
256                   else
257                    System.out.println("Your credit card info  is invalid, try again!");
258                  }
259
260                  System.out.println("Enter your pin code:");
261
262                  correctinput = false;
263                  while (correctinput == false)
264                  {
265                   creditCardPin = s.nextLine();
266
267                   if(creditCardPin.length() == 4)
268                    correctinput = true;
269                   else
270                    System.out.println("Your credit card pin  is invalid, try again!");
271                  }
272
273                  creditCardValidation = new Controller().sendPaymentAuthorizationRequest(creditCardInfo, creditCardPin);
274
275                  if(creditCardValidation == true)
276                  {
277                   payment = "Complete";
278                   continueprogram = true;
279                  new Controller().creditReceipt(creditCardInfo, licenseNumber, new Controller().cashToPay);
280                  }
281                 else
282                 {
283                  payment = "invalid";
284                  correctinput = false;
285                  continueprogram = false;
286                 }
287                  System.out.println("Payment is " + payment
288                    +"\n" );
289                 }
290             }
291             else
292                 System.out.println("Wrong input, try again!");
293         }
294             System.out.println("Inspection will start now..." + "\n");
295
296             new Controller().parts(inspectionPart);
297
298             for(int i = 0; i < 7; i++)
299             {
300             System.out.println("Inspect:" + inspectionPart[i] + " (pass or fail)");
301             inspect[i] = s.nextLine();
302
303             continueprogram = false;
304                 while(continueprogram == false)
305                 {
306                     if(inspect[i].equals("fail") || inspect[i].equals("Fail") || inspect[i].equals("pass") || inspect[i].equals("Pass"))
307                         continueprogram = true;
308
309                     else
310                     {
311                         System.out.println("Wrong input, try again!");
312                         continueprogram = false;
313                         inspect[i] = s.nextLine();
314                     }
315                 }
316         }
317             inspectionResults = new Controller().getInspections(inspect);
318             new Controller().printout(inspectionResults, inspectionPart);
319
320             System.out.println("Thank you for using Emils & Evan's Inspection agency!");
321     }
322 }
323
```

```java
package controller;
import view.View;
/**
 * The <code>Controller</code> class executes the requests from the <code>View</code> class.
 * Calls to the model pass through the <code>Controller</code>
 */

public class Controller {

public int cashToPay = new CashRegister().cashToPay;
public int cashLeftInRegister = new CashRegister().totalAmountCash + cashToPay;

    /**
     * <code>Controller</code> is used by the <code>StartUp</code> class and runs the <code>view</code> method.
     */
    public void controller()
    {
        new View().view();
    }

/**
 * <code>startNewInspection</code> gets the next queue number from the<code>disaplayNextNumber</code> method.
 * @param   Ready makes next number show if true.
 * @return  Returns the queue number.
 */
    public int startNewInspection(boolean ready)
    {
        int nextnumber;
        nextnumber = new Display().displayNextNumber(ready);
        return nextnumber;
    }

/**
 * @param command   Door opening command entered by user.
 * @return          Returns true if door is open false if not.
 */
    public boolean isOpen(String command)
    {
        boolean checkOpen;
        checkOpen = new GarageDoor().isOpen(command);
        return checkOpen;
    }

/**
 * Gives result of the license number validation
 * @param licensenumber The entered vehicle license number.
 * @return              Returns true if the license number is valid false if invalid.
 */
    public boolean enterVehicleInfo(String licensenumber)
    {
        if(new DbHandler().giveLicenseNumber(licensenumber) == true)
            return true;
        else
            return false;
    }

/**
 * Gets change for payment.
 * @param pay   The paid amount.
 * @return      Returns the change amount.
 */
    public int payWithCash(int pay)
    {
        return new CashRegister().payAndReturnChange(pay);
    }

    /**
     * Checks if there is enough cash for change.
     * @param cashPaying    Cash given by customer.
     * @return              Returns true if enough cash for change.
     */
    public boolean canCashRegisterPayChangeCheck(int cashPaying)
    {

        return new CashRegister().canCashRegisterPayChangeCheck(cashPaying);
    }
```

```
85
86      /**
87       * Sends payment validation request.
88       * @param creditCardInfo    Credit card number
89       * @param creditCardPin     Credit card pin code.
90       * @return                  Credit card validation.
91       */
92      public boolean sendPaymentAuthorizationRequest(String creditCardInfo, String creditCardPin)
93      {
94          return new PaymentAuthorizationSystem().sendPaymentAuthorizationRequest(creditCardInfo, creditCardPin);
95      }
96
97      /**
98       * Receipt info for cash payment.
99       * @param licenseNumber The license number.
100      * @param cashToPay     The inspection cost.
101      * @param payCash       The amount paid.
102      * @param totalChange   The change.
103      */
104     public void cashReceipt (String licenseNumber, int cashToPay, int payCash, int totalChange)
105     {
106         new Receipt().cashReceipt(licenseNumber, cashToPay, payCash, totalChange);
107     }
108
109     /**
110      * Receipt info for credit card payment.
111      * @param creditCardInfo    Credit card information.
112      * @param licenseNumber     License number.
113      * @param cashToPay         The inspection cost.
114      */
115     public void creditReceipt(String creditCardInfo, String licenseNumber, int cashToPay)
116     {
117         new Receipt().creditReceipt(creditCardInfo, licenseNumber, cashToPay);
118     }
119
120     /**
121      * Makes inspection results into true/false from pass/fail.
122      * @param inspect   Inspection results by user.
123      * @return          Inspection results boolean.
124      */
125     public boolean [] getInspections(String [] inspect)
126     {
127         boolean[] checkInspection;
128
129         checkInspection = new Inspections().getInspections(inspect);
130         return checkInspection;
131     }
132
133     /**
134      * Gives parts to view.
135      * @param inspectionpart    Part being inspected.
136      */
137     public void parts(String [] inspectionpart)
138     {
139         new Inspections().parts(inspectionpart);
140     }
141
142     /**
143      * Gives information to make printout.
144      * @param inspectionresults     Results from inspection.
145      * @param testnames             Names of inspections.
146      */
147     public void printout(boolean[] inspectionresults, String[] testnames)
148     {
149         Printout.printout(inspectionresults, testnames);
150     }
151
152 }
153
```

Figur 3. Bilder på klassen Controller.

```java
package dbhandler;

/**
 * Class to validate license number
 */
public class DbHandler {
    private  boolean [] valid = new boolean [6];
    private  boolean validate = false;

    /**
     *
     * @param licenseNumber The entered vehicle license number.
     * @return              Returns license number validation, true if valid false if not.
     */
    public boolean giveLicenseNumber(String licenseNumber)
    {

        if (licenseNumber.length() == 6)
        {

            for(int i = 0; i < 3; i++)
            {
                if(Character.isLetter(licenseNumber.charAt(i)))
                    if(Character.isDigit(licenseNumber.charAt(i+3)))
                    valid [i] = true;

                else
                    valid [i] = false;
            }
            if(valid[0] == true && valid[1] == true && valid[2] == true)
                validate = true;

            return validate;
        }
        else
            return false;
    }
}

```

Figur 4. En bild på klassen DbHandler.

```java
package model;

/**
 * Handles cash payment.
 */
public class CashRegister {

    public int cashToPay = 100;
    public int totalAmountCash = 2000;
    public int cashLeftInRegister = 0;
    private boolean booltotalAmountCash;

/**
 * @param payCash   Amount of cash paid.
 * @return          Returns change.
 */
    public int payAndReturnChange(int payCash)
    {
        return payCash - cashToPay;
    }
    /**
     * Checks if cash register can afford to pay change.
     * @param cashPaying    Amount paid.
     * @return              Returns true if register can afford to pay change.
     */
    public boolean canCashRegisterPayChangeCheck(int cashPaying)
    {

        if (totalAmountCash >= cashPaying)
        {

            booltotalAmountCash = true;
        }
        else
            booltotalAmountCash = false;

        return booltotalAmountCash;
    }
}

```

Figur 5. En bild på klassen CashRegister.

```java
package model;

/**
 * Handles queue number.
 */
public class Display {

    /**
     *
     * @param ready     User input if ready for inspection.
     * @return          Returns queue number.
     */
    public int displayNextNumber(boolean ready)
    {
        int currentnumber = 1336;
        if(ready == true)
            currentnumber += 1;
        return currentnumber;
    }
}

```

Figur 6. En bild på klassen Display.

```java
package model;
/**
 * Handles opening and closing of garage door.
 */
public class GarageDoor {

    /**
     * @param command   Open/close command by user.
     * @return          Returns door state.
     */
    public boolean isOpen(String command)
    {

        if(command.equals("o"))
        {
            return true;
        }
        else
        {
        return false;
        }
    }
}
```

Figur 7. En bild på klassen GarageDoor.

Figur 8. En bild på klassen Inspections.

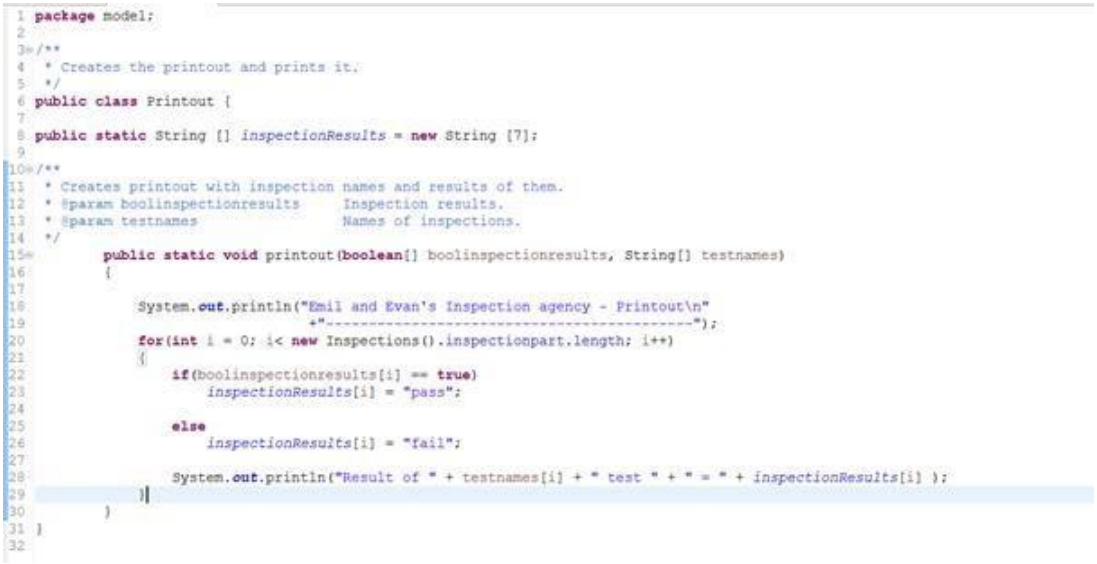
```java
package model;

/**
 * Handles inspection results and names.
 */
public class Inspections {
    public boolean [] boolInspections = new boolean[7];
    public String [] inspectionpart = new String[7];

    /**
     * <code>getInspections</code> turns the inspection results from pass/fail to true
     * @param inspections   Inspection results string
     * @return              Inspection result as boolean
     */
    public boolean [] getInspections (String [] inspections)
    {
        for(int i = 0; i < inspections.length; i++)
        {
            if(inspections[i].equals("pass")|| inspections[i].equals("Pass"))
            {
                boolInspections[i] = true;
            }
            else
                boolInspections[i] = false;
        }
        return boolInspections;
    }
    /**
     * <code>parts</code> contains a list of the parts that should be inspected.
     */
    public  void parts(String [] inspectionpart)
    {

        inspectionpart[0] = "Body";
        inspectionpart[1] = "Glass";
        inspectionpart[2] = "Suspension";
        inspectionpart[3] = "Lights & lenses";
        inspectionpart[4] = "Tires";
        inspectionpart[5] = "Engine";
        inspectionpart[6] = "Brakes";
    }
}
```

```java
package model;

/**
 * Validates the credit card information.
 */
public class PaymentAuthorizationSystem {

    for(boolean b : array) if(!b) return false;
    return true;
  }
}
```

Figur 9. En bild på klassen PaymentAuthorizationSystem.

```java
package model;
import java.util.Random;

/**
 * <code>Receipt</code> creates a receipt and prints it out.
 */
public class Receipt {

    private  Random rand = new Random();
    private  int receiptNumber = rand.nextInt(100);
    private  String creditCardNumberMasked= "XXXX XXXX XXXX ";

    /**
     * Created a receipt for cash payment and prints it out.
     * @param licenseNumber     Vehicle license number.
     * @param cashToPay         Inspection cost.
     * @param cashPayed         Amount paid in cash.
     * @param totalChange       Amount of change.
     */
    public  void cashReceipt (String licenseNumber, int cashToPay, int cashPayed, int totalChange)
    {
        System.out.println(".---.                  _              .-.");
        System.out.println(": .; :               :_;            .' `.");
        System.out.println(":    .' .--.  .--. .-. .--. .---.`. .'");
        System.out.println(": :.`.' '_.''  ..': :' '_.': .; `: :");
        System.out.println(":_;:_;`.__.'`.__.':_;`.__.': ._.':_;");
        System.out.println("                            : :       ");
        System.out.println("                            :_;       ");
        System.out.println("Emil and Evan's Inspection agency - Receipt\n"
                + "--------------------------\n"
                + "Receipt number: " + receiptNumber +"\n"
                + "License number: " + licenseNumber + "\n"
                + "Inpsection cost: " + cashToPay + " dollars\n"
                + "Cash payed: " + cashPayed + " dollars\n"
                + "Change: " + totalChange + " dollars");
    }
    /**
     * Creates a receipt for credit card payment.
     * @param creditCardInfo    Credit card information.
     * @param licenseNumber     Vehicle license number.
     * @param cashToPay         Inspection cost.
     */
    public  void creditReceipt(String creditCardInfo, String licenseNumber, int cashToPay )
    {
        for(int i = 12; i <16; i++)
        {
            creditCardNumberMasked += creditCardInfo.charAt(i);
        }
        System.out.println(".---.                  _              .-.");
        System.out.println(": .; :               :_;            .' `.");
        System.out.println(":    .' .--.  .--. .-. .--. .---.`. .'");
        System.out.println(": :.`.' '_.''  ..': :' '_.': .; `: :");
        System.out.println(":_;:_;`.__.'`.__.':_;`.__.': ._.':_;");
        System.out.println("                            : :       ");
        System.out.println("                            :_;       ");
        System.out.println("Emil and Evan's Inspection agency - Receipt\n"
                + "--------------------------\n"
                + "Receipt number: " + receiptNumber +"\n"
                + "License number: " + licenseNumber + "\n"
                + "Inpsection cost: " + cashToPay + " dollars\n"
                + "Payed with credit card: " + creditCardNumberMasked);
    }
}
```

Figur 10. En bild på klassen Receipt.

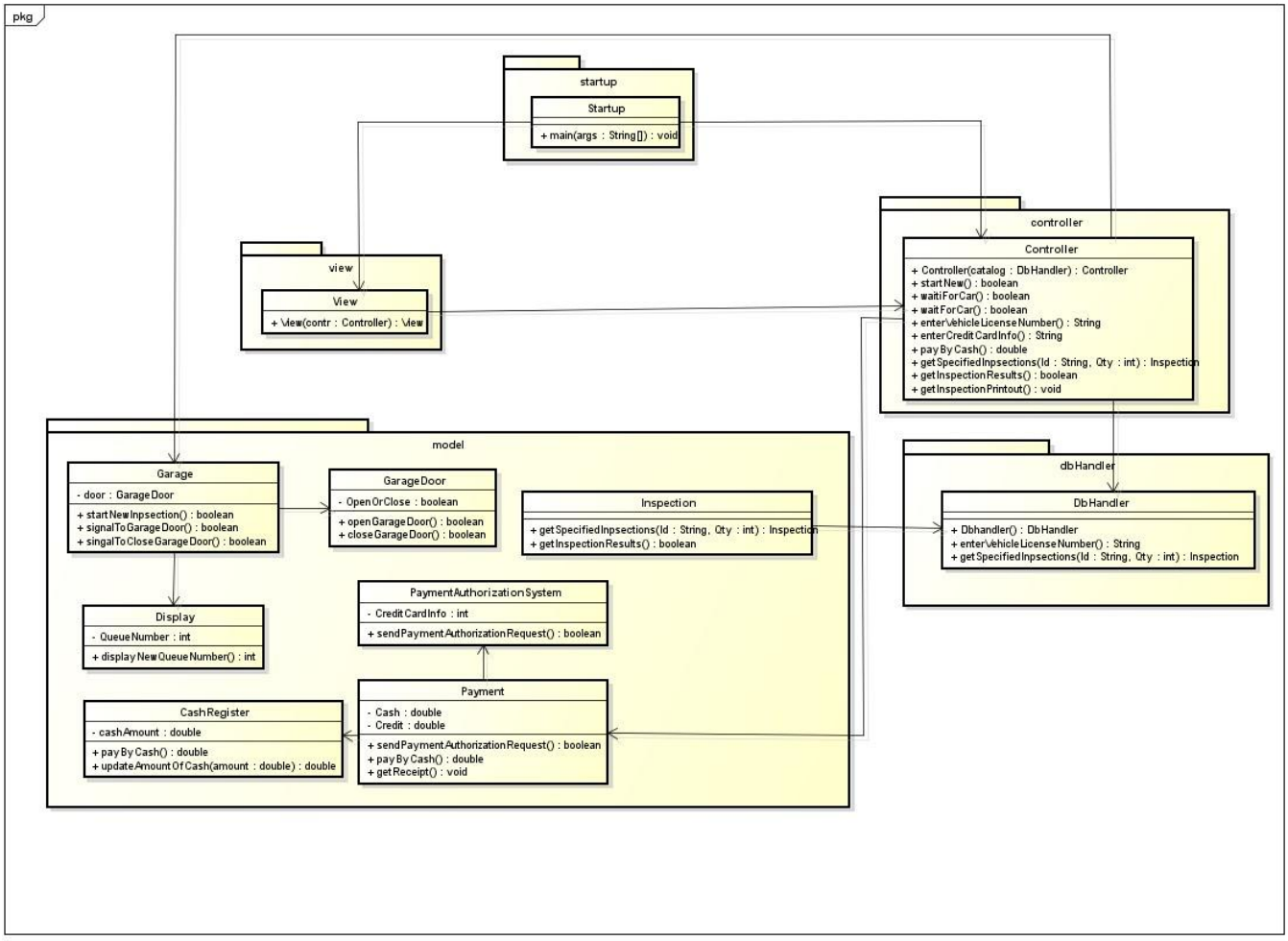


```java
package model;

/**
 * Creates the printout and prints it.
 */
public class Printout {

public static String [] inspectionResults = new String [7];

/**
 * Creates printout with inspection names and results of them.
 * @param boolinspectionresults     Inspection results.
 * @param testnames                 Names of inspections.
 */
      public static void printout(boolean[] boolinspectionresults, String[] testnames)
      {

          System.out.println("Emil and Evan's Inspection agency - Printout\n"
                             +"-----------------------------------------");
          for(int i = 0; i< new Inspections().inspectionpart.length; i++)
          {
              if(boolinspectionresults[i] == true)
                  inspectionResults[i] = "pass";

              else
                  inspectionResults[i] = "fail";

              System.out.println("Result of " + testnames[i] + " test " + " = " + inspectionResults[i] );
          }
      }
}

```

Figur 11. En bild på klassen Printout.

```
78  @Test
79  public void CashRegisterTest() {
80
81      for(int i = 0; i < 10000001; i++)
82      {
83   cashRegisterResults = cashRegister.payAndReturnChange(i);
84   assertEquals((i - cashToPay), cashRegisterResults);
85      }
86
87      for(int i = 0; i <= 2000; i++)
88      {
89          boolcashRegisterResults = cashRegister.canCashRegisterPayChangeCheck(i);
90           assertEquals(true, boolcashRegisterResults);
91      }
92
93      for(int i = 0; i > 2000; i++)
94      {
95          boolcashRegisterResults = cashRegister.canCashRegisterPayChangeCheck(i);
96           assertEquals(false, boolcashRegisterResults);
97      }
98  }
99
100 @Test
101 public void PaymentAuthorizationSystemTest(){
102  String testing = "1";
103  for(int i= 0; i <14; i++)
104  {
105   creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest(testing += "1", "7778");
106   assertEquals(false, creditCardResults);
107  }
108  creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest("1234123412341234", "7778");
109  assertEquals(true, creditCardResults);
110
111  creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest("15642", "7778");
112  assertEquals(false, creditCardResults);
113
114  creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest("1564233534630424", "778");
115  assertEquals(false, creditCardResults);
115  assertEquals(false, creditCardResults);
116
117  creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest("1564233534630", "778");
118  assertEquals(false, creditCardResults);
119
120  creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest("15642335a630", "7a78");
121  assertEquals(false, creditCardResults);
122
123  creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest("15642335a630", "778");
124  assertEquals(false, creditCardResults);
125
126  creditCardResults = paymentAuthorizationSystem.sendPaymentAuthorizationRequest("15642335346304a4", "1a11");
127  assertEquals(false, creditCardResults);
128
129     for(int i = 0; i < 10;i++)
130     {
131         allTrue[i] = true;
132     }
133      creditCardResults = paymentAuthorizationSystem.areAllTrue(allTrue);
134      assertEquals(true, creditCardResults);
135
136         for(int i = 0; i < 10;i++)
137         {
138             if(i < 4)
139             allTrue[i] = true;
140             else
141                 allTrue[i] = false;
142         }
143          creditCardResults = paymentAuthorizationSystem.areAllTrue(allTrue);
144          assertEquals(false, creditCardResults);
145  }
146 }
 73  inspectionResults = inspections.getInspections(inspectionPart);
 74  for(int i= 0; i < 7; i++)
 75   assertEquals(false, inspectionResults[i]);
 76  }
```

Figur 12. Bilder på klassen ModelTest.



Figur 13. En bild på klass diagrammet som vi delvis använde.

```
Start new inspection? (y/n)
y
Current number is 1337
Press 'o' to open door
o
The door is open
Press 'c' to close door
c
The door is closed
Enter vehicle license number:
ijk123
The licenseNumber is valid
Your inspection payment is 100 dollars
Do you want pay with cash or credit? (cash/credit)
credit
Enter your credit card info:
1234123412341234
Enter your pin code:
1234
.---.                 _              .-.
: .; :               :_;            .' `.
:   .' .--.  .--. .-. .--. .---.`. .'
: :.`.' '_.'' ..': :' '_.': .; `: :
:_;:_;`.__.'`.__.':_;`.__.': ._.':_;
                              : :
                              :_;
Emil and Evan's Inspection agency - Receipt
---------------------------
Receipt number: 43
License number: ijk123
Inpsection cost: 100 dollars
Payed with credit card: XXXX XXXX XXXX 1234
Payment is Complete

Inspection will start now...

Inspect:Body (pass or fail)
pass
Inspect:Glass (pass or fail)
pass
Inspect:Suspension (pass or fail)
fail
Inspect:Lights & lenses (pass or fail)
fail
Inspect:Tires (pass or fail)
pass
Inspect:Engine (pass or fail)
pass
Inspect:Brakes (pass or fail)
fail
Emil and Evan's Inspection agency - Printout
-------------------------------------------
Result of Body test  = pass
Result of Glass test  = pass
Result of Suspension test  = fail
Result of Lights & lenses test  = fail
Result of Tires test  = pass
Result of Engine test  = pass
Result of Brakes test  = fail
Thank you for using Emils & Evan's Inspection agency!
```

Figur 14. Bilder på Console printout