UML Basics

By Adam Higherstein

About Sofware Engineering

Software development phases

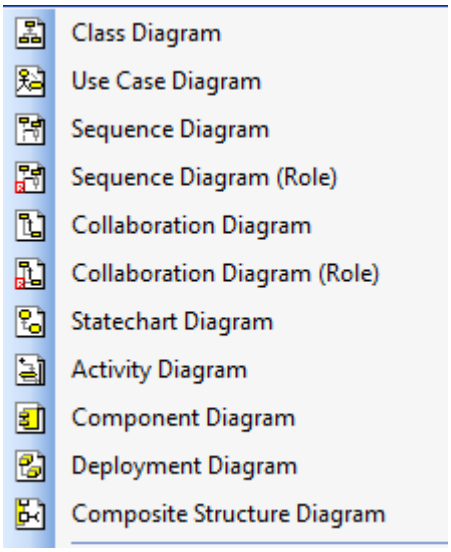
Requirements
- what is to be created, user's view of the product
Analysis
- 1. plan of the product
- what kind of product satisfies customer's needs
- structural and functional model of the application
Design
- how is the application built
- detailed model of the product
- architecture
- programming language
Implementation
- programming
Testing

Every phase is documented!

Modeling gives models that are added to documents.

UML is used as a modeling language.

UML = Unified Modeling Language



In this document we discuss UML diagrams and OOP.

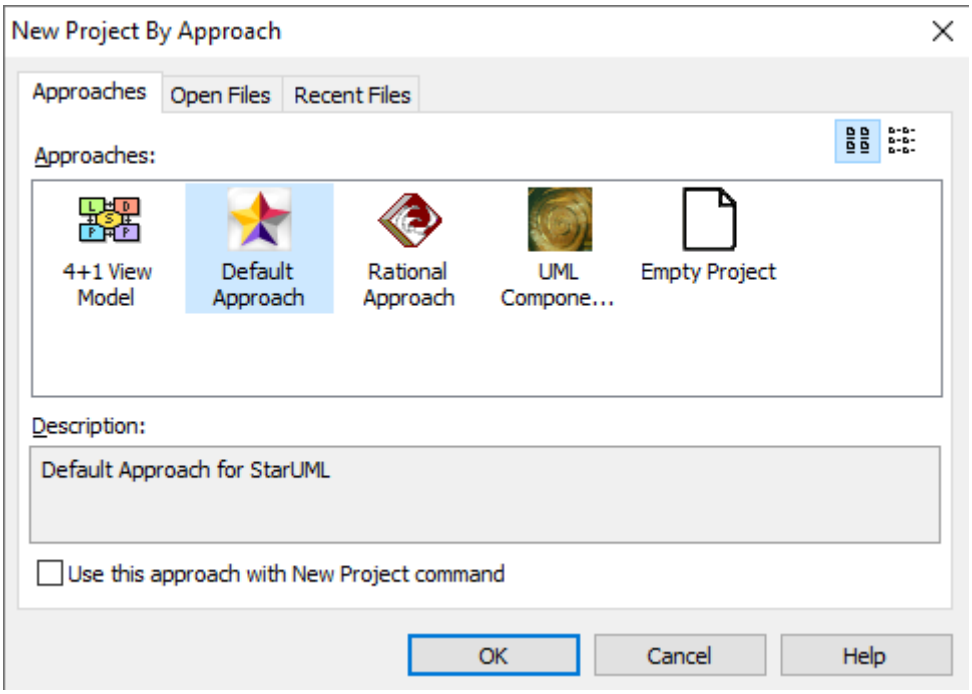
There are several examples and exercises.

In practical examples we use C# - some projects are GUI projects.

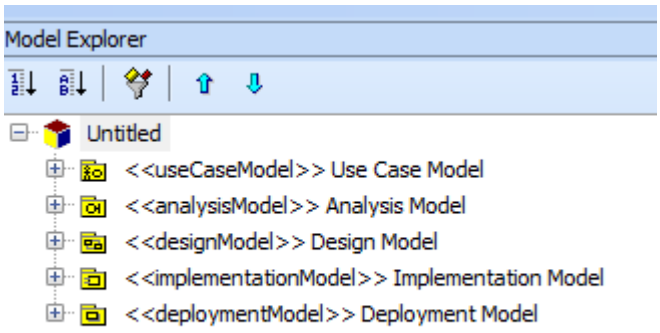
UML tools

Main tool used is WhitestarUML

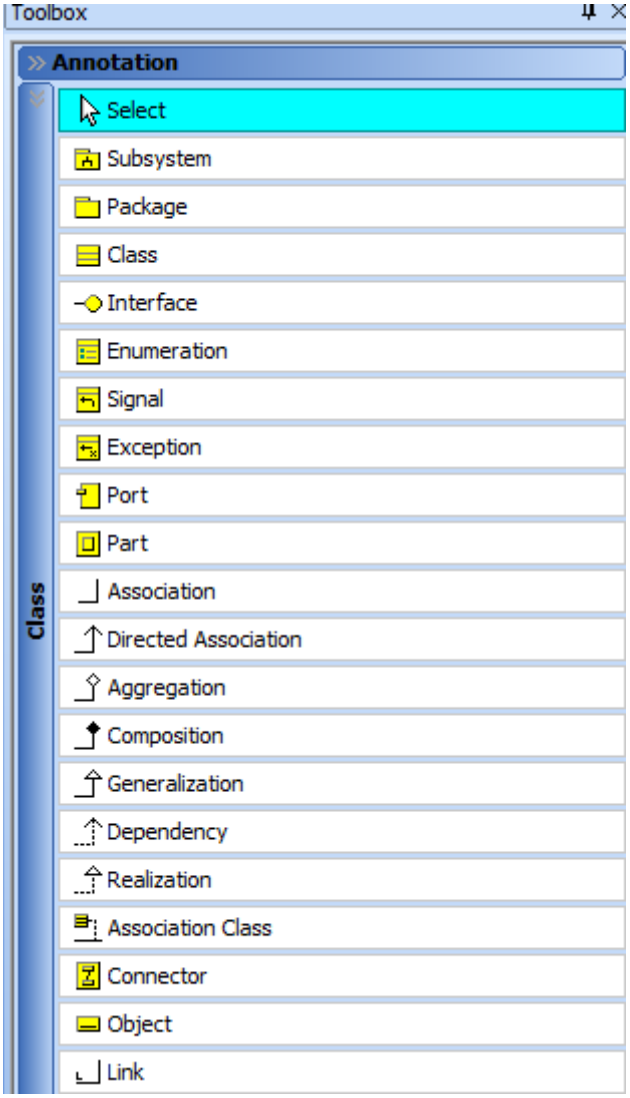
Start a new project



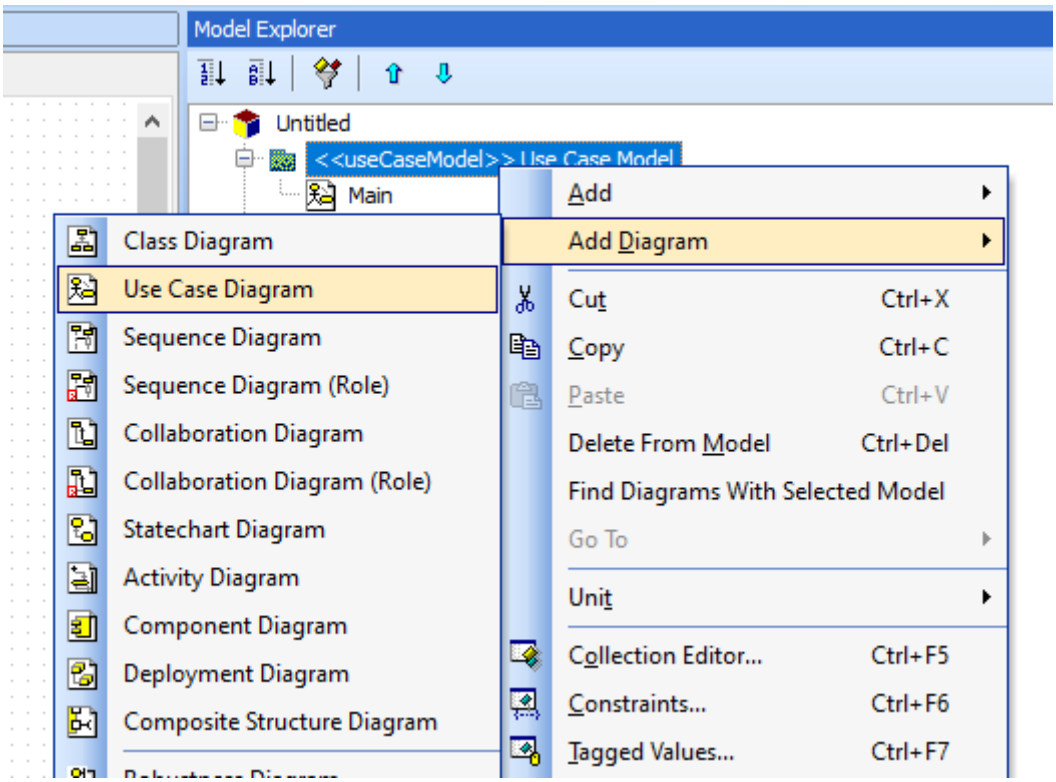
When we start a new project, on the right side of the screen we see model explorer

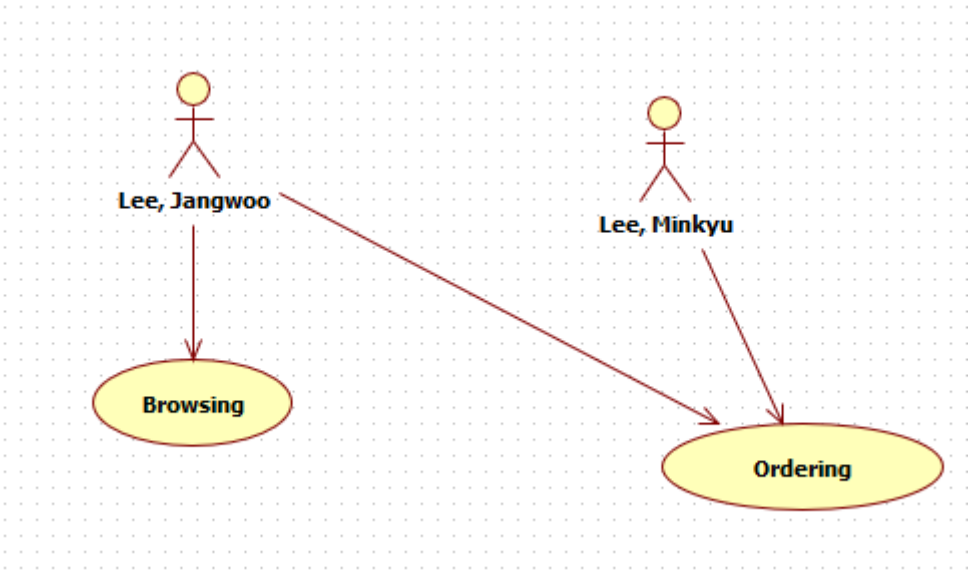


As default are class diagra elements shown on the left:

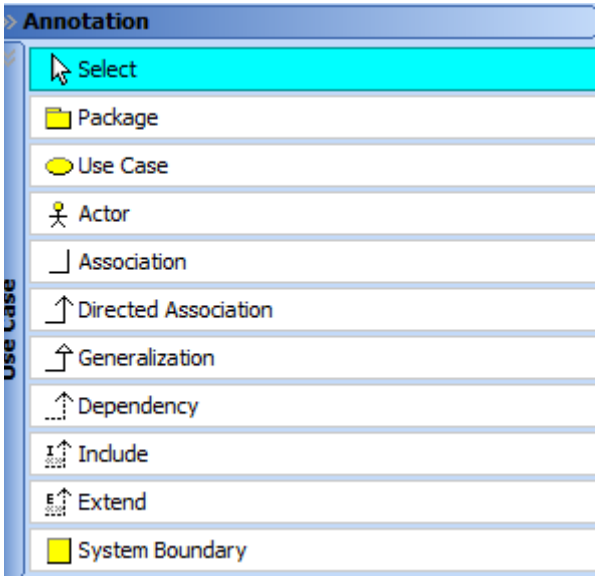


Normally we start a project by requirements phase.
Then we use UML diagram called Use case diagram:



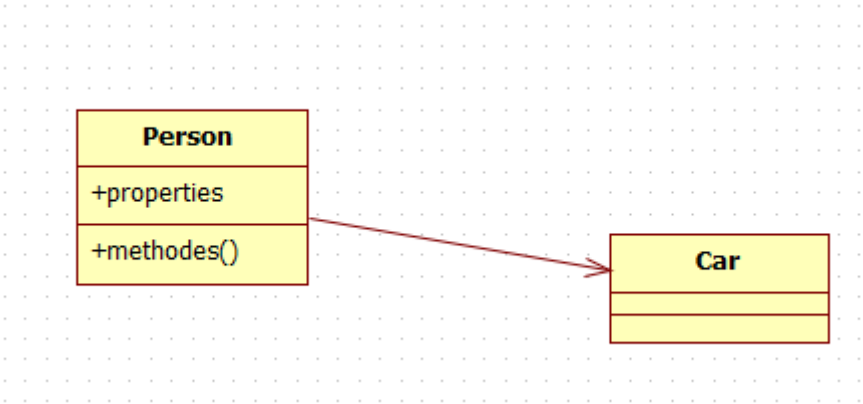


Use case elements.

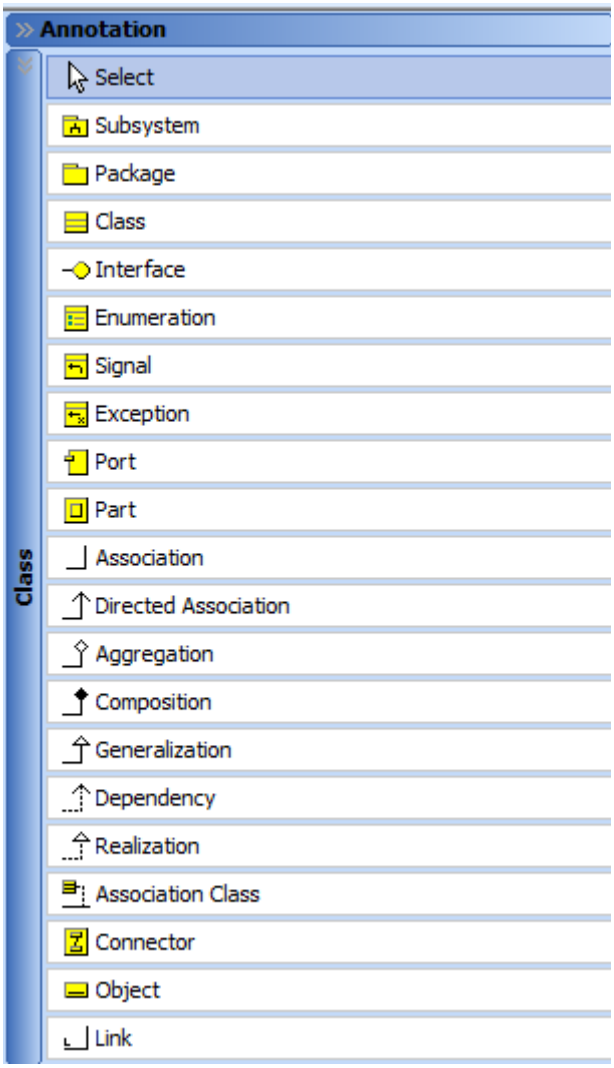


Then are use cases explained using text and thus documented using some defined template.

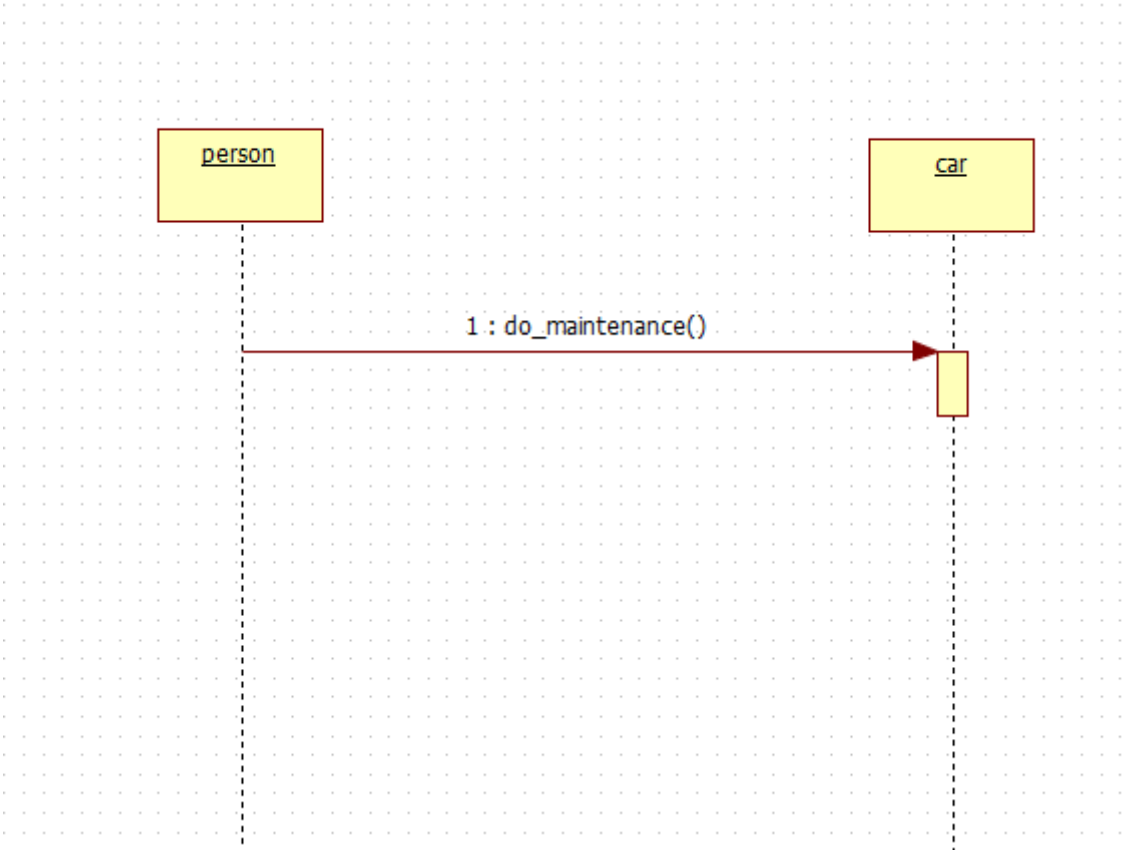
From Use case diagram we can search for class candidates and draw class diagram.



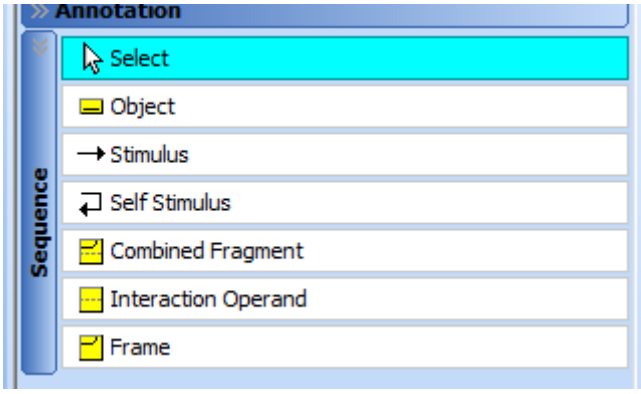
Class diagram elements:



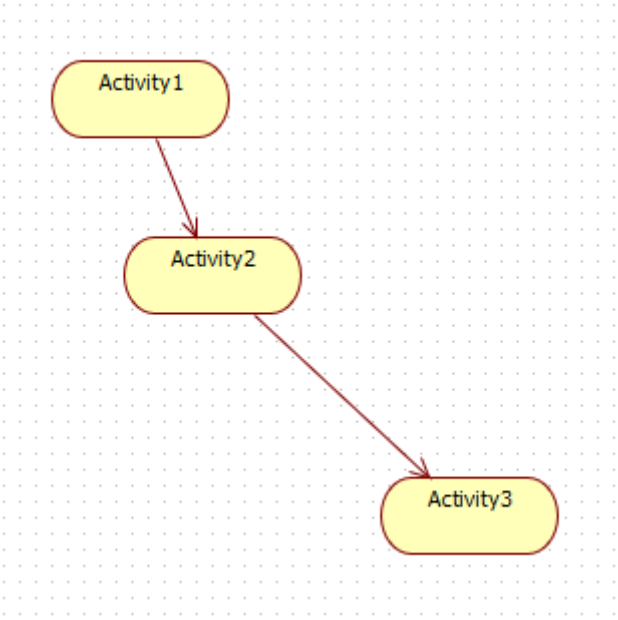
To find out communication between objects we can use sequence diagram.
Comunication is then implemented by adding methods to classes.



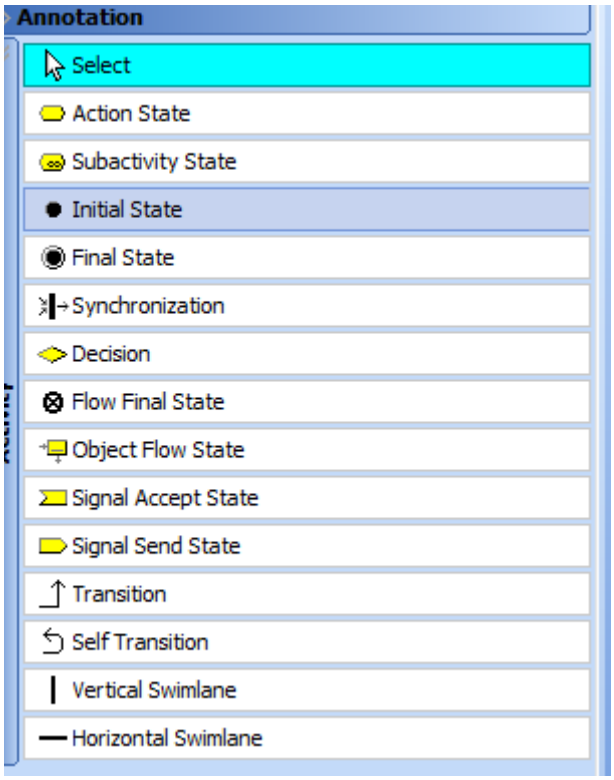
Use case elements



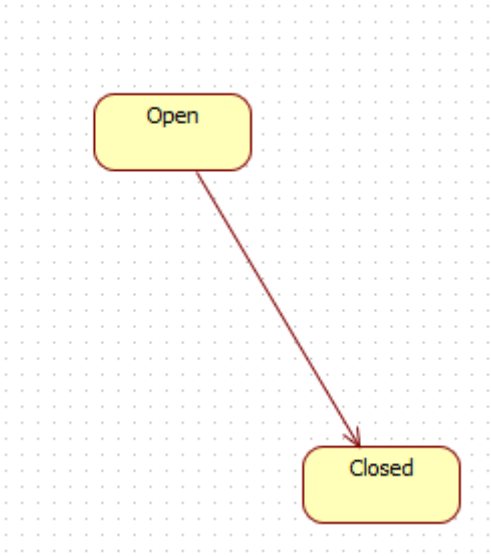
To define implementation of a method, we can use activity diagram.



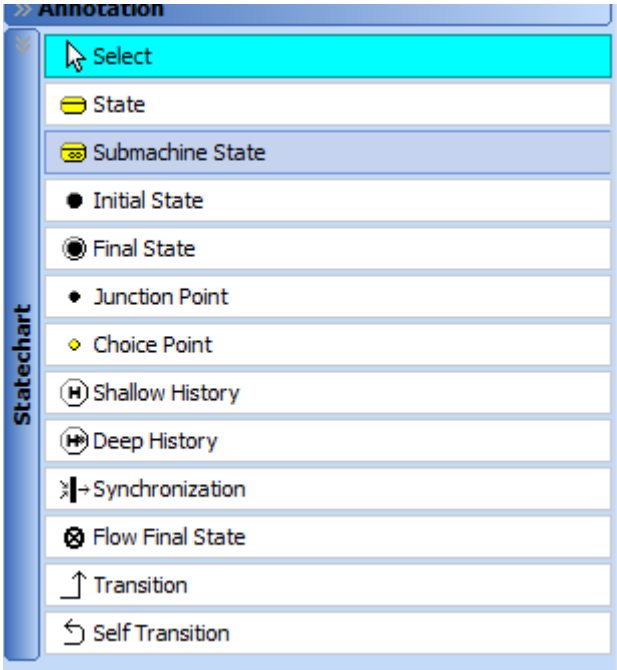
Elements:



With State chart we can make clear different states of some object and thus add or update methods that changes a state.



Elements



There are main diagrams: there are more diagrams that can be used.

This text is from
https://www.researchgate.net/publication/322991896_A_Study_of_Importance_of_UML_diagrams_With_Special_Reference_to_Very_Large-sized_Projects

If you want to decide the importance of all the UML diagrams for small projects then it will be as follows:

Sr .No.	Name of the Diagram	Points
1	Class Diagram	92
2	Sequence Diagram	86
3	Use Case Diagram	84
4	Activity Diagram	78
5	Component Diagram	72
6	Deployment Diagram	70
7	Collaboration Diagram	60

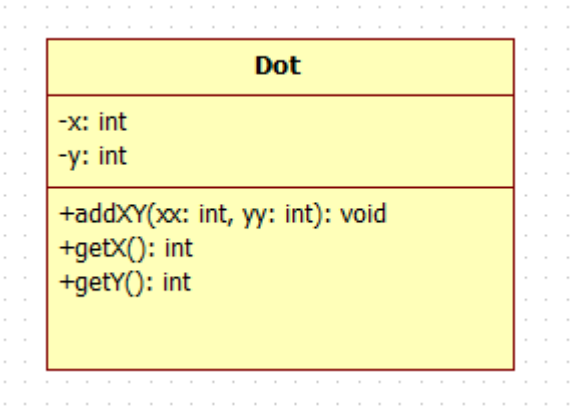
So, good!

There is no need to use or understand all diagrams :)

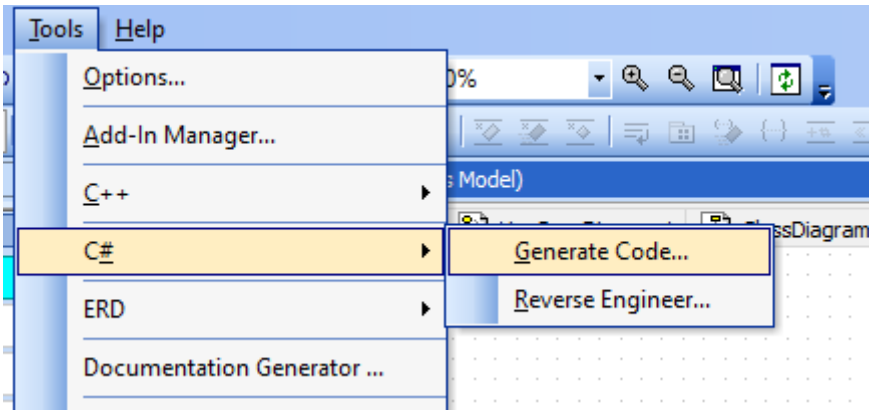
Some UML tools allow to generate code from class diagrams.

Here is an example:

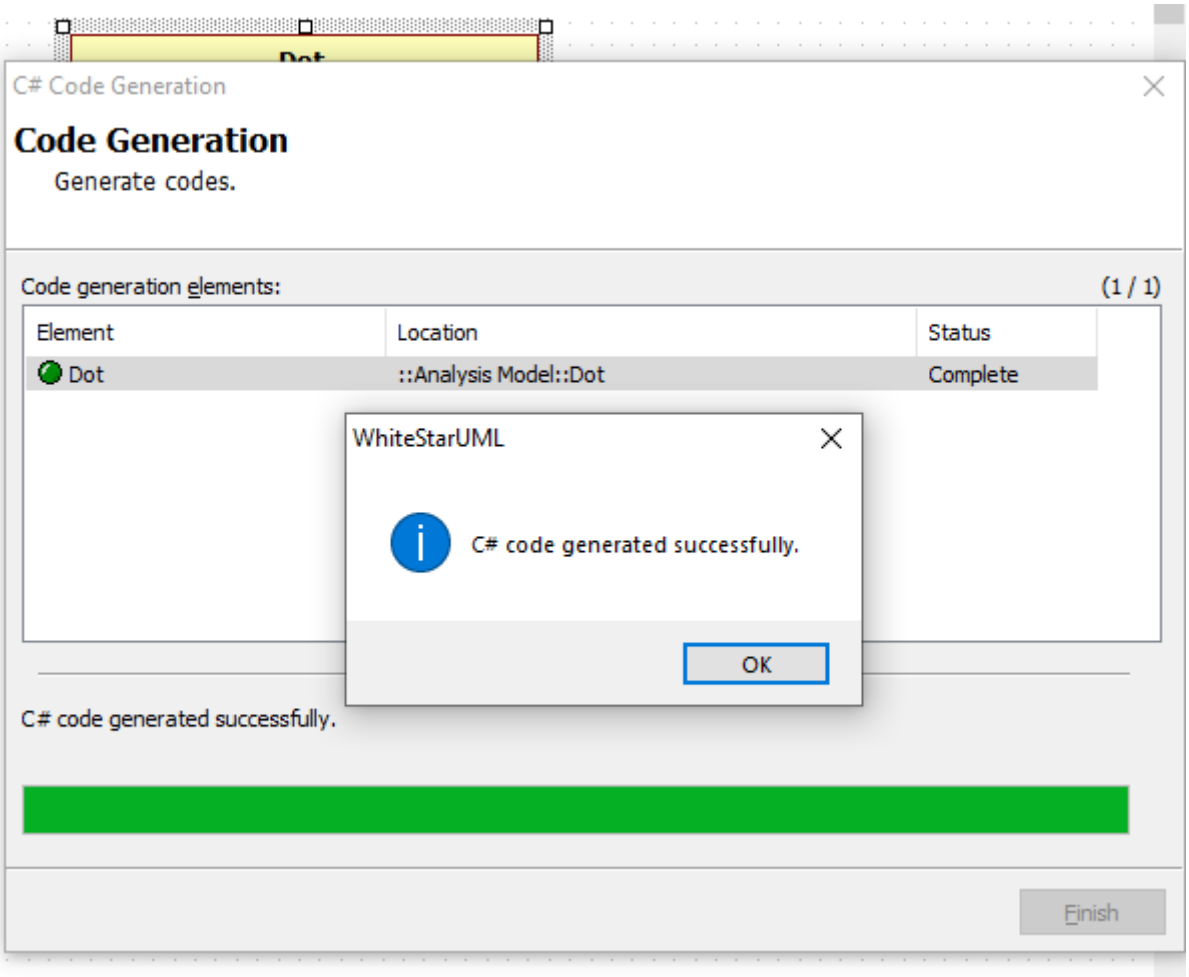
Dot class has been defined here



Now we generate C# code from the class:



And



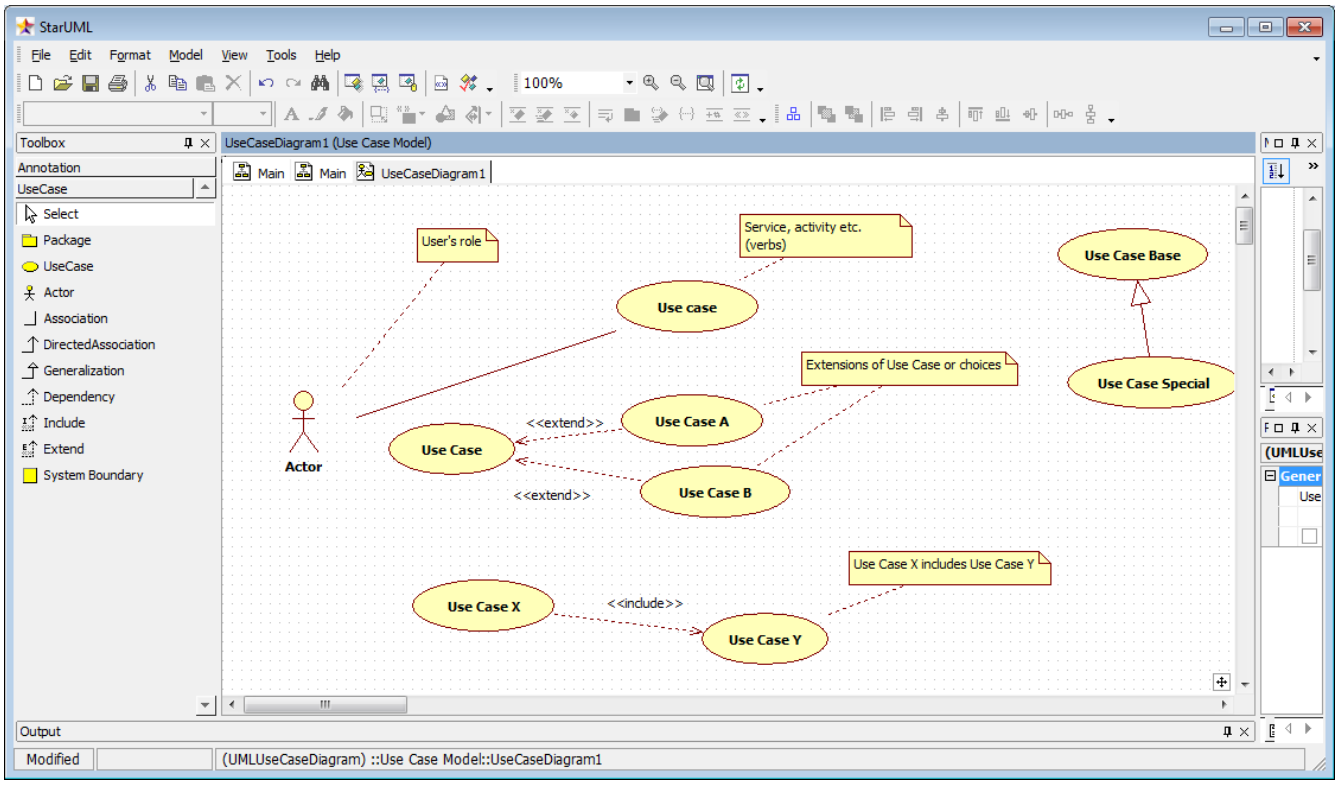
Result is here

```
public class Dot {
    private int x ;
    private int y ;
    public void addXY(int xx, int yy){
    }
    public int getX(){
    }
    public int getY(){
    }
}
```

Requirement definition phase

Use case diagram is used.

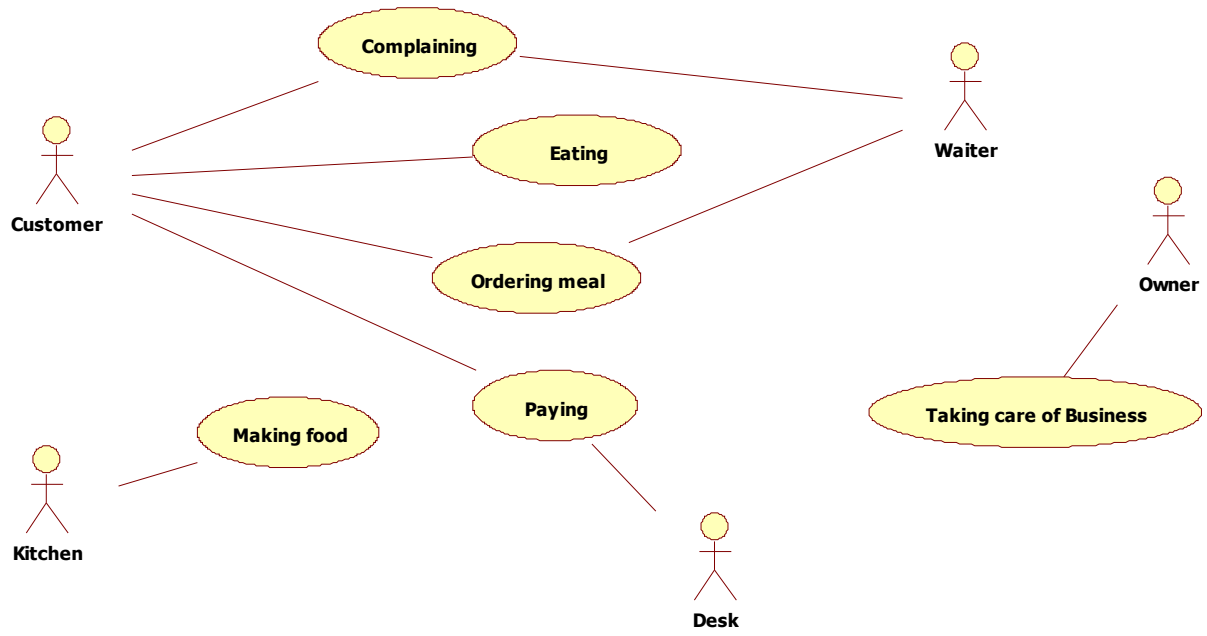
Notation



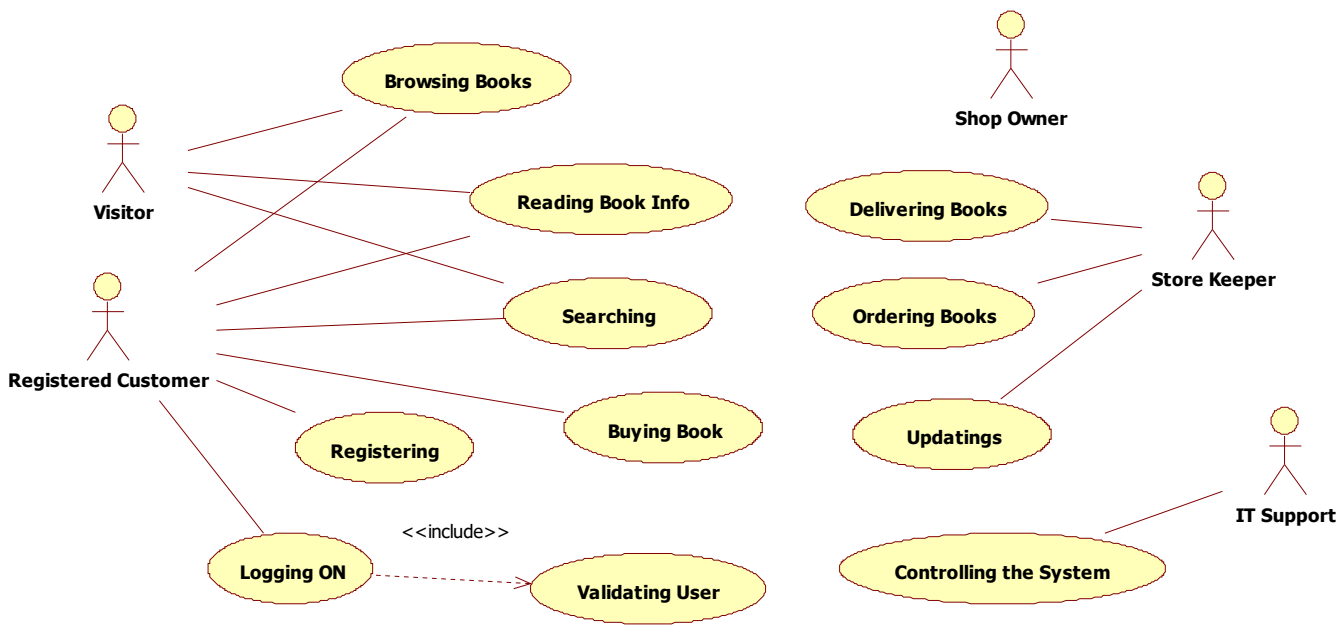
USE CASE EXERCISES

Restaurant

Actors
Customer
Owner
Waiter
Kitchen personnel
Desk personnel



eBook Shop



Use case documentation: "template"

Name of the use case
Actors
Pre-conditions Situation before the use case is started
Description of the use case Describe the use case using clear sentences.
Post-conditions Situation after use case
Exceptions descriptions Describe possible exceptions

Example

Name of the use case: Logging on
Actors: all
Pre-conditions **Home page is open. It shows logging screen that has 2 textboxes and 1 button.** **System is in idle state.**
Description of the use case **User types the username and the password to textboxes and clicks the button.** **System checks if given data is correct. If they are ok, system opens [Exception 1].**
Post-conditions **System is open for use.**
Exceptions descriptions **Exception 1: data is not correct and user is given a message about the situation and logging screen is opened again.**

Searching for a book

Name of the use case **Searching for a book**
Actors **Registered Customer, Visitor**
Pre-conditions **System is open for visitors.**
Description of the use case **User can search for books by different keywords: title, category, author, price, publisher.** **User can choose keyword by using radio buttons or by typing to a textbox.** **User clicks a button and system searches the archive and shows results.** **if nothing is found user is given a message...**
Post-conditions **System is idle...**
Exceptions descriptions **Describe possible exceptions**

Use case exercise

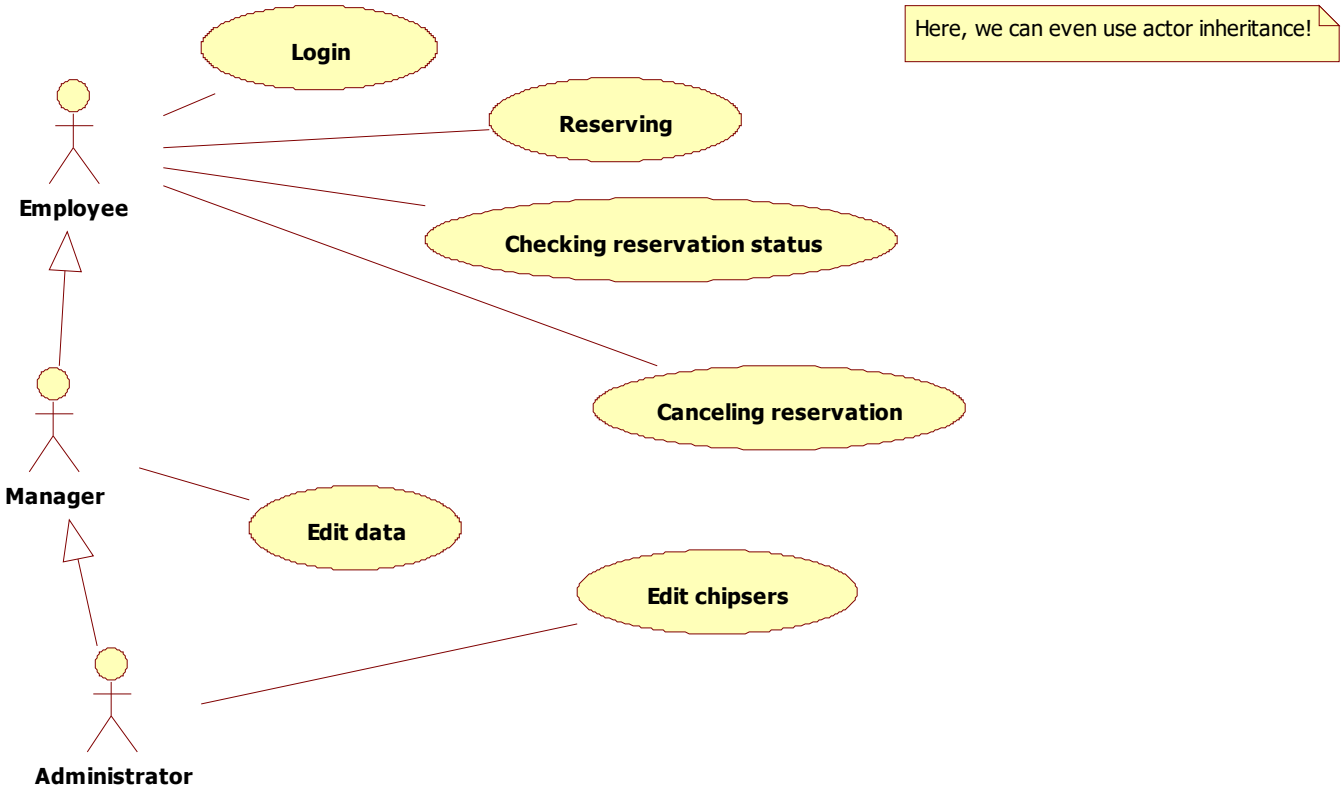
Draw an use case diagram for the following problem:
Company has several holiday homes in their possession and it rents them to employees. Company wants an information system that would allow room reservation on the net.

Employee will be able to reserve a room in desired holiday home for himself and his family members. Besides he can review his reservations and be able to cancel them.

Holiday home manager will be able to do same things as employee - manager will be able to add new reservation (ie. for people that doesn't have access to the net and would like to reserve by phone), review reservations and cancel any reservation. Manager will also be able to edit info about particular holiday home.

Administrator will have the same access as the manager and he is responsible to add and edit ciphers. Everyone must log into the system before they can use it.

One solution



Class diagram

One way to start: read use cases and try to find class candidates (nouns).

Notation

Class Name
+Attributes
+Operations()

Access specifiers (visibility)

+ public
- private
 # protected

Example Horse

Horse
-name: String -age: int
+setData(n: String, a: int): void +getName(): String +getAge(): int

Code with C#

```
class Horse
  {
      private String name;
      private int age;

      public void setData(String n, int a)
      {
          name = n;
          age = a;
      }

      public String getName()
      {
          return name;
      }
      public int getAge()
      {
          return age;
      }

  }
```

```
Horse horse;
        private void button1_Click(object sender, EventArgs e)
        {
            String n = textBox1.Text;
            int a = Convert.ToInt16(textBox2.Text);

            horse = new Horse();
            horse.setData(n, a);
            label3.Text = "horse object created";

        }

        private void button2_Click(object sender, EventArgs e)
        {
            String info = "Horse's name is " + horse.getName();
            info = info + " and its age is " + horse.getAge();

            label4.Text = info;
        }
```

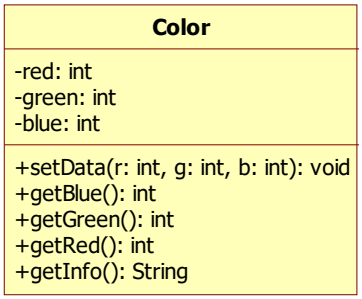
Exercise: color class

Create a color class (with r, g, b attributes, getters and setters)

a) model

b) application

c) show some color in some component




```
OwnColor oc;
        private void button3_Click(object sender, EventArgs e)
        {
            oc = new OwnColor();
            int r, g, b;
            r = Convert.ToInt16(textBox3.Text);
            g = Convert.ToInt16(textBox4.Text);
            b = Convert.ToInt16(textBox5.Text);

            oc.setData(r, g, b);
            label5.Text = "Color created: " + oc.getInfo();

        }

        private void button4_Click(object sender, EventArgs e)
        {
            Color c = new Color();
            c = Color.FromArgb(oc.getRed(),
            oc.getGreen(), oc.getBlue());
            label6.ForeColor = c;
        }
```

```
  class OwnColor
    {
        private int red;
        private int green;
        private int blue;

        public void setData(int r, int g, int b)
        {
            red = r;
            green = g;
            blue = b;
        }

        public int getRed()
        {
            return red;
        }
```

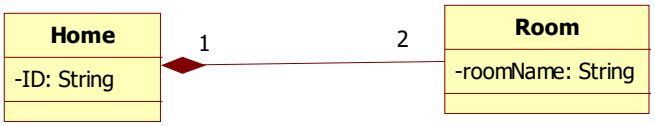
```
        public int getGreen()
        {
            return green;
        }

        public int getBlue()
        {
            return blue;
        }

        public String getInfo()
        {
            String info = "Color's components: red=" + red;
            info = info + ", green=" + green + ", blue=" + blue;
            return info;

        }
    }
```

Relationships between classes

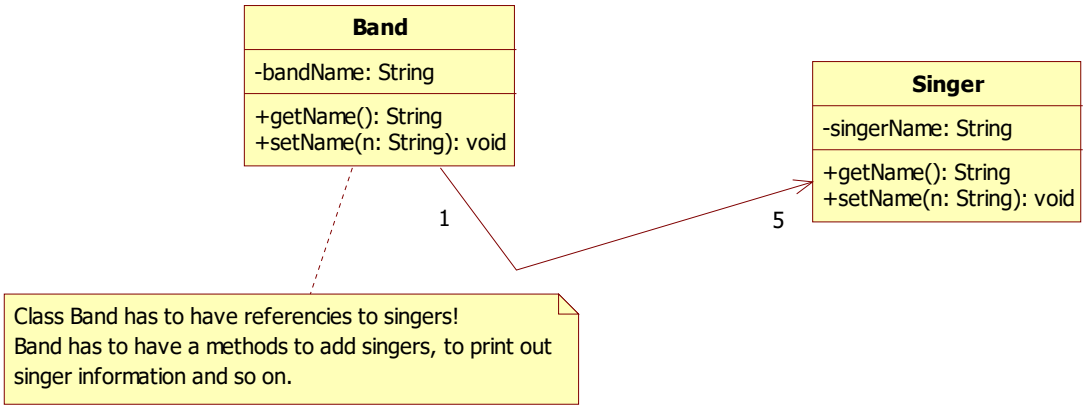
Aggregation

Student and Team
Team has 3 members.

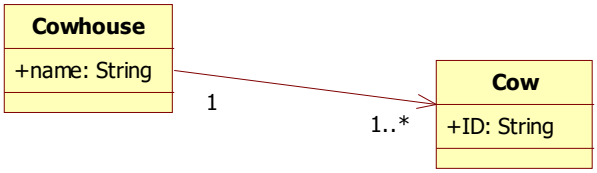
Composition



Association

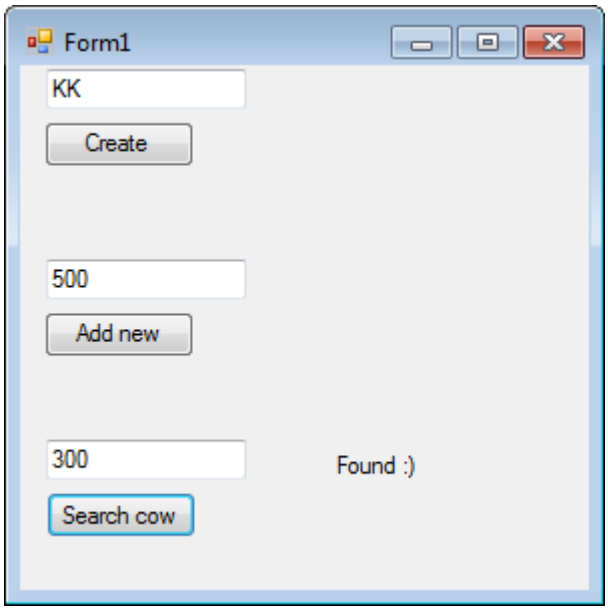


```
Band myBand;
        private void button1_Click(object sender, EventArgs e)
        {
            String name = textBox1.Text;
            myBand = new Band();
            myBand.setName(name);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            String n1, n2, n3, n4, n5;
            n1 = textBox2.Text;
            n2 = textBox3.Text;
            n3 = textBox4.Text;
            n4 = textBox5.Text;
            n5 = textBox6.Text;
            Singer s1 = new Singer();
            Singer s2 = new Singer();
            Singer s3 = new Singer();
            Singer s4 = new Singer();
            Singer s5 = new Singer();

            s1.setName(n1);
            s2.setName(n2);
            s3.setName(n3);
            s4.setName(n4);
            s5.setName(n5);

            myBand.addSingers(s1, s2, s3, s4, s5);

        }

        private void button3_Click(object sender, EventArgs e)
        {
            label1.Text = myBand.getBandInfo();
        }
```




```
class Cow
{
    public String ID;

}



class Cowhouse
    {
        public String name;

        ArrayList cows = new ArrayList ();

        List<Cow> cows2 = new List<Cow>();

        public void addCow(Cow cow)
        {
            cows2.Add(cow);
        }

        public String searchCow(Cow x)
        {
            int size = cows.Count;
            String result = "not found :(";
            for (int k = 0; k < size; k++)
                if (cows[k].Equals(x))
                    result = "Found :)";
            return result;
        }

        public String searchCow(String id)
        {
            int size = cows2.Count;
            String result = "not found :(";
            for (int k = 0; k < size; k++)
                if (cows2[k].ID == id )
                    result = "Found :)";
```

```
            return result;
        }

Cowhouse cowhouse;
        private void button1_Click(object sender, EventArgs e)
        {
            cowhouse = new Cowhouse();
            String n = textBox1.Text;
            cowhouse.name = n;
        }


        private void button2_Click(object sender, EventArgs e)
        {
            Cow newOne = new Cow();
            String n = textBox2.Text;
            newOne.ID = n;

            cowhouse.addCow(newOne);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            String x = textBox3.Text;
            label1.Text = cowhouse.searchCow(x);

        }
```



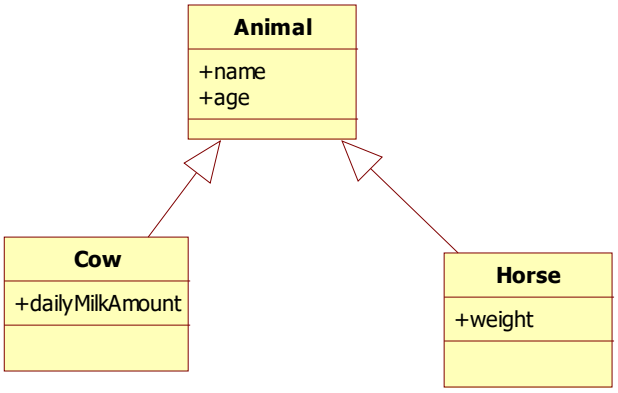
Inheritance

When 2 or more classes have 1 or more same features.

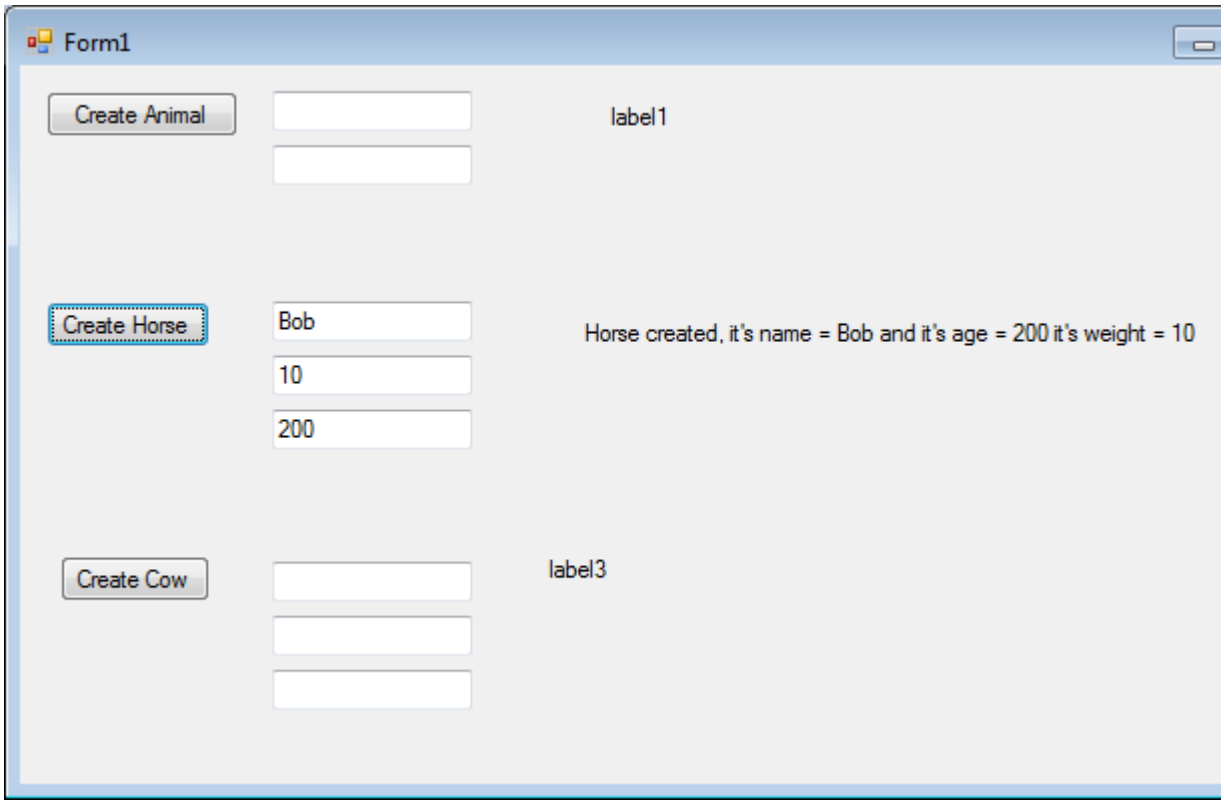
Then, we create a base class containing those shared features and create subclasses that are inherited from base class and can have their own special features.

Animal, cow and horse

class diagram



Program




```
class Animal
{
    protected String name;
    protected int age;

    public String getName()
    {
        return name;
    }

    public int getAge()
    {
        return age;
    }

    public void setData(String n, int a)
    {
        name = n;
        age = a;
    }
```

```
    }
}
```

```
    class Horse: Animal
    {
        private int weight;

        public int getWeight()
        {
            return weight;
        }

        public void setInfo(int w, String n, int a)
        {
            weight = w;
            name = n;
            age = a;
        }

    }
}
```

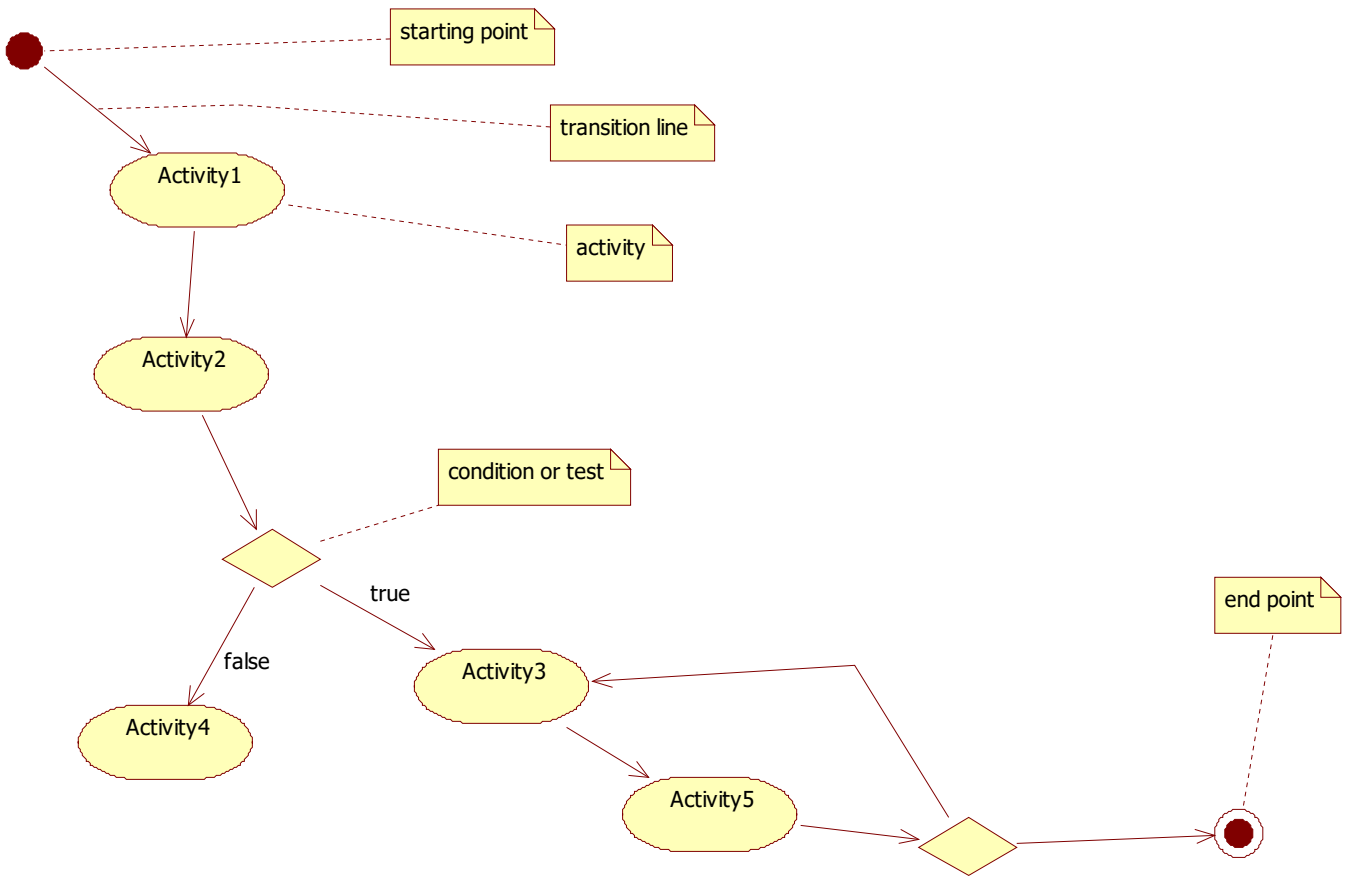
```
    class Cow: Animal  // inheritance!
    {
        private int dailyMilkAmount;

        public int getMilkAmount()
        {
            return dailyMilkAmount;
        }

        public void setInfo(int m, String n, int a)
        {
            dailyMilkAmount = m;
            name = n;
            age = a;
        }

    }
}
```

```
private void button1_Click(object sender, EventArgs e)
        {
            Animal animal1 = new Animal();
            String a = textBox1.Text;  // name
            String b = textBox2.Text;   // age
            int c = Convert.ToInt16(b);  // to int

            animal1.setData(a, c);

            String info = "Animal created, it's name = ";
            info += animal1.getName() + " and it's age = ";
            info += animal1.getAge();

            label1.Text = info;
        }

        private void button2_Click(object sender, EventArgs e)
        {

            Horse horse1 = new Horse();
            String a = textBox3.Text;  // name
            String b = textBox4.Text;   // age
            int c = Convert.ToInt16(b);  // to int

            String d = textBox5.Text;   // weight
            int f = Convert.ToInt16(d);  // to int

            horse1.setInfo(c, a, f);

            String info = "Horse created, it's name = ";
            info += horse1.getName() + " and it's age = ";
            info += horse1.getAge() + " it's weight = ";
            info += horse1.getWeight();

            label2.Text = info;
        }
```

Activity diagram

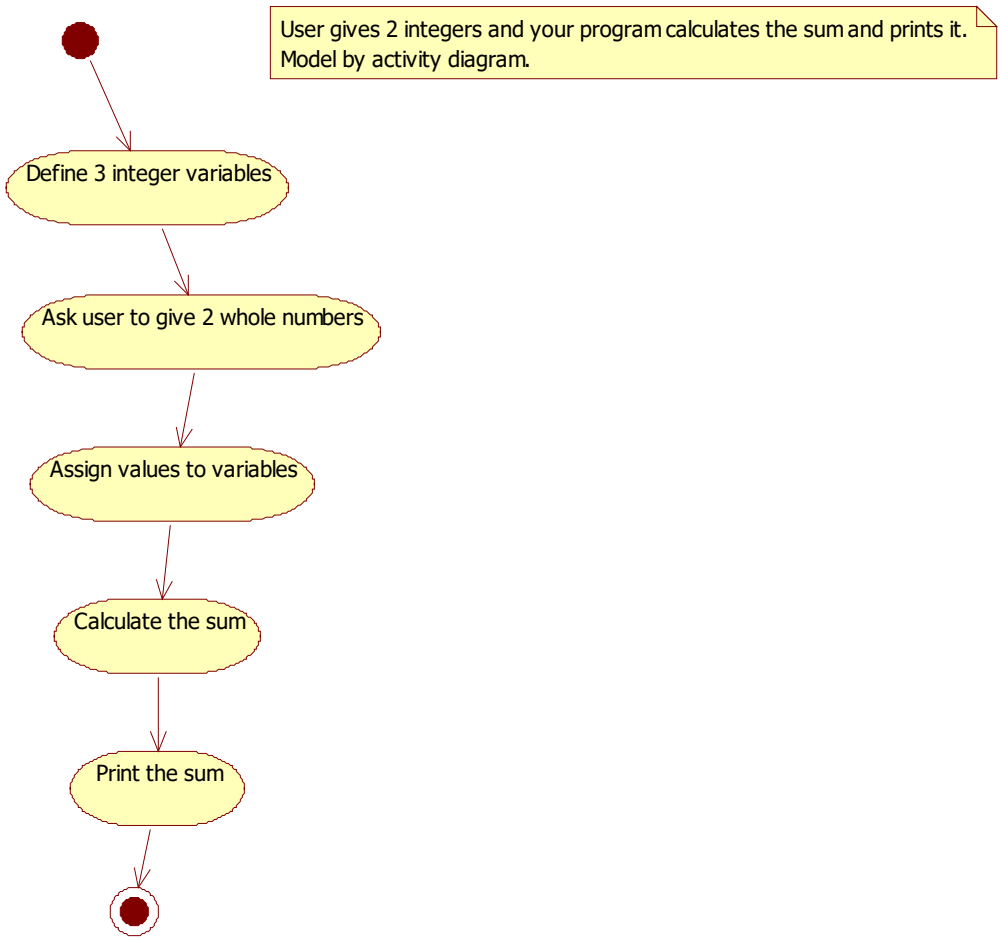
Used to model program (function, algorithm) flow

Elements



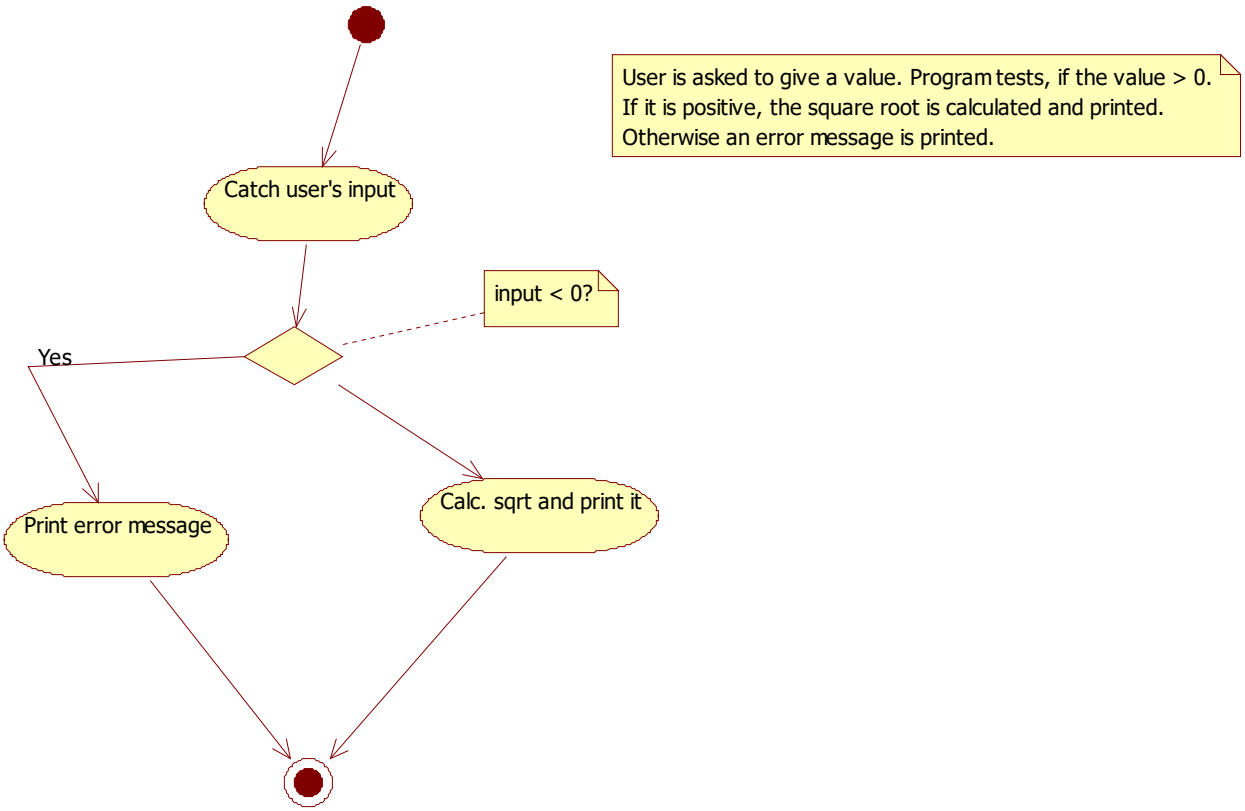
Examples/exercises

Use gives 2 integers and your program calculates the sum and prints it.
Model by activity diagram.

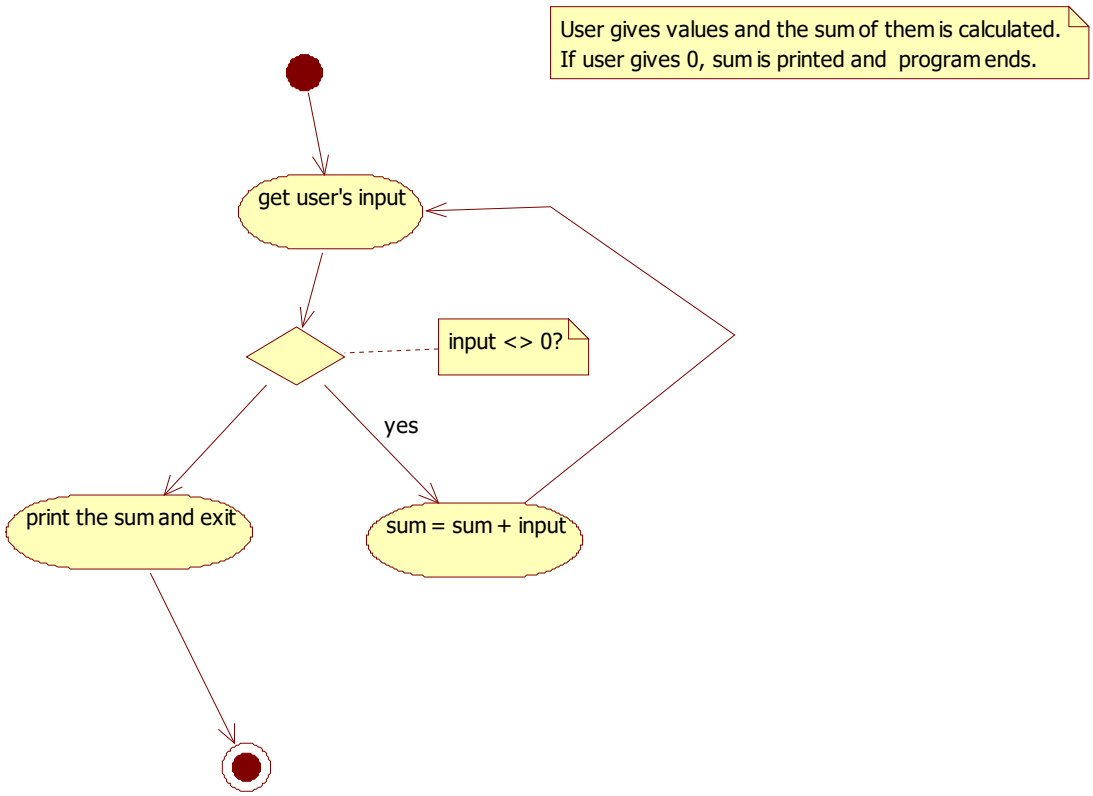


User is asked to give a value. Program tests, if the value > 0.
If it is positive, the square root is calculated and printed.

Otherwise an error message is printed.


User is asked to give a value. Program tests, if the value > 0.
If it is positive, the square root is calculated and printed.
Otherwise an error message is printed.
Catch user's input
input < 0?
Yes
Print error message
Calc. sqrt and print it

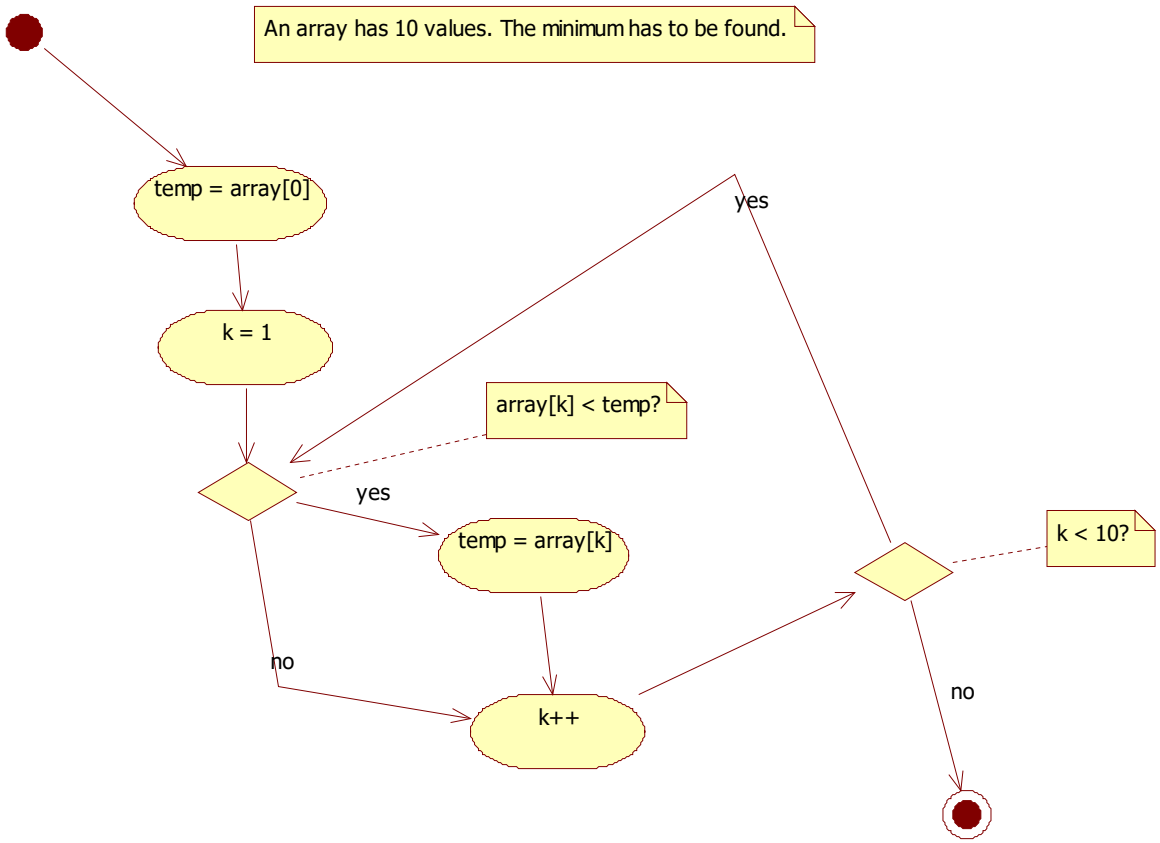
User gives values and the sum of them is calculated.
If user gives 0, program ends.



Exercise

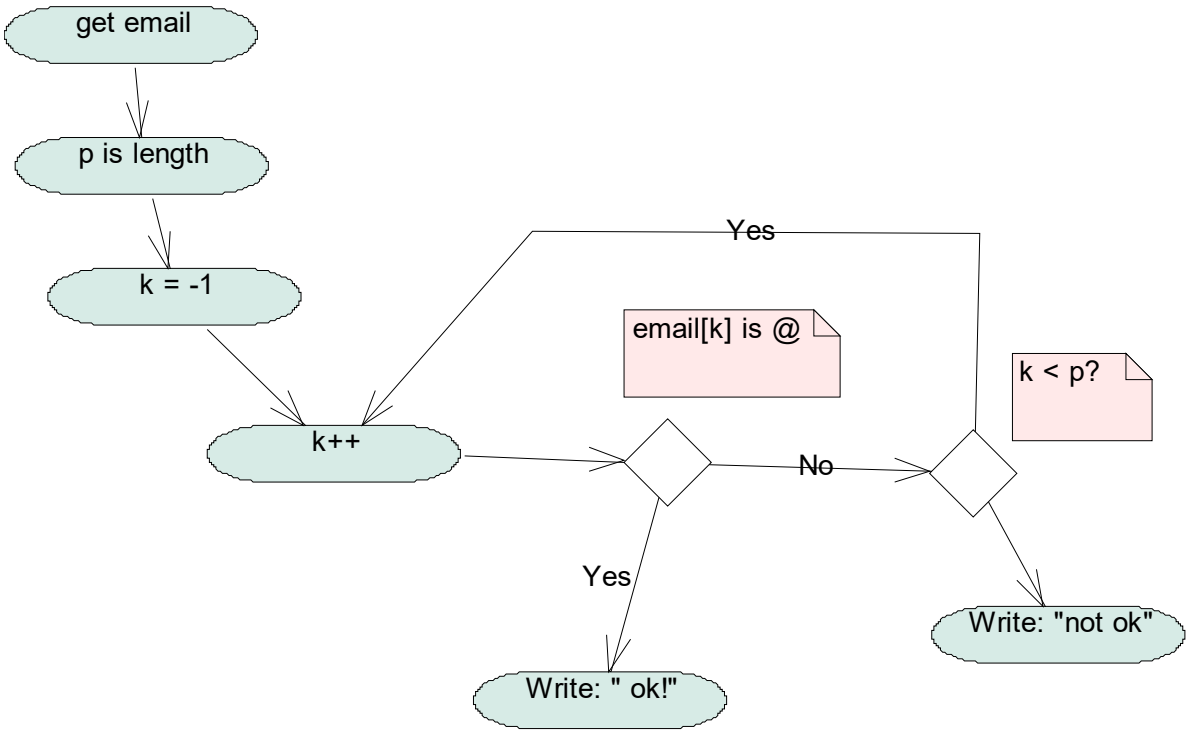
An array has 10 values. The minimum has to be found.

Try to create a program that follows your diagram.



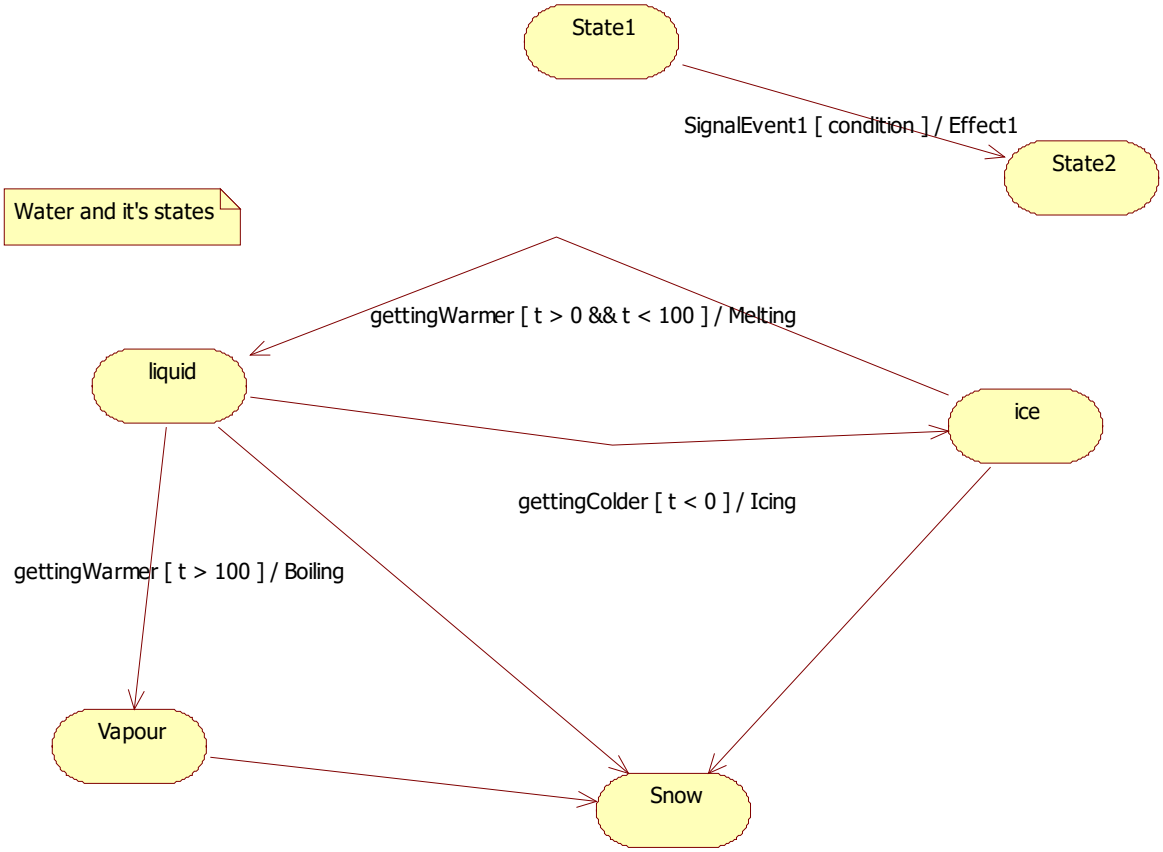
Excercise

Email adress is to be tested like this: does it have sign '@' or not.



State diagrams

To model the behaviour of an object.
Object's state change means that it's attribute value changes.
By using state diagram we can find operations.
What operations change the state?



Sequence diagram

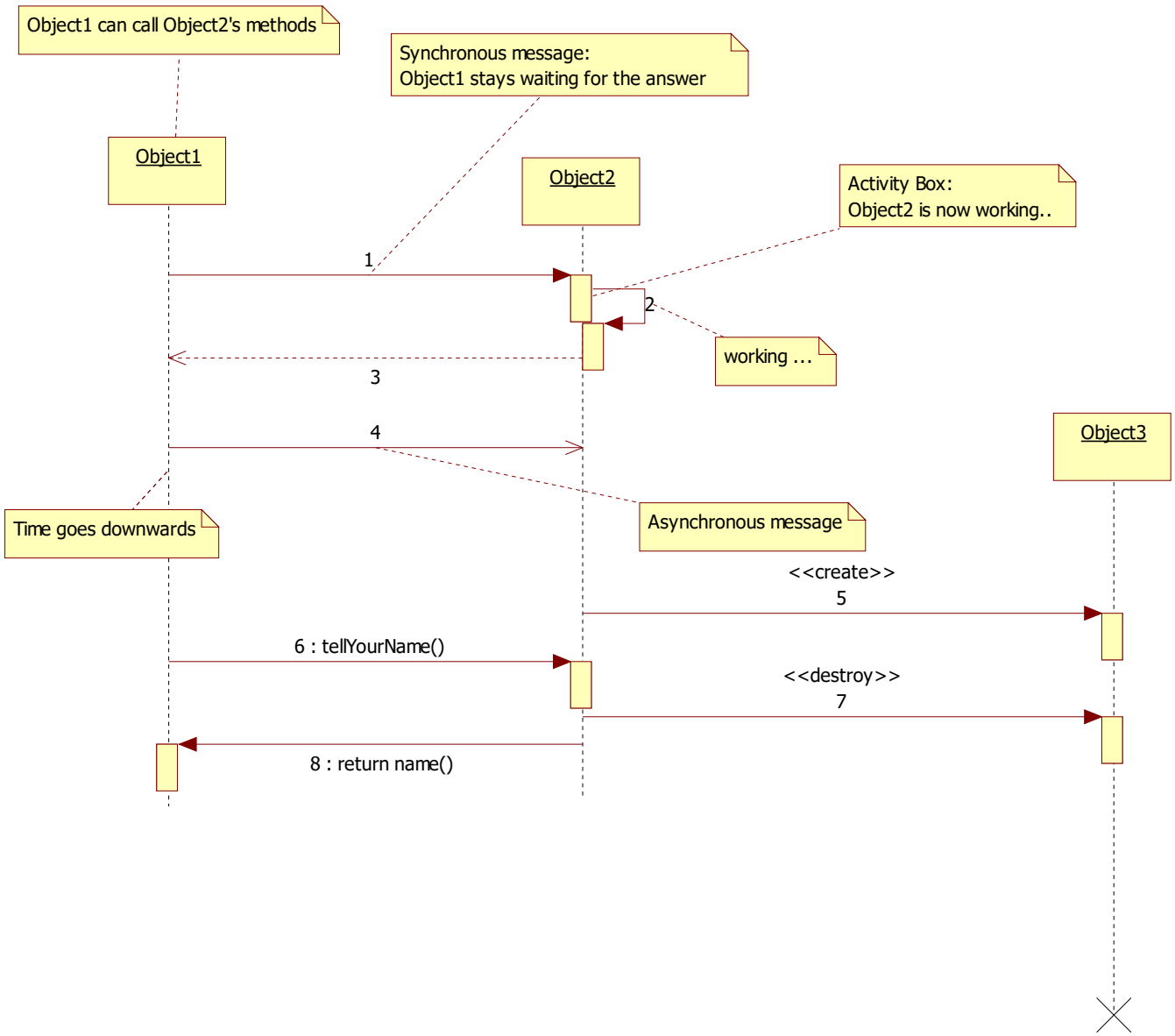
Used to find operations
To find method parameters, their data types and return types.

In analysis phase:
preliminary sequence diagram
In design phase:
final sequence diagram

How objects are communicating with each other?
What messages are sent?
What is returned?
What is the order of messages?

Sequence diagrams are used together with class diagrams iteratively!!

Notation


Object1 can call Object2's methods
Synchronous message:
Object1 stays waiting for the answer
Object1
Object2
Activity Box:
Object2 is now working..
1
2
working ...
3
4
Object3
Time goes downwards
Asynchronous message
<<create>>
5
6 : tellYourName()
<<destroy>>
7
8 : return name()

'

Example

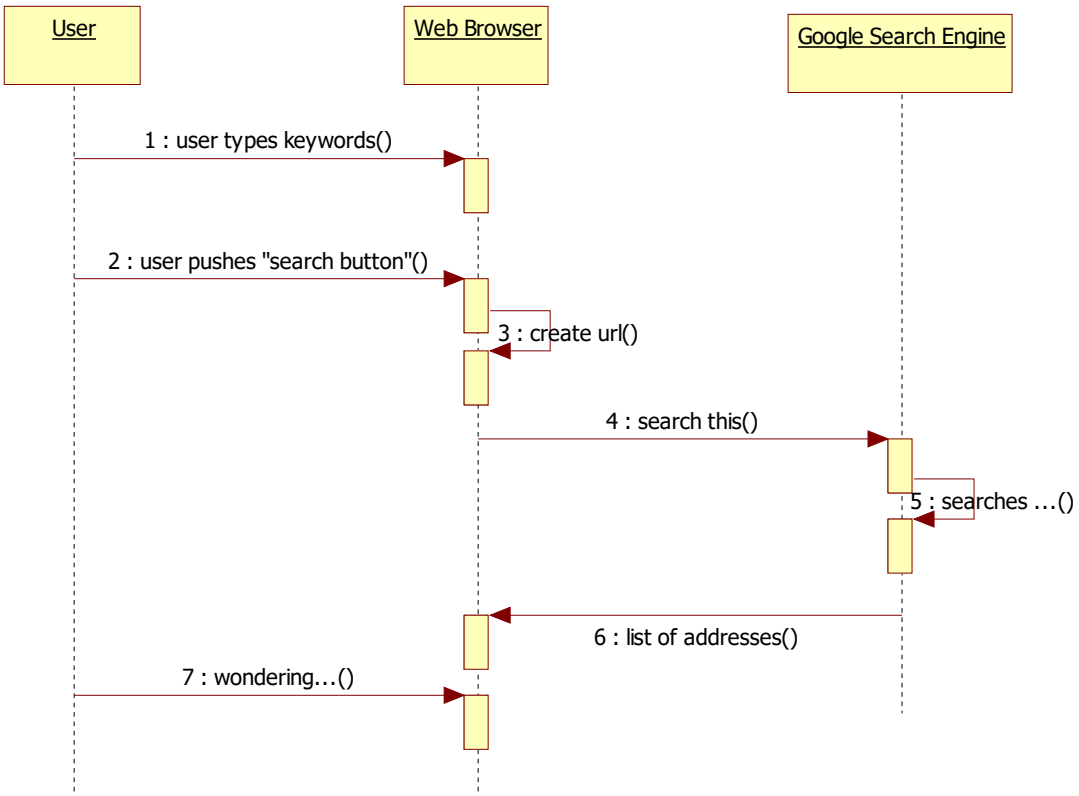
User searches for information via Google.

Objects?

Web Browser

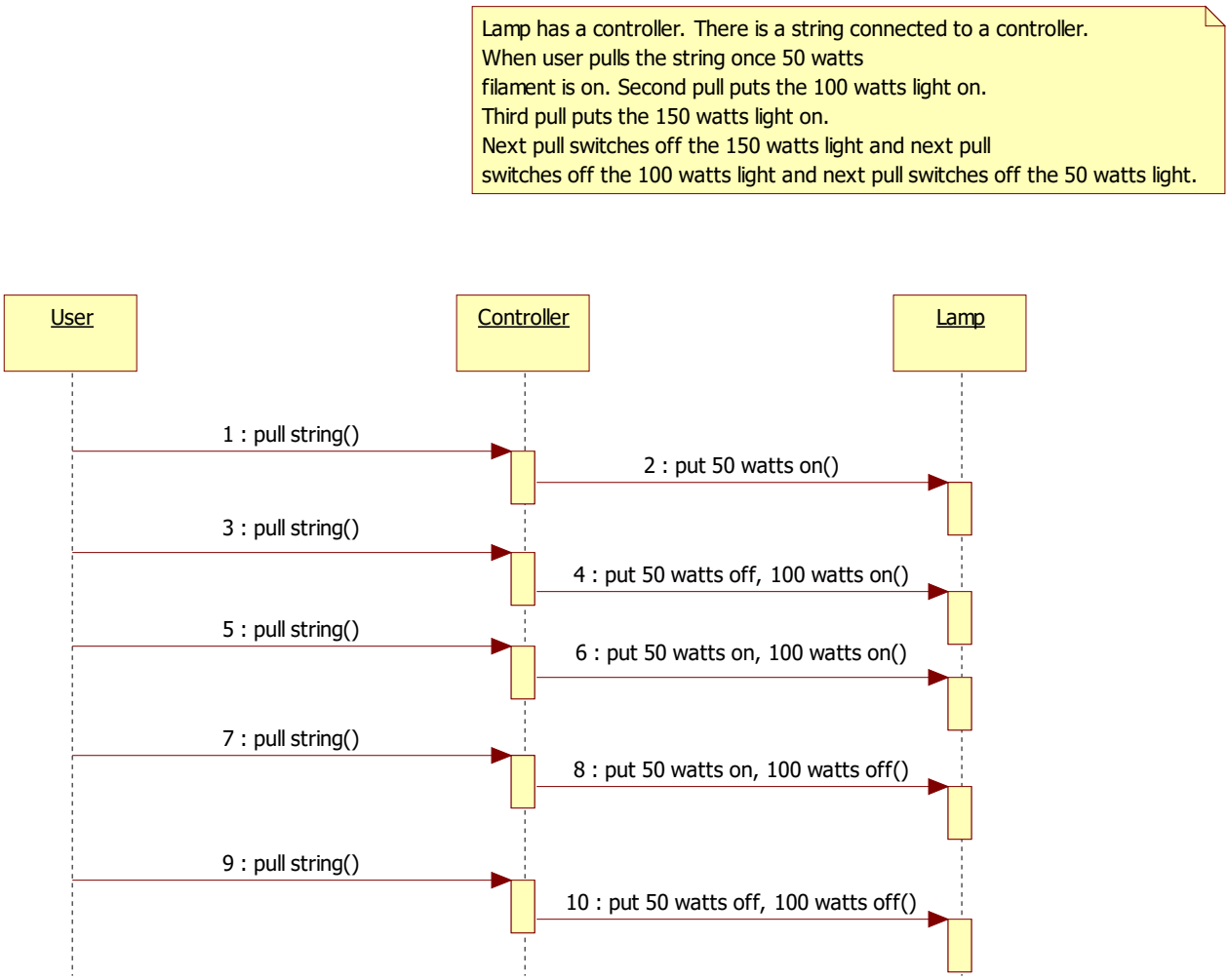
User

Google Search Engine



Exercise

Lamp has a controller. There is a string connected to a controller.
When user pulls the string once 50 watts
filament is on. Second pull puts the 100 watts light on.
Third pull puts the 150 watts light on.
Next pull switches off the 150 watts light and next pull
switches off the 100 watts light and next pull switches off the 50 watts light.



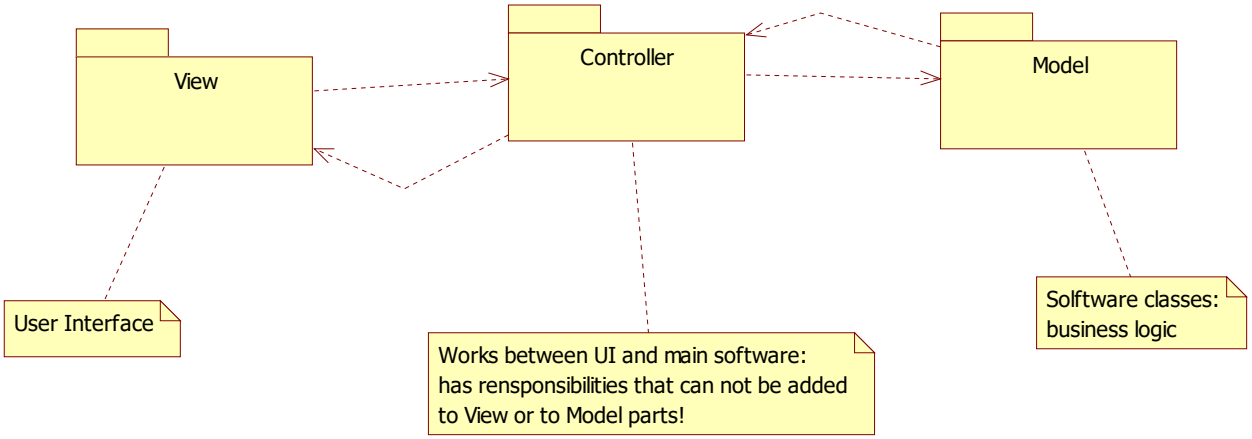
About Architecture

The "big picture" of the system to be developed.
Main parts of the system and their relationships are modelled.
Common architectures:
<u>Layered architecture</u>
examples:
ISO OSI model
most operating systems

Lower layers give services to upper layers.

<u>MVC</u>
Commonly used as a software architecture

MVC = Model View Controller



Example
User gives three integers by using gui. Controller checks if they can be converted to real int values. If not, user is given a message.
Class Analyzer calculates the sum of those values and returns it.

use case

sequence diagram

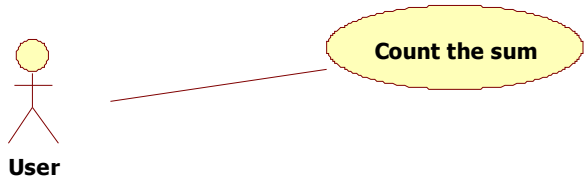
class diagram

gui prototype

final program

****

use case



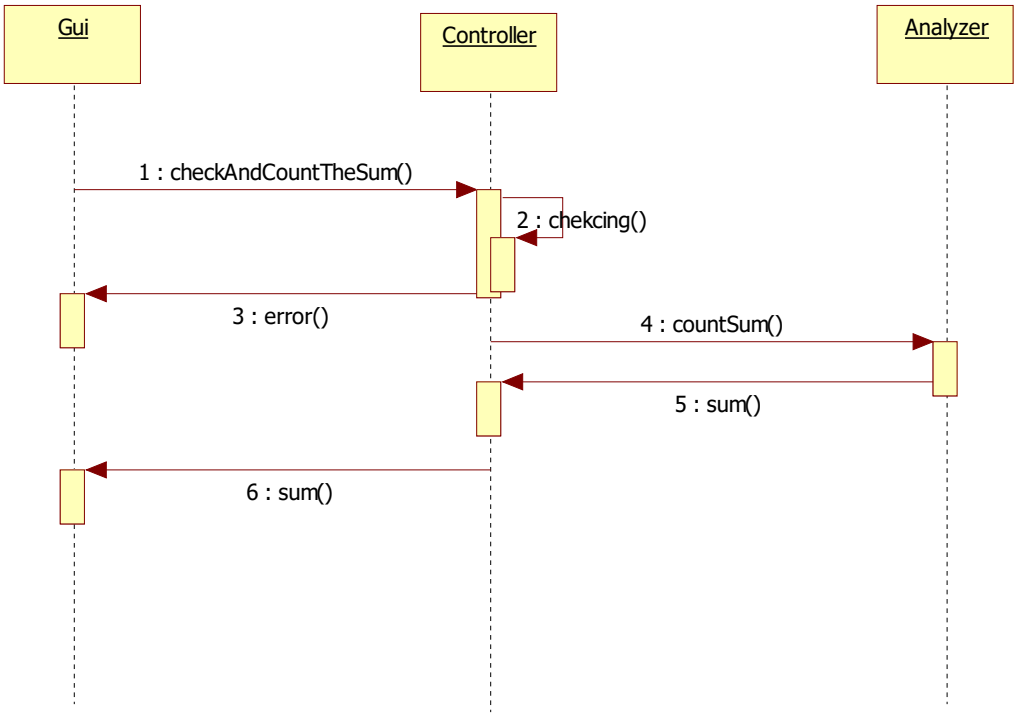
Documentation

Explanation:

User gives three integers, the sum is calculated.

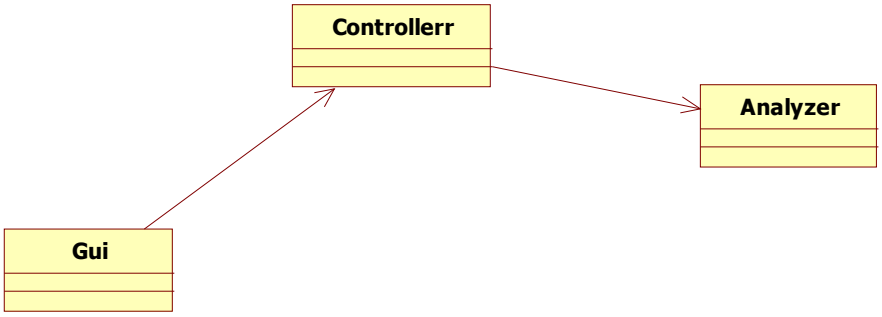
****

sequence diagram



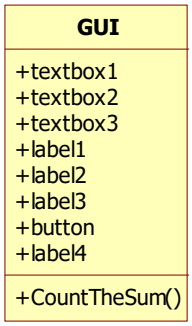
****

class diagram

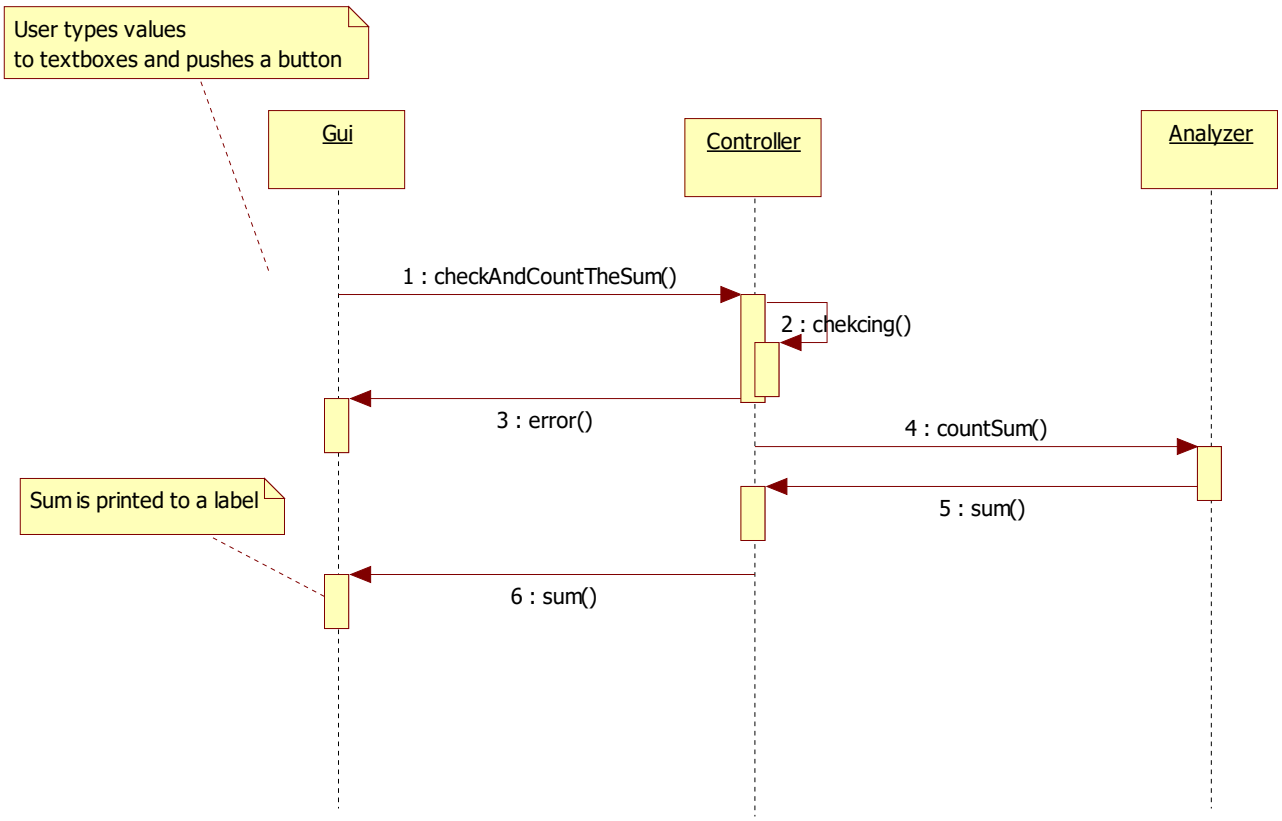


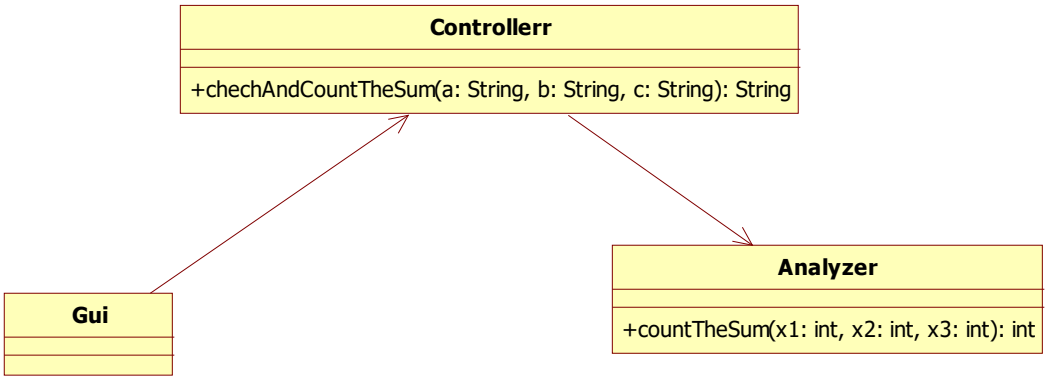
****

Gui prototype

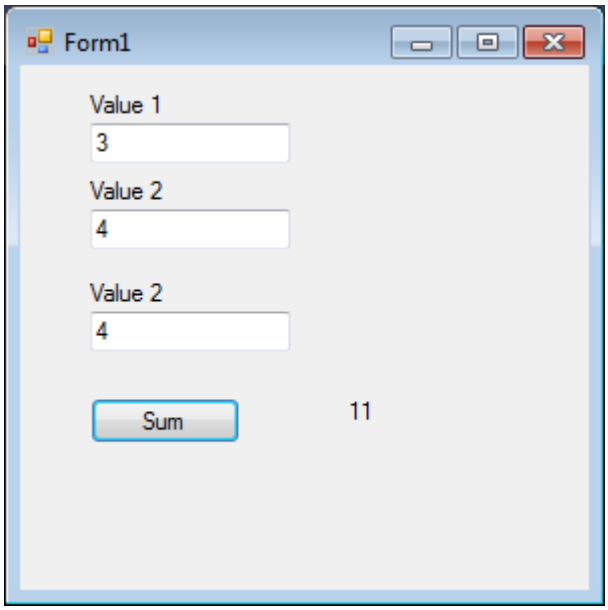


****


User types values
to textboxes and pushes a button
Gui
Controller
Analyzer
1 : checkAndCountTheSum()
2 : chekcing()
3 : error()
4 : countSum()
5 : sum()
Sum is printed to a label
6 : sum()


Controllerr
+chechAndCountTheSum(a: String, b: String, c: String): String
Gui
Analyzer
+countTheSum(x1: int, x2: int, x3: int): int

Program



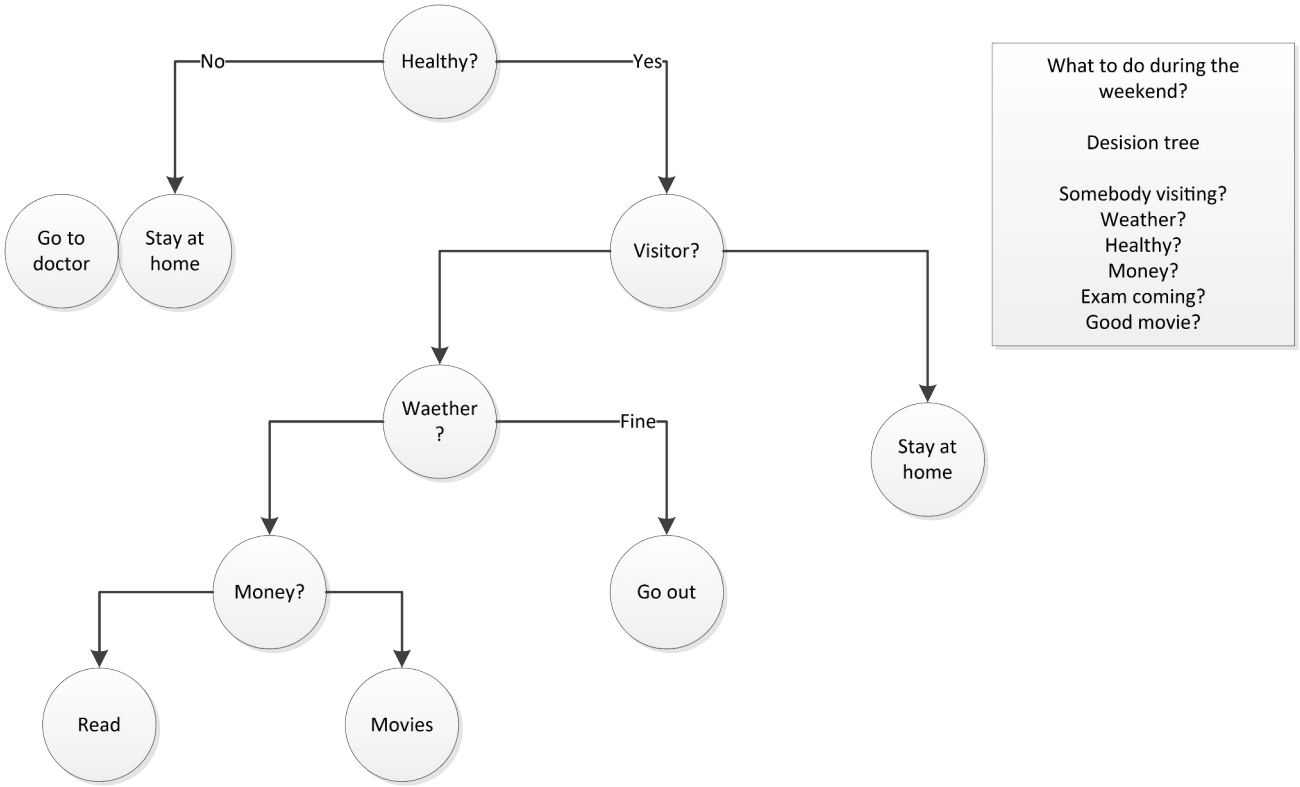

```
 Controller cont = new Controller();
private void button1_Click(object sender, EventArgs e)
{
    String a, b, c;
    a = textBox1.Text;
    b = textBox2.Text;
    c = textBox3.Text;

    String result = cont.checkAndCountTheSum(a, b, c);
    label4.Text = result;
}
```

```
class Controller
    {
        Analyzer analyz = new Analyzer();

        public String checkAndCountTheSum(String a, String b,
        String c)
        {
            String res = "";
            if (a.Length == 0 || b.Length == 0
                || c.Length == 0)
                return "Wrong data!";
            else
            {
```

```
            int aa = Convert.ToInt32(a);
            int bb = Convert.ToInt32(b);
            int cc = Convert.ToInt32(c);
            int sum = analyz.countTheSum(aa, bb, cc);
            res = res + sum;
        }

        return res;

    }




public class Analyzer
{
 public int countTheSum(int x1, int x2, int x3)
 {
     return (x1 + x2 + x3);

  }

}
```

Final topics

Desicion tree (outside UML)



Data Dictionary

Universal way to define attributes and their values.

1. Here you have data dictionary symbols:

Symbol	Meaning
+	And
()	Optional (can be missing)
{}	Repetition
N{}M	Repetition N to M times (N=min, M=max)
[]	Choices
|	Choice separator
@	Unique field
*	Comment

Example
Person name + birthday + marr. status + age + weight + gender + @ID-number + carMake

name = 1{firstname}3 + lastname

firstname = 1{alphabet}50

birhday = 1{number_1}2 + 1{number_1}2 + 4{number_1}4

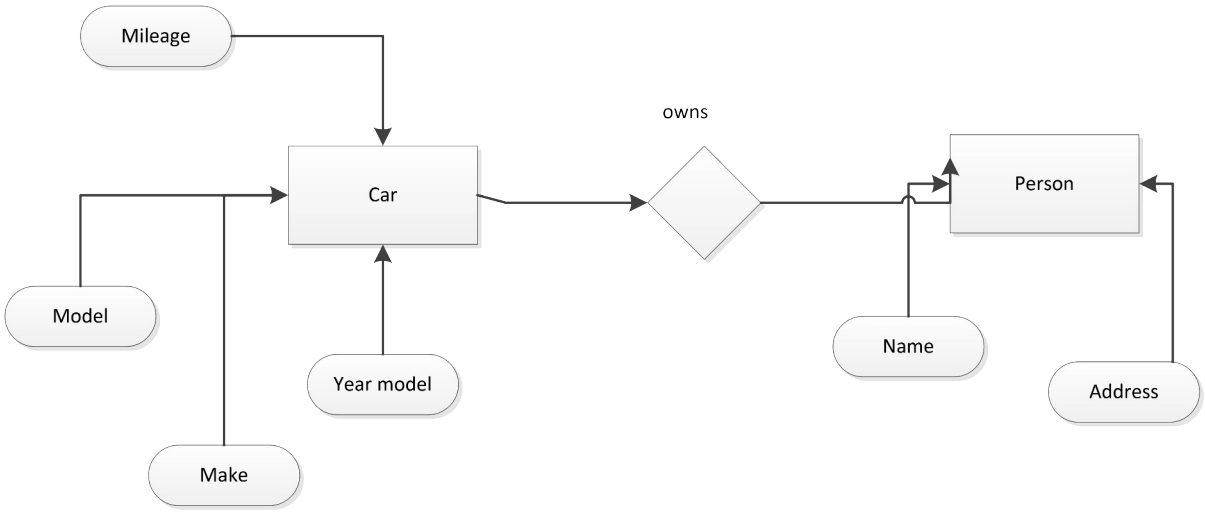
number_1 = [0 | 1 | 2 .... 9]

marr.status = [single | married | widow | devorced]

age = 1{number_1}3

*** free comment**

ER graph



Case 1: Restaurant Table Booking

Case about restaurant table booking is presented now.

All the steps from requirements (now use cases mainly) to finished application are shown.

Case 2: Blackjack Card Game

Case concerning card game is presented here.

Case 1. Restaurant Table Booking

Requirements document

Version 0.01

Topic

Restaurant table booking

Introduction

Sometimes it is time to book a table from a restaurant – for one evening.
The app that is to be created has to show free tables: user can then book some of those tables to next evening. User gets confirmation and that table is to be set reserved.
Restaurant desk person can see the reservation and keep it reserved until the right customers comes...

Use cases

Actors
Customer
Desk person

Use case documentations

Name of the use case
Booking a table

Actor
Customer

Precondition
App has been launched

Defitinion of the use case
Customer lists all the free tables by clicking a button
Customer then chooses one of free tables and confirms booking
Exception: no free tables

Exceptions
Customer is shown a message that there are no free tables

Postcondition
App is in idle state, can be closed

Use case name
Confirming

Actors
Employee

Preconditions
App is open, tables are listed and new booking is shown

Defitinion
Employee confirms the bookind by clicking a button

Postcondition
App stays in idle mode

Customer data: name
Table data: number, state

Design document

Version 0.5

By searching for nouns from use case texts we find these classes:

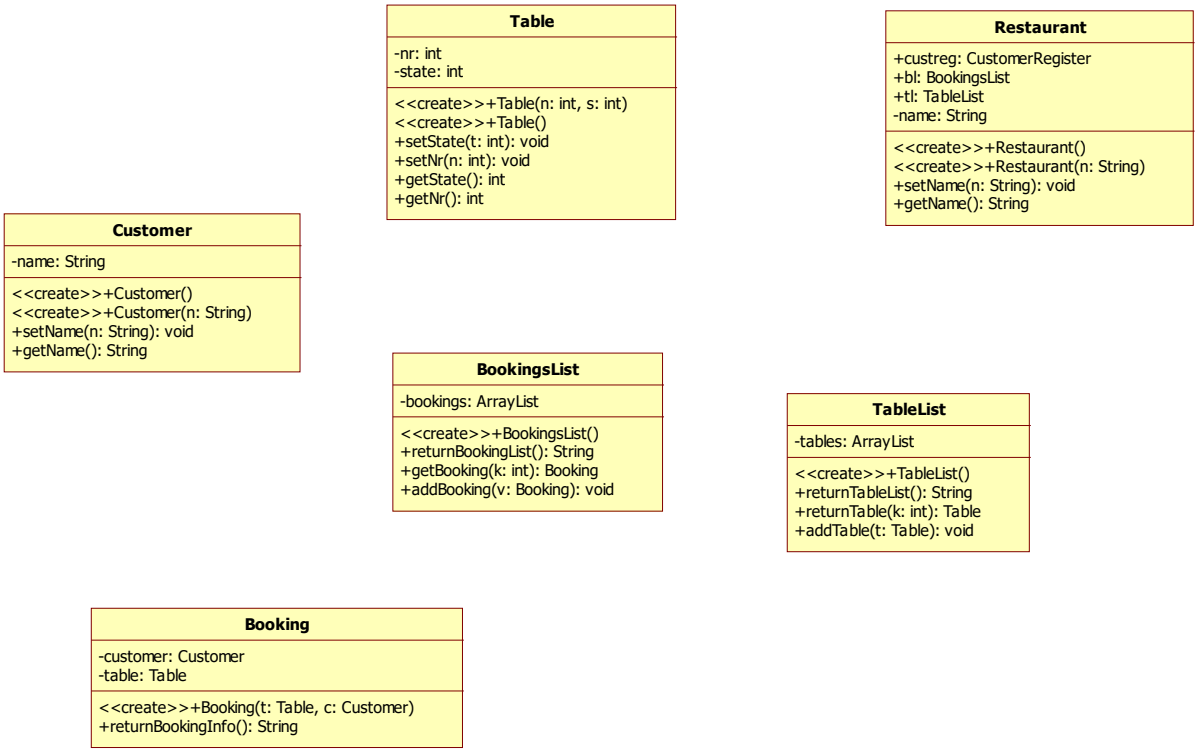
Customer

Restaurant

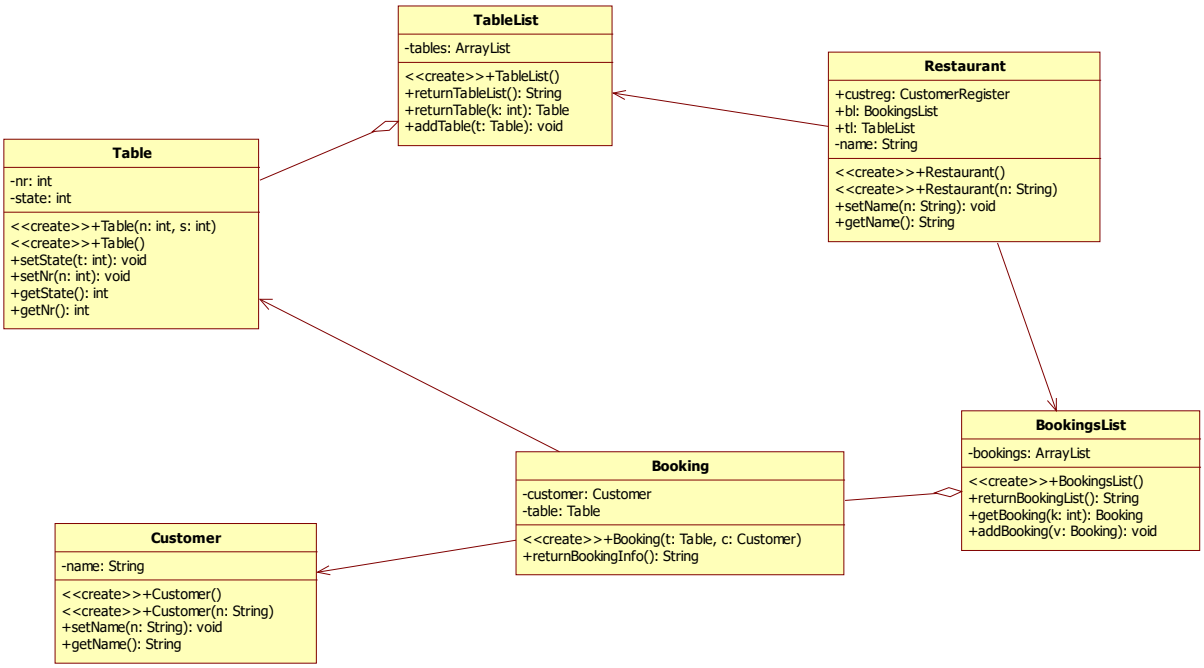
Table

Booking

Class definitions and relationships



Relationships



Implementation

Classes

```
class Customer
    {
        private String name;

        public Customer()
        {

        }

        public Customer(String n)
        {
            name = n;
        }

        public void setName(String n)
            {
              name  = n;
            }

            public String getName()
            {
                return name;
            }
```

```
        }



class Table
    {
        public Table(int n, int s)
        {
            nr = n;
            state = s;
        }

        public Table()
        {

        }


        public void setState(int t)
        {
          state = t;

        }

        public void setNr(int n)
        {
            nr = n;
        }

        public int getState()
        {;
            return state;
        }

        public int getNr()
        {
            return nr;
        }



        // Members
        int nr;
        int state;
    }




class Booking
    {
        private Customer customer;
        private Table table;

        public Booking(Table t, Customer c)
        {
            customer = c;
            table = t;
```

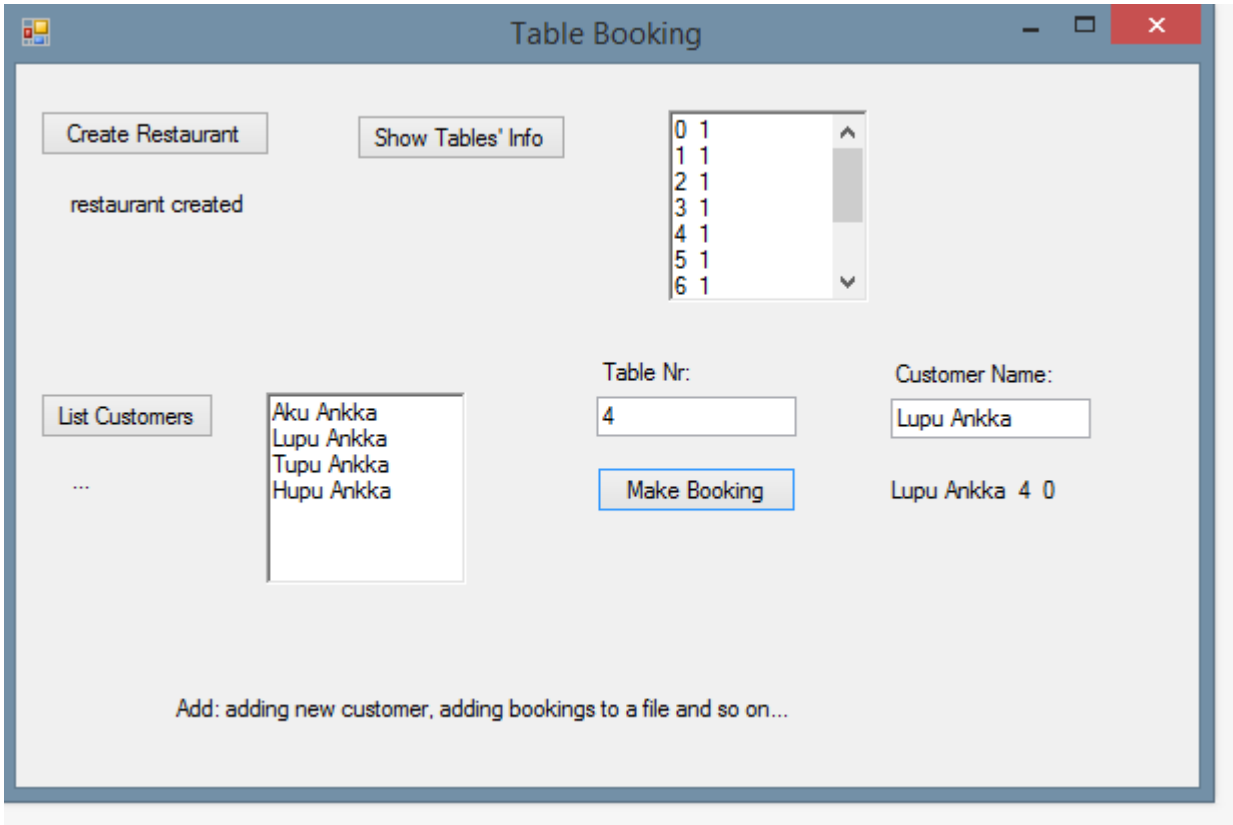
```
        }

        public String returnBookingInfo()
        {
            String info = "Date and time: " + DateTime.Now;
            info += "   " + customer.getName() + " " + table.getNr();

            return info;

        }
    }




class CustomerRegister
    {
        ArrayList customers;

        public  CustomerRegister()
        {
            customers = new ArrayList();
        }

        public String returnCustomerList()
        {
            String list = "";
            for (int k = 0; k < customers.Count; k++)
            {

                Customer c;
                c = (Customer) customers[k];
                list += c.getName();
            }
            return list;

        }


        public Customer returnCustomer(int k)
        {
            return (Customer) customers[k];
        }

        public Customer returnCustomer(String n)
        {
            Customer temp = null;
            foreach (Customer c in customers)
            {
                if (c.getName().Equals(n))
                    temp = c;

            }
            return temp;
        }



        public void addCustomer(Customer t)
```

```
        {
            customers.Add(t);
        }
    }




class TableList
    {
        private ArrayList tables;
        public  TableList()
        {
            tables = new ArrayList();
            for (int k = 0; k < 10; k++)
            {
                Table t = new Table(k, 1);
                tables.Add(t);
            }


         }



        public String returnTableList()
        {
            String list = "";
            for (int k = 0; k < tables.Count; k++)
            {

                Table t = new Table();
                t = (Table)tables[k];
                list += t.getNr() + "   " +  t.getState() + "\n";
            }
            return list;

        }


        public Table returnTable(int k)
        {
            return (Table)tables[k];
        }

        public void addTable(Table t)
        {
            tables.Add(t);
        }


    }




class BookingsList
    {
```

```
        ArrayList bookings;

        public BookingsList()
        {
            bookings = new ArrayList();
        }


        public String returnBookingList()
        {
            String list = "";
            for (int k = 0; k < bookings.Count; k++)
            {

                Booking  c;
                c = (Booking) bookings[k];
                list += c.returnBookingInfo() + "\n";
            }
            return list;

        }


        public Booking getBooking(int k)
        {
            return (Booking)bookings[k];
        }

        public void addBooking(Booking v)
        {
            bookings.Add(v);
        }

    }




class Restaurant
    {
       public CustomerRegister custreg;
      public BookingsList bl;
        public TableList tl;

        public Restaurant()
        {
            custreg = new CustomerRegister();
            bl = new BookingsList();
            tl = new TableList();
        }

        public Restaurant(String n)
        {
            custreg = new CustomerRegister();
            bl = new BookingsList();
            tl = new TableList();
            name = n;
        }
```

```
        private String name;

        public void setName(String n)
        {
            name = n;
        }

        public String getName()
        {
            return name;
        }


    }
```

Gui prototype



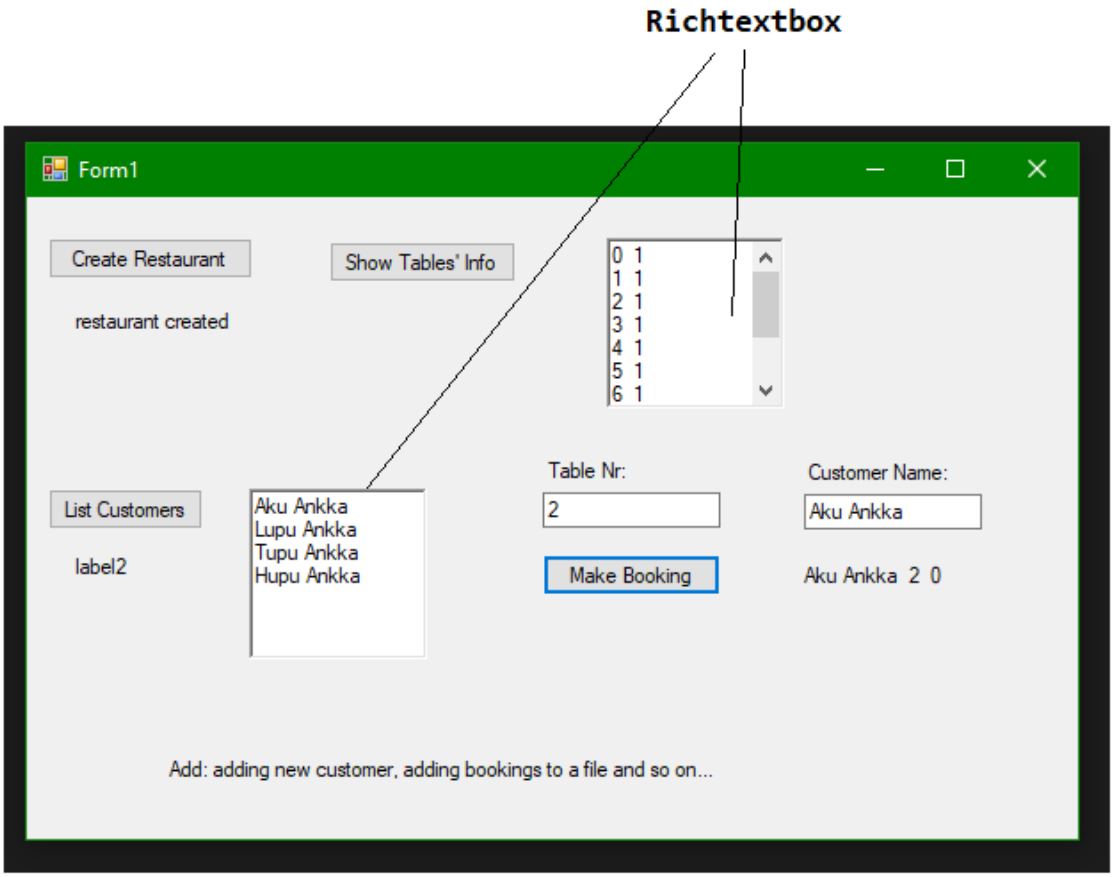
Create the app!

Make it more versatile and personal!

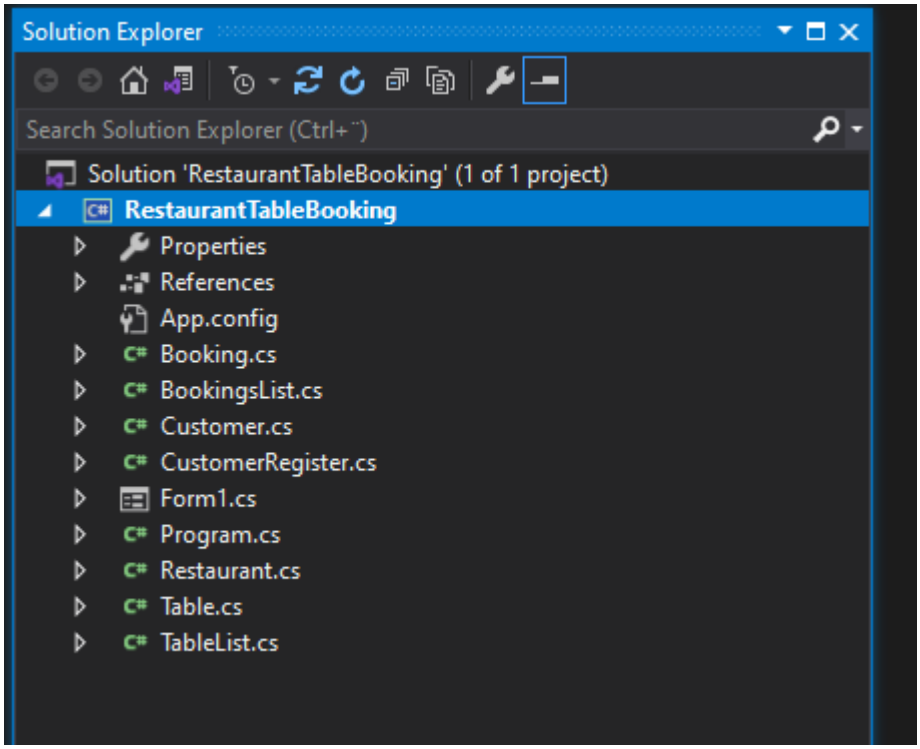
Restaurant

GUI

Create this

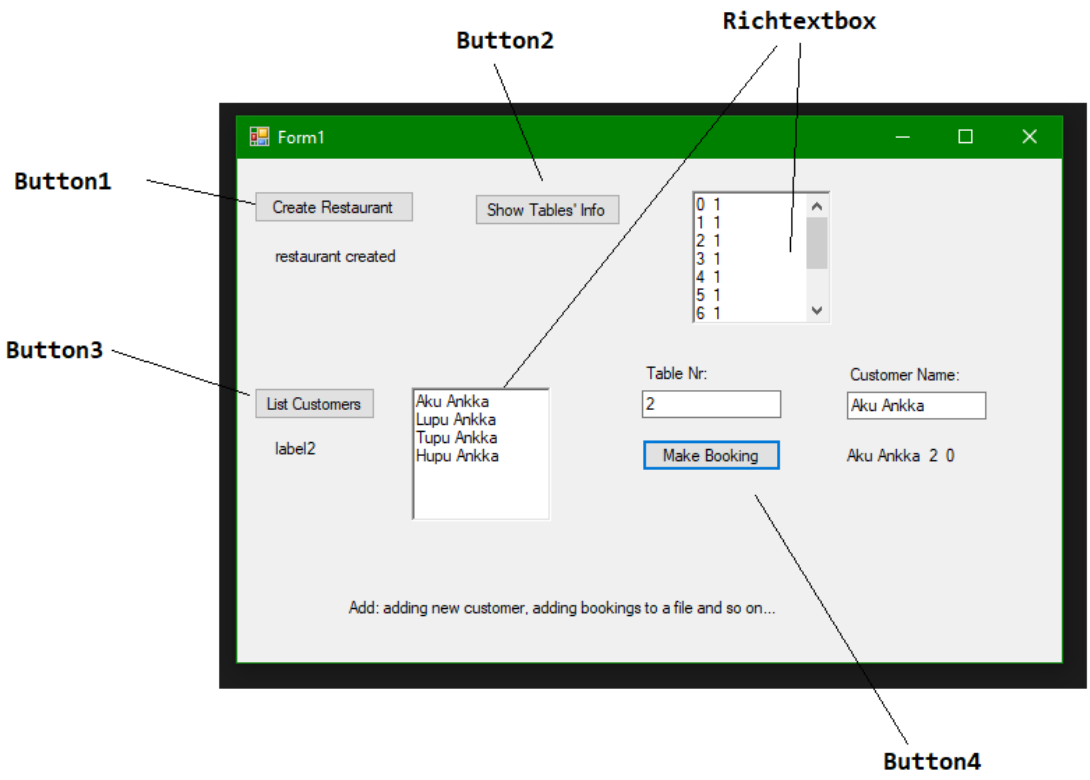


Add these classes to the project (take from console project)



Add these codes

Check button names here first




```
namespace RestaurantTableBooking
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        Restaurant rest;

        private void button1_Click(object sender, EventArgs e)
        {
            rest = new Restaurant();
            label1.Text = "restaurant created";

        }
        private void button2_Click(object sender, EventArgs e)
        {
            richTextBox1.Text = rest.tl.returnTableList();

        }
        private void button3_Click(object sender, EventArgs e)
        {
            String path = "customersFile.txt";
            String allNames = "";
            try
            {
```

```
            using (StreamReader sr = File.OpenText(path))
            {
                string s = "";
                while ((s = sr.ReadLine()) != null)
                {
                    rest.custreg.addCustomer(new Customer(s));
                    allNames += s + "\n";
                }
            }
        }
        catch (System.SystemException ee)
        {
            label2.Text = "Error - check the folder!!!";
        }
        richTextBox2.Text = allNames;
    }


    private void button4_Click(object sender, EventArgs e)
    {
        int tableNr = Convert.ToInt16(textBox1.Text);
        Table bookedTable = rest.tl.returnTable(tableNr);
        rest.tl.returnTable(tableNr).setState(0);

        String customerName = textBox2.Text;
        Customer booker = rest.custreg.returnCustomer(customerName);

        label3.Text = booker.getName() + "  " + bookedTable.getNr() +
            "  " + bookedTable.getState();

        rest.bl.addBooking(new Booking(bookedTable, booker));     }  }}
```

Test!

Add new features!

Case 2: BlackJack Card Game

BlackJack Card Game

Introduction

Equally well known as Twenty-One.

the popularity of Blackjack dates from World War I, its roots go back to the 1760s in France, where it is called Vingt-et-Un (French for 21). Today, Blackjack is the one card game that can be found in every American gambling casino. As a popular home game, it is played with slightly different rules.

The Pack The standard 52-card pack is used, but in most casinos several decks of cards are shuffled together.

Object of the Game Each participant attempts to beat the dealer by getting a count as close to 21 as possible, without going over 21.

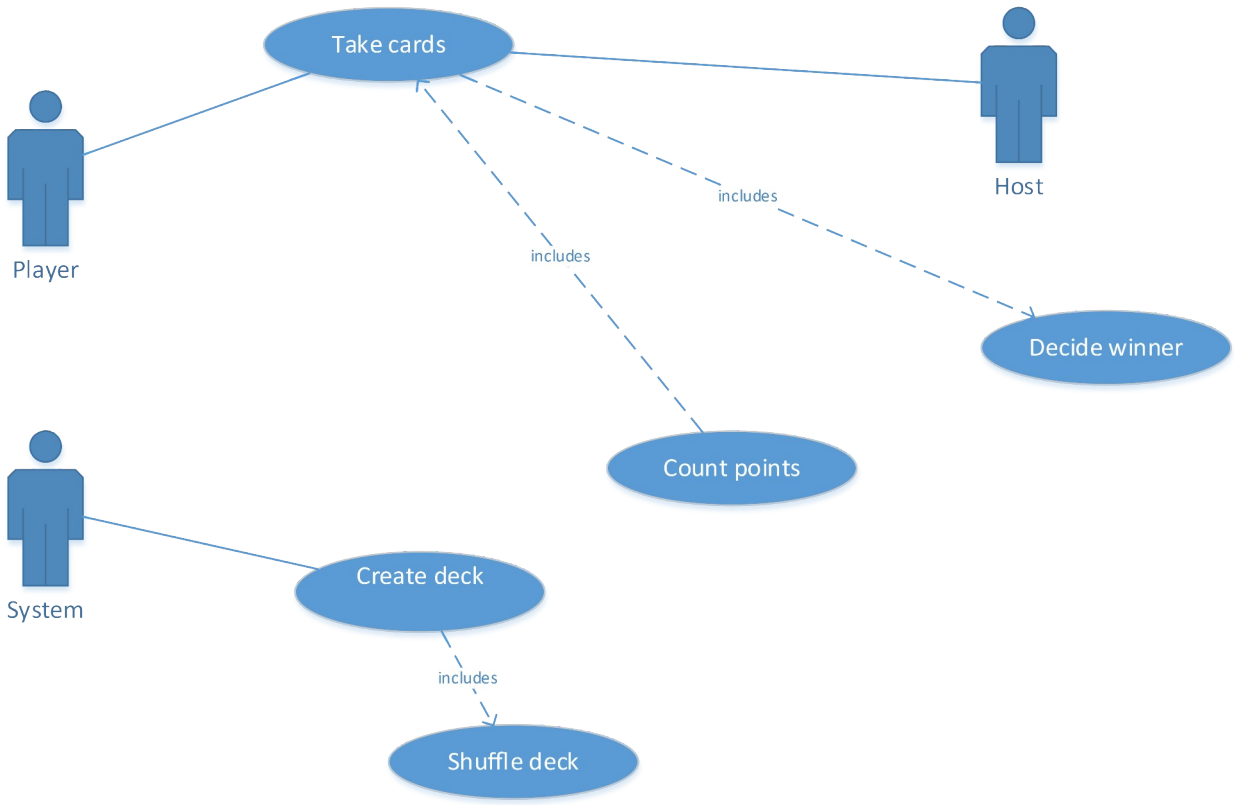
Betting is possible if wanted.

The goal is the get 21 points. Ace is here 1.

If host gets same points as player, host wins.

Requirements

Use case: functional requirements



Name	Take cards
Actors	Player, host
Preconditions	Deck is created and shuffled
Explanation	Player takes cards from the deck one by one by clicking a button. Points and card images are Shown on a form. Is points are 21, player is winner and message is shown. If points are over 21 host has won. If points are 18 – 20, host may start taking cards. Exception: Deck is empty
Exceptions	Deck is empty: if deck is empty, it has to be filled again and shuffled. Depenging on the situation player's and host's cards are taken with or not.
Postconditions	A new round can be started.

Design

From use case 1 we get this scenario:

Player takes cards from the deck one by one by clicking a button. Points and card imeges are

Shown on a form. Is points are 21, player is winner and message is shown.

If points are over 21 host has won.

If points are 18 – 20, host may start taking cards.

We can underline nouns that are class candidates:

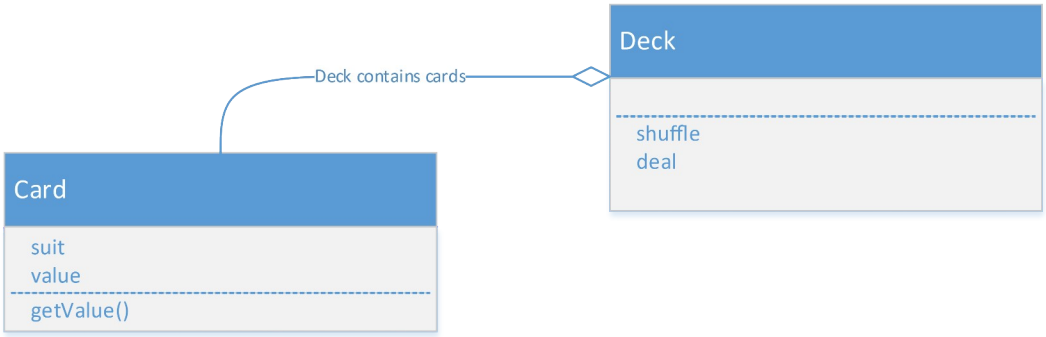
<u>Player</u> takes <u>cards</u> from the <u>deck</u> one by one by clicking a button. <u>Points</u> and card images are

Shown on a form. Is points are 21, player is winner and message is shown.

If points are over 21 host has won.

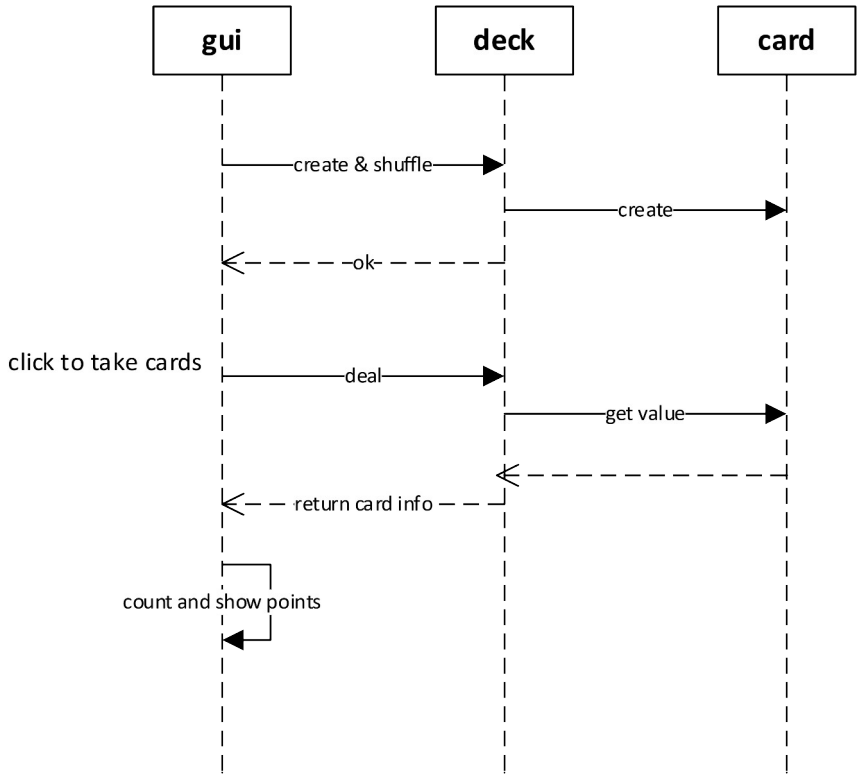
If points are 18 – 20, host may start taking cards.

Class diagram

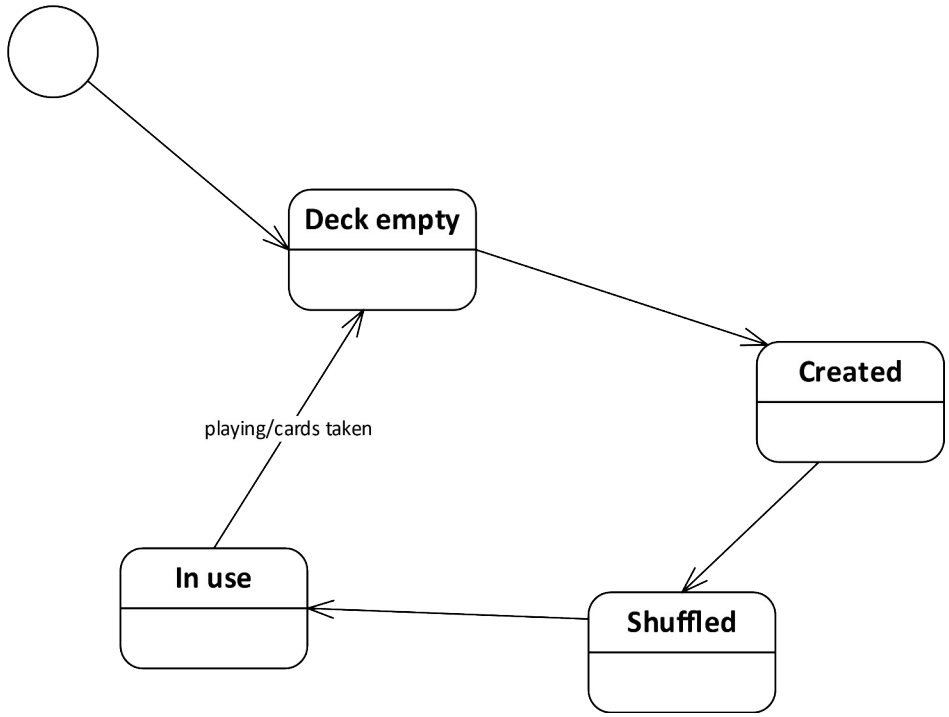


Gui prototype

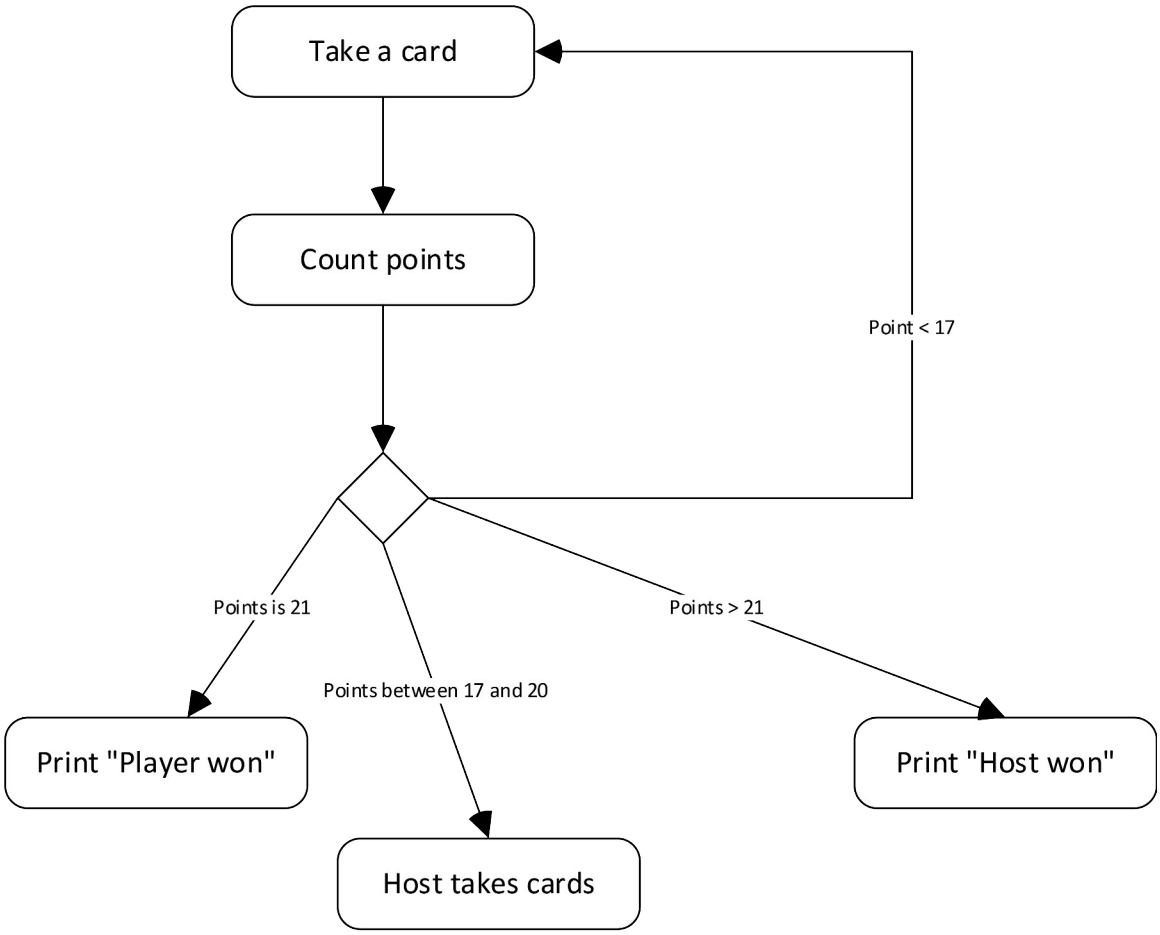
Seguence diagram


gui
deck
card
create & shuffle
create
ok
click to take cards
deal
get value
return card info
count and show points

State chart


Deck empty
Created
playing/cards taken
In use
Shuffled

Activity diagram



Final classes in code

```
class Card
    {
        private String suit;
        private String value;
        public  Card(String m, String a)
        {
            suit = m;
            value= a;
        }

        public String getFileName()
        {
```

```
            String name = "" + suit + value + ".png";
            return name;
        }


        public String returnSuit()
        {
            return suit;
        }

        public String returnValue()
        {
            return value;
        }

        public String returnCardInfo()
        {
            String[] suits = { "Club", "Spade", "Heart", "Diamond" };
            String[] values = {"Ace", "2", "3", "4", "5", "6", "7", "8", "9", "10", "Jack",
"Queen", "King" };
            int ind1 = Convert.ToInt16(suit);
            int ind2 = Convert.ToInt16(value)-1;
            String cardSuit = suits[ind1];
            String cardValue = values[ind2];
            String cardInfo = cardSuit + "  " + cardValue;
            return cardInfo;
        }
    }
```

```
class Deck
  {
      ArrayList cards = new ArrayList();

      public  Deck()
      {
          int k = 0;
          for (int m = 0; m < 4; m++)
              for (int a = 1; a < 14; a++)
              {
                  cards.Add(new Card("" + m, "" + a));
                  k++;
              }
      }


  public String  printDeck()
  {
          String allCards = "";
          foreach (Card c in cards)
              allCards += c.returnCardInfo() + "\n";

          return allCards;

  }


      public int deckSize()
      {
          return cards.Count;

      }

      public Card getFromTop()
      {
          Card temp = (Card) cards[0];
          cards.RemoveAt(0);
          return temp;
      }

      public String getThisCardFileName(int n)
      {
         Card x = (Card)cards[n];
         return x.getFileName();
      }

      public void shuffle()
      {
          Random rr = new Random();
          for (int i = 0; i < 1000; i++)
          {
              int x = rr.Next(0, 52);
              int y = rr.Next(0, 52);
              Card temp = (Card)cards[x];
               cards[x] = (Card)cards[y];
               cards[y] = temp;

          }
      }
  }
```

Cards

Cards are in png-format, added to subfolder cards:

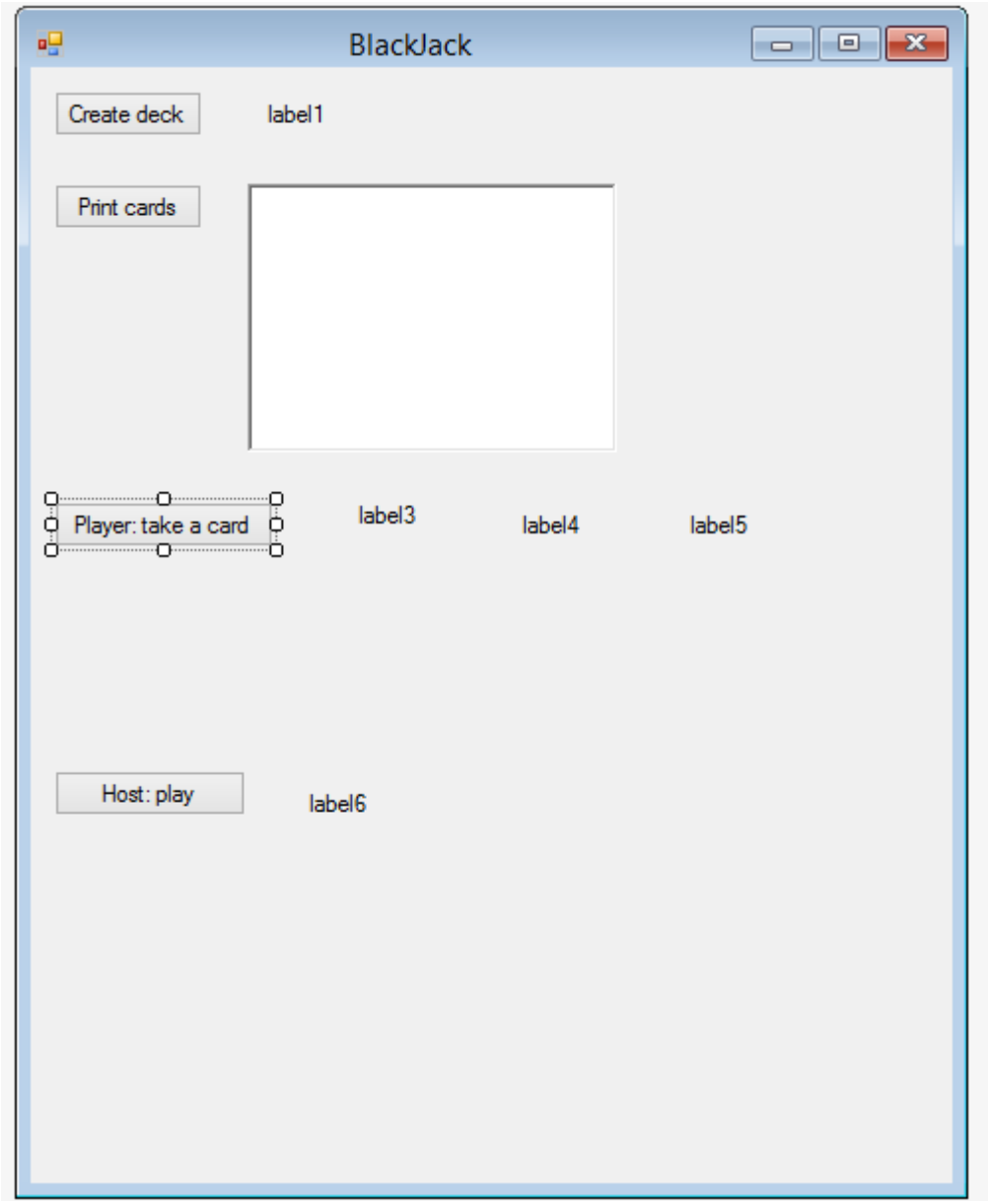
Desktop ▸ OO2017 ▸ BlackJack2017 ▸ BlackJack2017 ▸ bin ▸ Debug ▸ cards

Name	Date	Type	Size	Tags
01.png	1.4.2017 16:04	PNG File	1 KB	
02.png	1.4.2017 16:04	PNG File	1 KB	
03.png	1.4.2017 16:04	PNG File	1 KB	
04.png	1.4.2017 16:04	PNG File	1 KB	
05.png	1.4.2017 16:04	PNG File	1 KB	
06.png	1.4.2017 16:04	PNG File	1 KB	
07.png	1.4.2017 16:04	PNG File	1 KB	
08.png	1.4.2017 16:04	PNG File	1 KB	

Cards are named so that the 1. Number tells the suit and 2. Value the card's value.

Desktop ▸ OO2017 ▸ BlackJack2017 ▸ BlackJack2017 ▸ bin ▸ Debug ▸ cards

Final gui



Using of exceptions

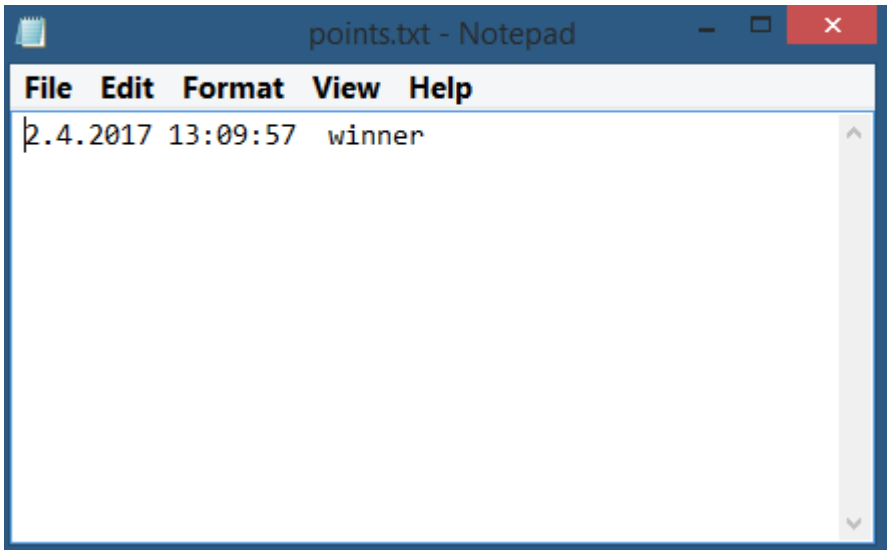
Here, if deck is empty when you try to take a card, an exception is thrown - programn does not crash byt gives a message about the situation:

```
 Card taken = null;
try
{
    taken = deck.getFromTop();
}
catch (Exception ex)
{
```

```
            label3.Text = "deck is empty!!  " + ex.Message;
        }
```

File IO

Date and time and winner are added to the file, points.txt that is saved to debug-folder:



Function that handles file IO:

```
static String saveToFile(int w)
      {
          String winner = "";
          if (w == 1)
              winner = "player";
          else
              winner = "host";
          String now = "" +  DateTime.Now;
          String message = now + "  winner";

          String filename = "points.txt";
          FileStream fs;
          try
          {
              using (fs = new FileStream(filename, FileMode.Append,
                 FileAccess.Write))
              using (StreamWriter sw = new StreamWriter(fs))
              {
                  sw.WriteLine(message);
              }
          }
          catch (System.SystemException ee)
          {
              return ("Error - check the folder!!!");
          }
          return (winner);
```

```
}
```

Test:

What can be added to the next version?
Betting

Starting a new round

Choosing the value of ace (1 or 11 or 14)

And so on.

Appendix 1: Whole code

```
sing System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace BlackJack2017
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        PictureBox[] boxes = new PictureBox[10];

        private void Form1_Load(object sender, EventArgs e)
        {
           for (int k = 0; k < 10; k++)
            {
                boxes[k] = new PictureBox();
                boxes[k].SizeMode = PictureBoxSizeMode.StretchImage;
                boxes[k].Height =70;
                boxes[k].Width = 50;
                boxes[k].Parent = this;

            }
        }

        Deck deck;

        private void button1_Click(object sender, EventArgs e)
        {
            deck = new Deck();
            deck.shuffle();

            label1.Text = "Deck ok";
        }



        private void button3_Click(object sender, EventArgs e)
```

```
{
    richTextBox1.Text = deck.printDeck();
}



static String saveToFile(int w)
{
    String winner = "";
    if (w == 1)
        winner = "player";
    else
        winner = "host";
    String now = "" +  DateTime.Now;
    String message = now + "  winner";

    String filename = "points.txt";
    FileStream fs;
    try
    {
        using (fs = new FileStream(filename, FileMode.Append,
           FileAccess.Write))
        using (StreamWriter sw = new StreamWriter(fs))
        {
            sw.WriteLine(message);
        }
    }
    catch (System.SystemException ee)
    {
        return ("Error - check the folder!!!");
    }
    return (winner);

}

String path = "cards\\";
int points1 = 0;
int count1 = -1;
private void button5_Click(object sender, EventArgs e)
{
    count1++;
    Card taken = null;
    try
    {
        taken = deck.getFromTop();
    }
    catch (Exception ex)
    {
       label3.Text = "deck is empty!!  " + ex.Message;
    }
    points1 += Convert.ToInt16(taken.returnValue());


    label3.Text = "" + points1;
    String picFile = path + taken.getFileName();
    //deck.getThisCardFileName(0);
    label4.Text = picFile + " " + deck.deckSize();
    boxes[count1].Top = 260;
    boxes[count1].Left = 10 + count1 * 60;
    boxes[count1].Image = Image.FromFile(picFile);
```

```
            if (points1 >= 18 && points1 < 21)
                label5.Text = "Host's turn!";
            else if (points1 == 21)
            {

                label5.Text = saveToFile(1);

            }
            else if (points1 > 21)
                label5.Text = saveToFile(0);

            count2++;

        }
        int count2;
        int points2 = 0;
        int place = 0;
        private void button6_Click(object sender, EventArgs e)
        {
            place++;

            count2++;
            Card taken = deck.getFromTop();
            points2 += Convert.ToInt16(taken.returnValue());

            label6.Text = "" + points2;
            String picFile = path + taken.getFileName();
            boxes[count2].Top = 400;
            boxes[count2].Left = 10 + place * 60;
            boxes[count2].Image = Image.FromFile(picFile);

            if (points2 >= points1 && points2 < 21)
                label6.Text = saveToFile(0);
            else if (points2 > 21)
                label6.Text = saveToFile(1);

        }
    }
}
```