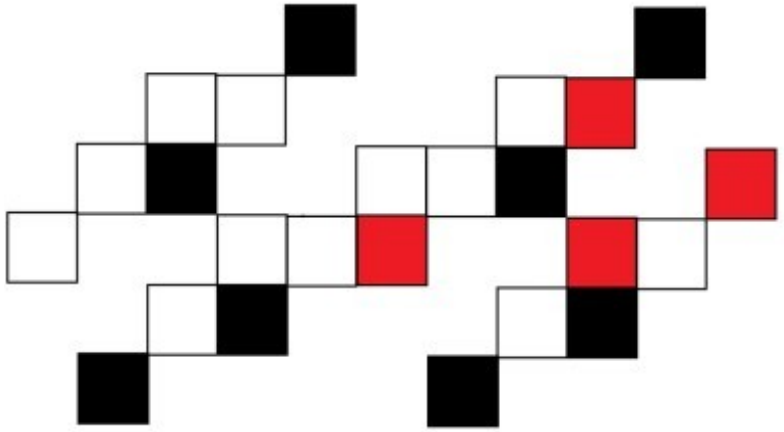
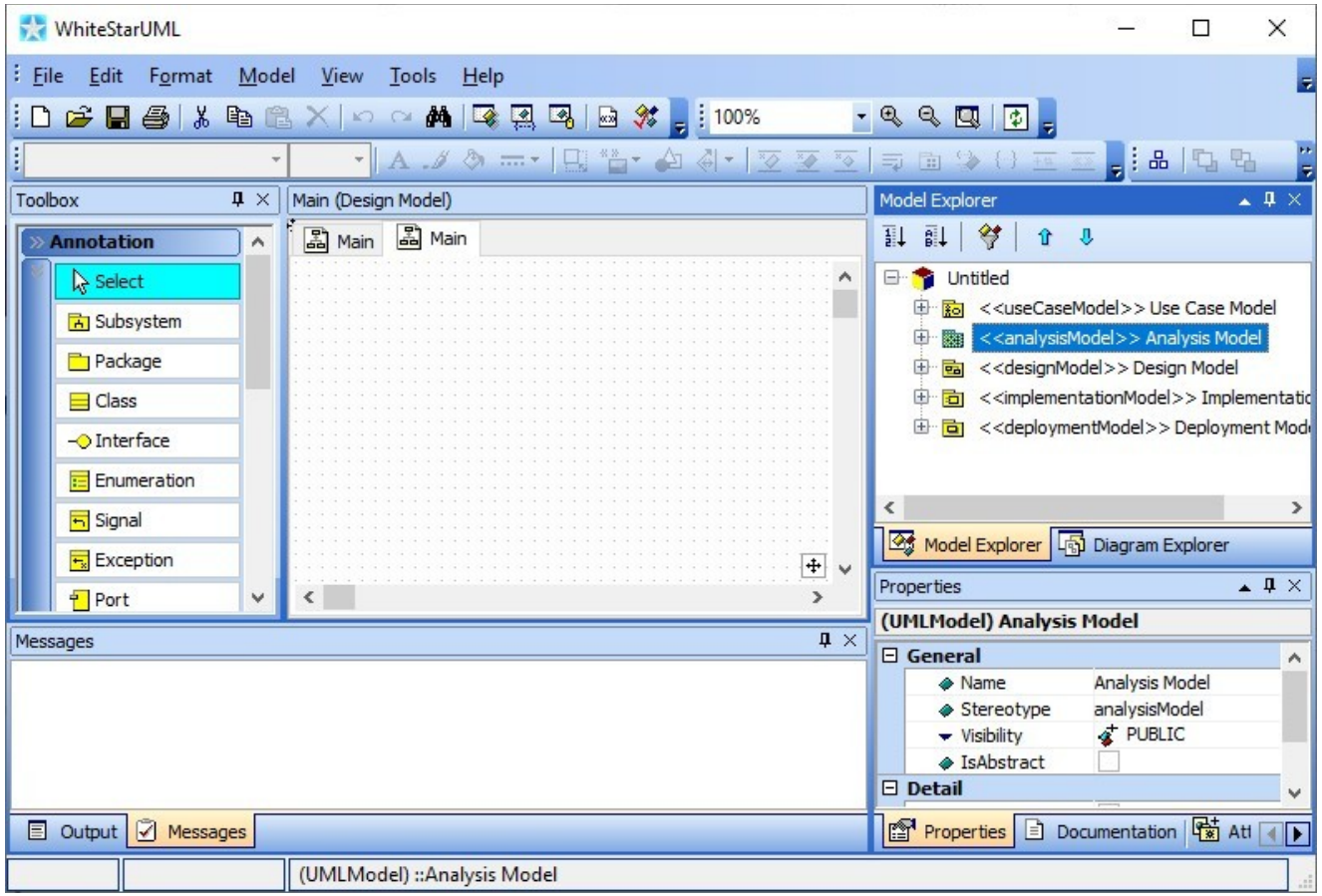
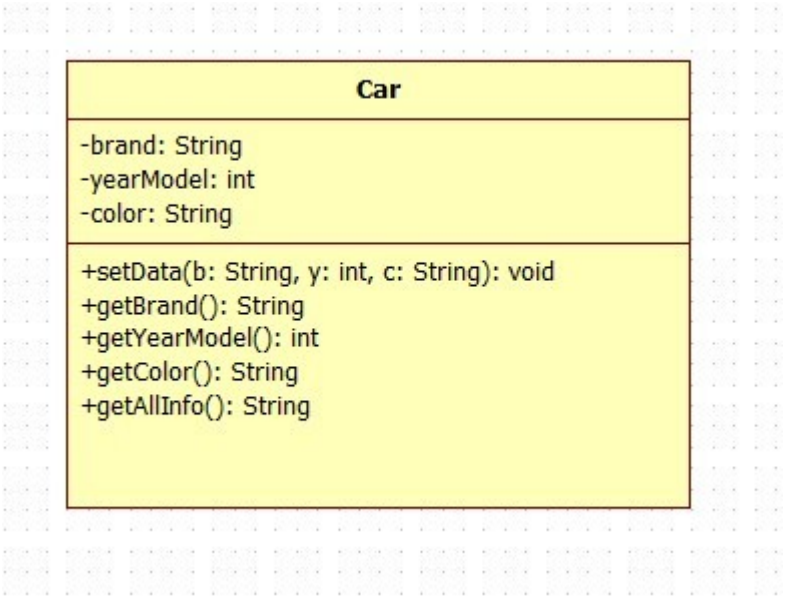
Visual Studio: OOP part 1



Visual Studio: OOP part 1 Tool for modelling: WhiteStarUML

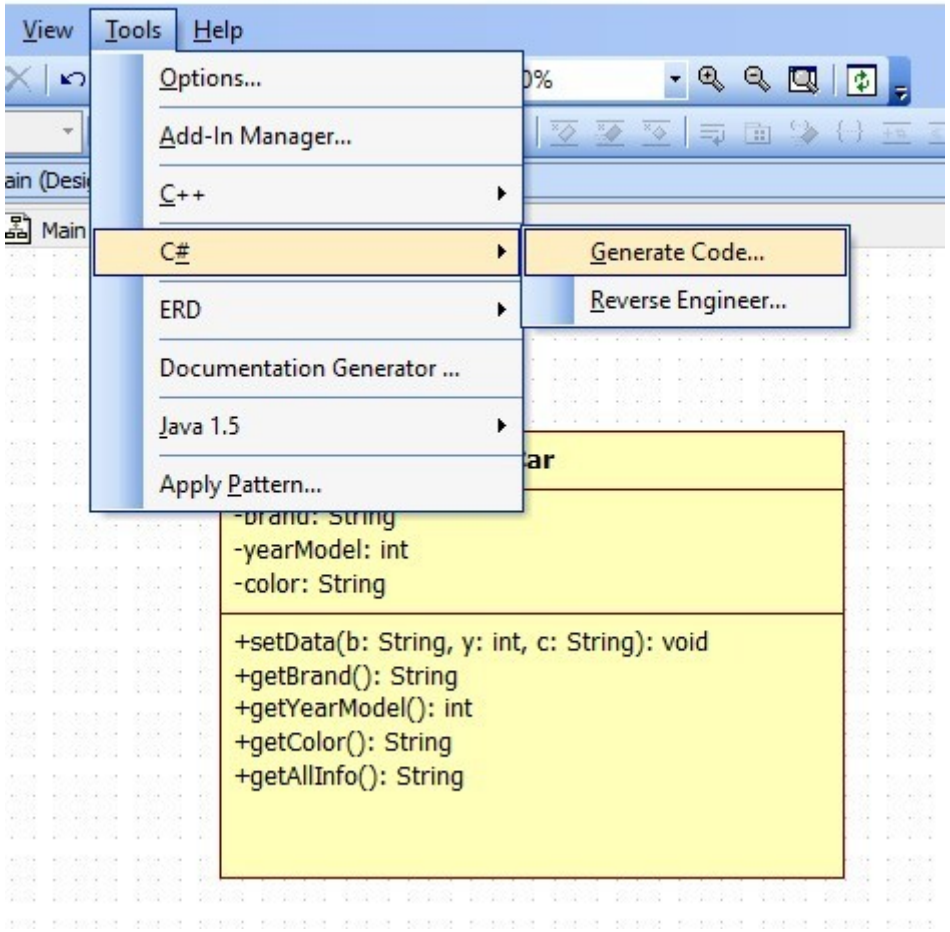


Visual Studio: OOP part 1 Car class modelled



Visual Studio: OOP part 1
Add C# profile,
Generate code

Visual Studio: OOP part 1
Add C# profile,
Generate code



Visual Studio: OOP part 1
Add C# profile,
Generate code

```
//  @ File Name : Car.cs

public class Car {
    private String brand ;
    private int yearModel ;
    private String color ;
    public void setData(String b, int y, String c){
    }
    public String getBrand(){
    }
    public int getYearModel(){
    }
    public String getColor(){
    }
    public String getAllInfo(){
    }
}
```

Visual Studio: OOP part 1 Take code to Visual Studio project

Program.cs

ConsoleApp11

```
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApp11
8   {
9       public class Car
10      {
11          private String brand;
12          private int yearModel;
13          private String color;
14          public void setData(String b, int y, String c)
15          {
16          }
17          public String getBrand()
18          {
19          }
20          public int getYearModel()
21          {
22          }
23          public String getColor()
24          {
25          }
26          public String getAllInfo()
27          {
28          }
29      }
30
31      class Program
32      {
33          static void Main(string[] args)
34          {
35          }
36      }
37  }
```

Visual Studio: OOP part 1 Take code to Visual Studio project

Program.cs

ConsoleApp11

```
3   using System.Linq;
4   using System.Text;
5   using System.Threading.Tasks;
6
7   namespace ConsoleApp11
8   {
9       public class Car
10      {
11          private String brand;
12          private int yearModel;
13          private String color;
14          public void setData(String b, int y, String c)
15          {
16          }
17          public String getBrand()
18          {
19          }
20          public int getYearModel()
21          {
22          }
23          public String getColor()
24          {
25          }
26          public String getAllInfo()
27          {
28          }
29      }
30
31      class Program
32      {
33          static void Main(string[] args)
34          {
35          }
36      }
37  }
```

Visual Studio: OOP part 1

Implement methods

```
public void setData(String b, int y, String c)
{
    brand = b;  yearModel = y; color = c;
}

public String getBrand()
{
    return brand;
}
public int getYearModel()
{
    return yearModel;
}
public String getColor()
{
    return color;
}
public String getAllInfo()
{
    String info = "Brand is " + brand + ", year model  is " + yearModel +
        " and color is " + color;
    return info;
}
```

Visual Studio: OOP part 1

Test:

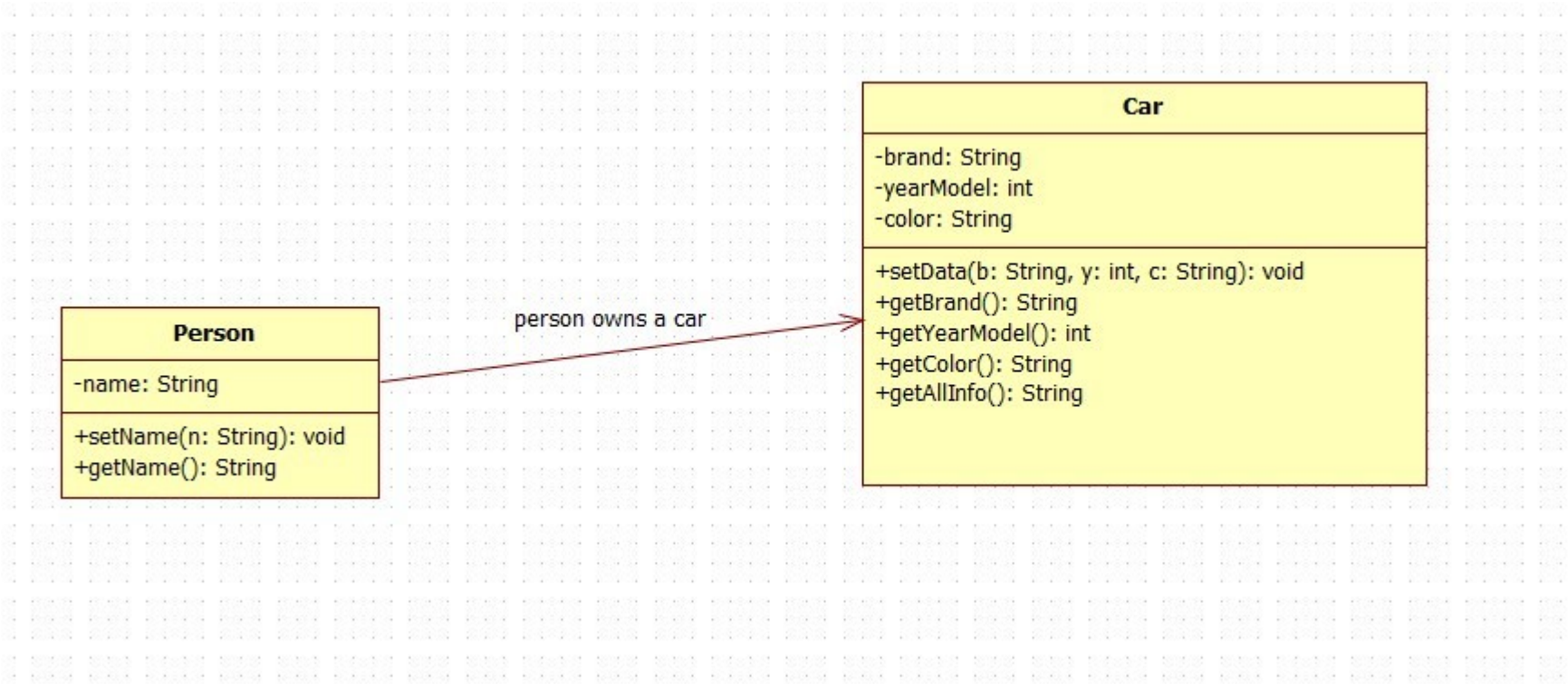
```
static void Main(string[] args)
{
    Car myCar = new Car();
    myCar.setData("Fiat", 2010, "Blue");
    Console.WriteLine(myCar.getAllInfo());

    Console.Read();
}
```

```
Brand is Fiat, year model  is 2010 and color is Blue
```

Visual Studio: OOP part 1

Let's put owner with!

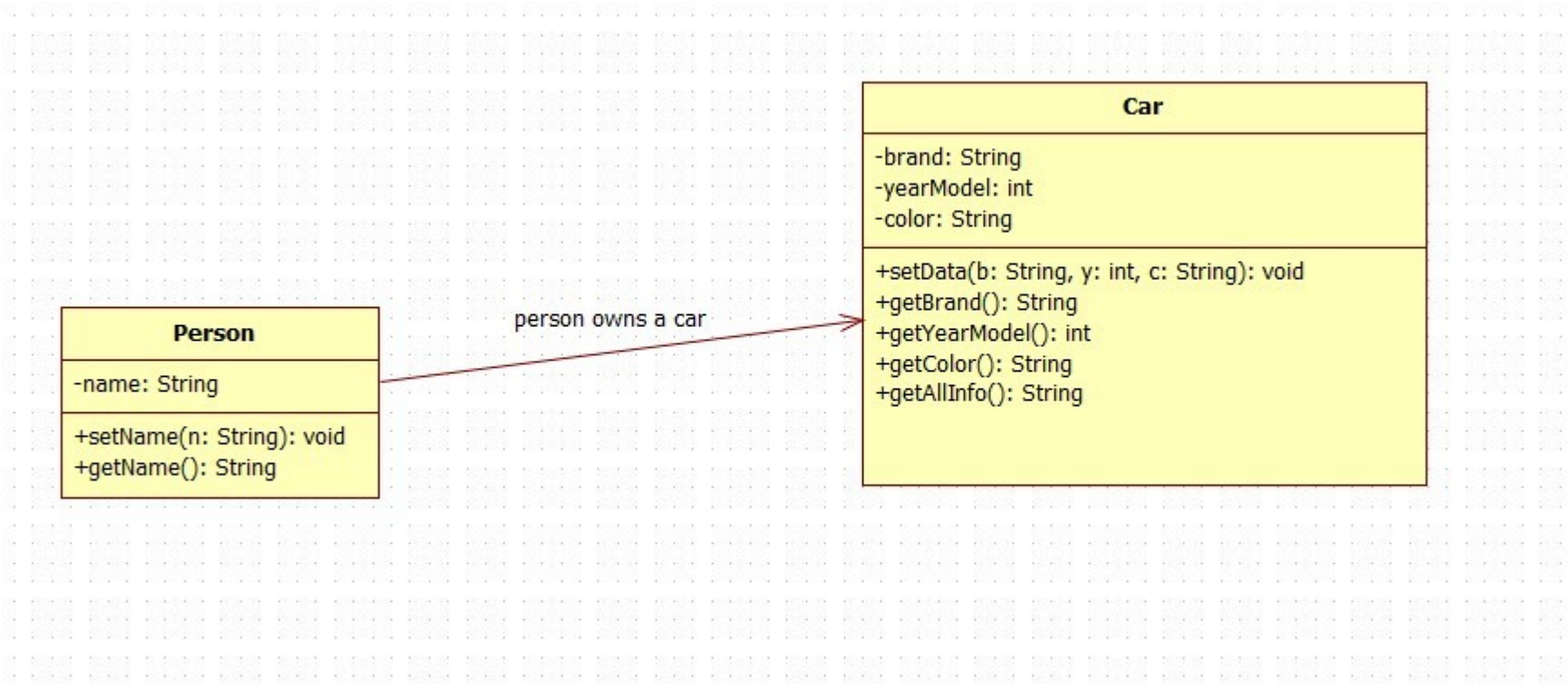
Association between person and car



Visual Studio: OOP part 1

Let's put owner with!

Association between person and car



Visual Studio: OOP part 1

Let's put owner with!

```java
public class Person {
    private String name ;
    public void setName(String n){
    }
    public String getName(){
    }
}
```

```java
public class Person
{
    private String name;
    public void setName(String n)
    {
        name = n;
    }
    public String getName()
    {
        return name;
    }
}
```

Visual Studio: OOP part 1 Associaton has to be implemented

```
public class Person
{
    private String name;
    private Car ownedCar;
    public void setCar(Car car)
    {
        ownedCar = car;
    }

    public String getAllPersonInfo()
    {
        String personInfo = "Name is " + name + " and car info is:" + ownedCar.getAllInfo();
        return personInfo;
    }
    public void setName(String n)
    {
        name = n;
    }
    public String getName()
    {
        return name;
    }
}
```

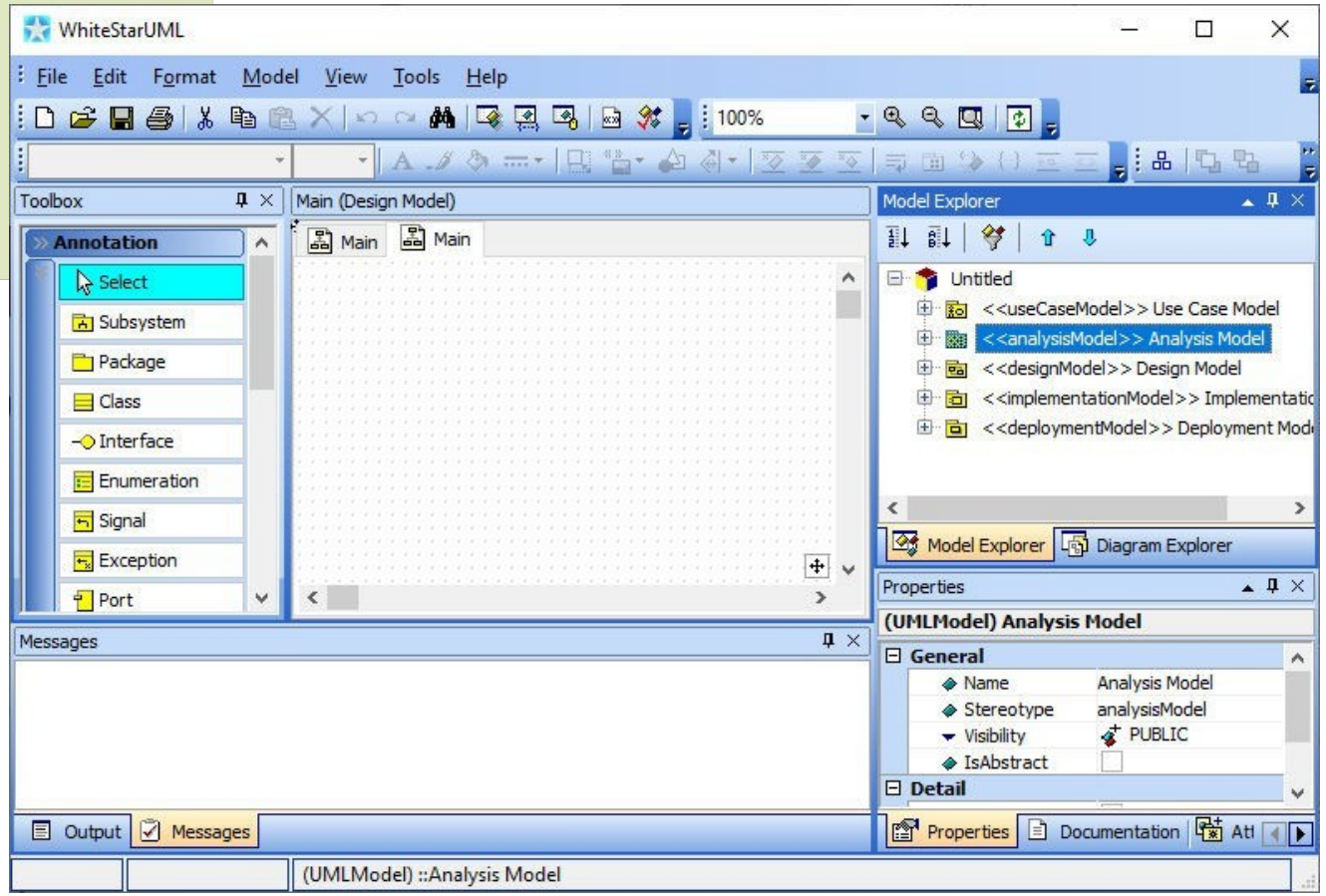
Visual Studio: OOP part 1
Testing

```
Car myCar = new Car();
myCar.setData("Fiat", 2010, "Blue");
Console.WriteLine(myCar.getAllInfo());

Person person1 = new Person();
person1.setName("Jekaterina Bella Tsitzix");
person1.setCar(myCar);
Console.WriteLine(person1.getAllPersonInfo());
```

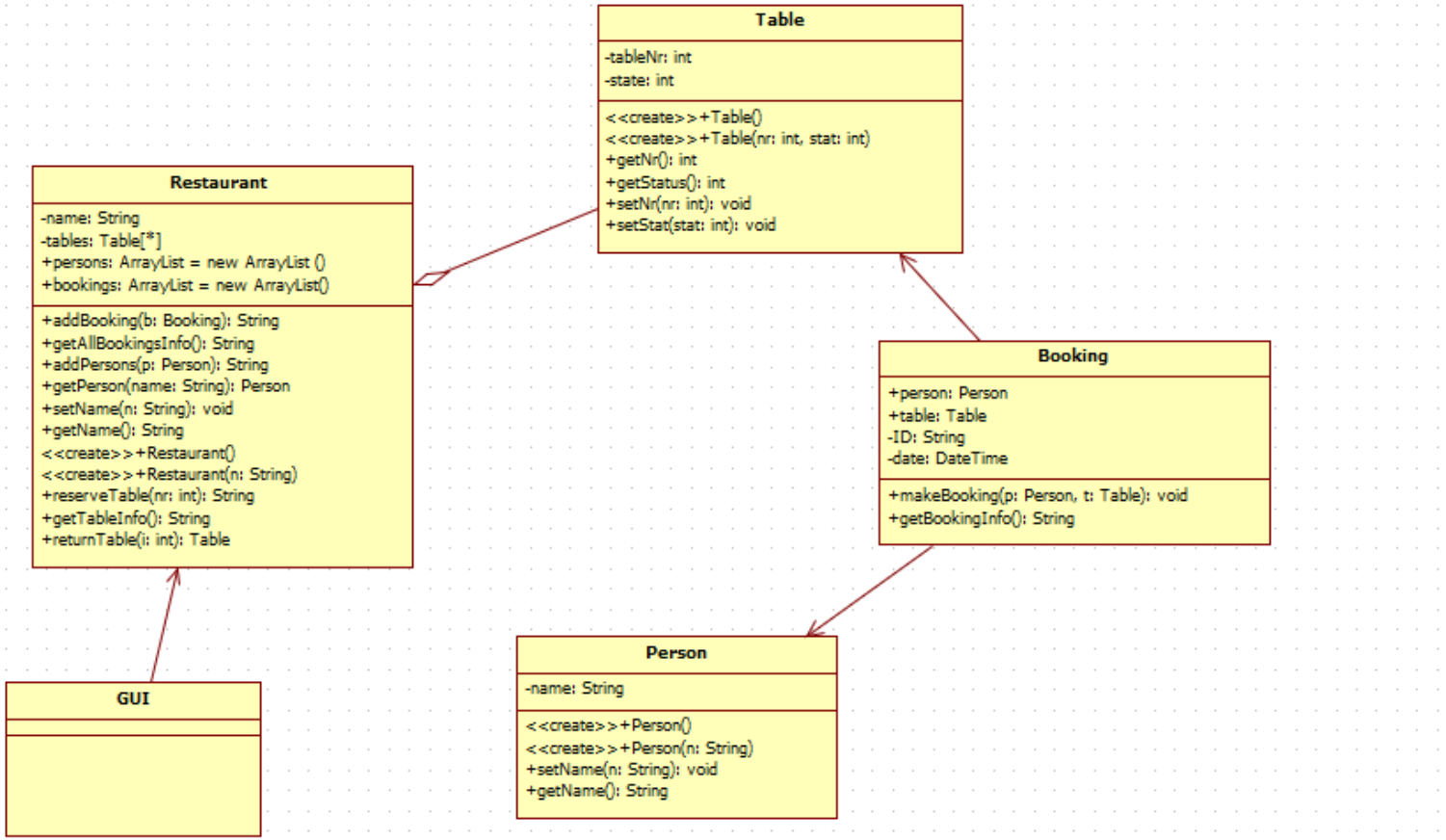
```
Brand is Fiat, year model  is 2010 and color is Blue
Name is Jekaterina Bella Tsitzix and car info is: Brand is Fiat, year model  is 2010 and color is Blue
```

Visual Studio: OOP part 2

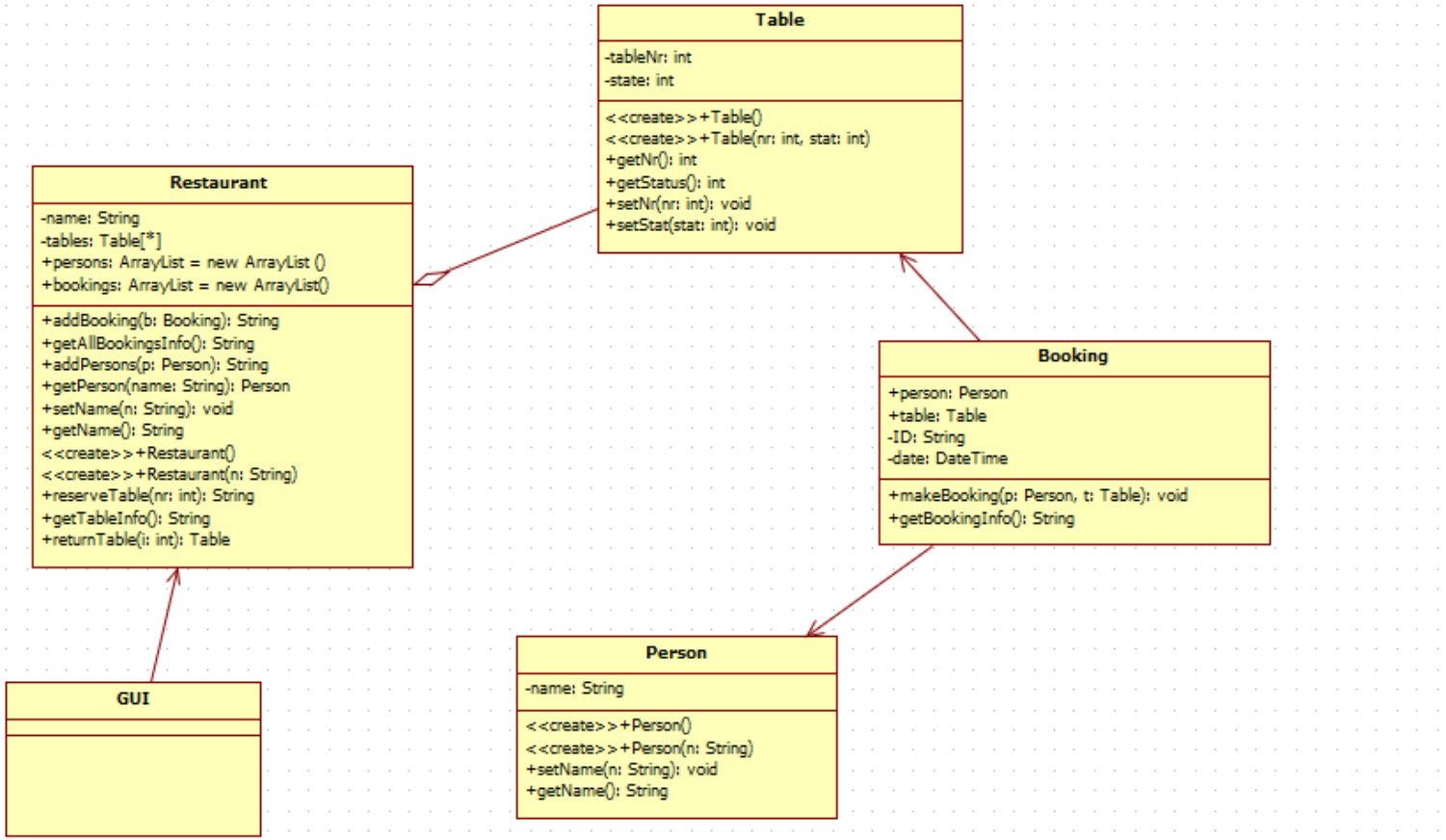
Tool for modelling: WhiteStarUML



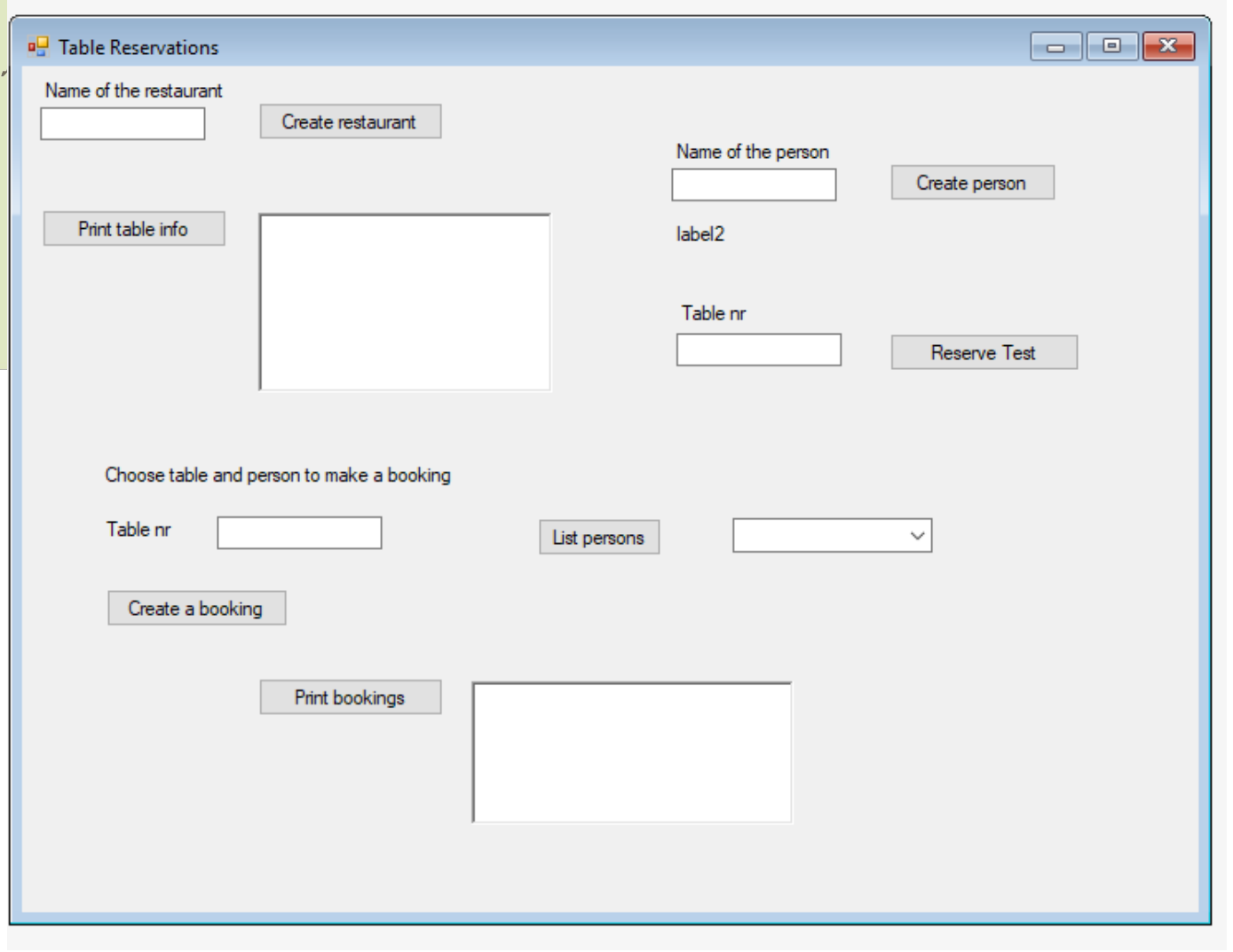
Visual Studio: OOP part 2
Restaurant Table Booking modelled



Visual Studio: OOP part 2
Restaurant Table Booking modelled



Visual Studio: OOP part 2
Create a new Visual Studio Project with this kind of gui



Visual Studio: OOP part 2
Create codes

1) classes: Person

```
public class Person
{
    private String name;
     public Person()
     {

     }
     public Person(String n)
     {
         name = n;
     }
     public void setName(String n)
     {
         name = n;
     }
     public String getName()
     {
         return name;
     }
}
```

Visual Studio: OOP part 2
Create codes

1) classes: Table

```
class Table
{
    private int tableNr;
    private int state;
    public Table()
    {
        tableNr = -1;
        state = -1;
    }
    public Table(int nr, int stat)
    {
        tableNr = nr;
        state = stat;
    }
    public int getNr()
    {
        return tableNr;
    }
    public int getStatus()
    {
        return state;
    }
    public void setNr(int nr)
    {
        tableNr = nr;
    }

    public void setStat(int stat)
    {
        state = stat;
    }

}
```

Visual Studio: OOP part 2

Create codes

1) classes: Restaurant a)

```
class Restaurant
{
    private String name;
    private Table[] tables;
    public ArrayList persons = new ArrayList ();
    public ArrayList bookings = new ArrayList();
```

Visual Studio: OOP part 2

Create codes

1) classes: Restaurant b)

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

```
public Restaurant()
{
    tables = new Table[10];
    for (int k = 0; k < 10; k++)
    {
        tables[k] = new Table(k, 0);
    }
}
public Restaurant(String n)
{
    name = n;
    tables = new Table[10];
    for (int k = 0; k < 10; k++)
    {
        tables[k] = new Table(k, 0);
    }
}
```

Visual Studio: OOP part 2

Create codes

1) classes: Restaurant c)

```
public String addBooking(Booking b)
{
    bookings.Add(b);
    return "Booking done";
}
public String getAllBookingsInfo()
{
    String info = "";
    foreach (Booking bb in bookings)
    {
        info += bb.getBookingInfo();
    }
    return info;
}
```

Visual Studio: OOP part 2

Create codes

1) classes: Restaurant d)

```
public String addPersons(Person p)
{
    persons.Add(p);
    return "person added to the storage";
}
public Person getPerson(String name)
{
    foreach (Person pp in persons)
    {
        if (pp.getName().Equals(name))
            return pp;
    }
    return null;
}
```

Visual Studio: OOP part 2

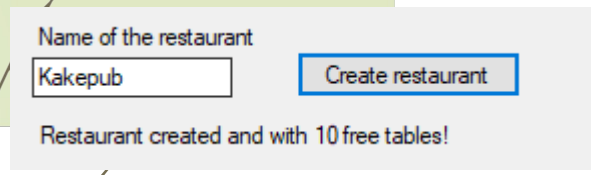
Create codes

1) classes: Restaurant f)

```
public String reserveTable(int nr)
{
    String info = "Table is reserved to you!";
    if (tables[nr].getStatus() == 0)
        tables[nr].setStat(1);
    else
        info = "Table can not be reserved!";
    return info;
}
public String getTableInfo()
{

    String ti = "Table situation: \n";
    for (int k = 0; k < 10; k++)
    {
        ti = ti + "nro: " + tables[k].getNr() + " is reserved? " +
                tables[k].getStatus() + "\n";
    }
    return ti;
}
public Table returnTable(int i)
{
    return tables[i];
}
```

Visual Studio: OOP part 2

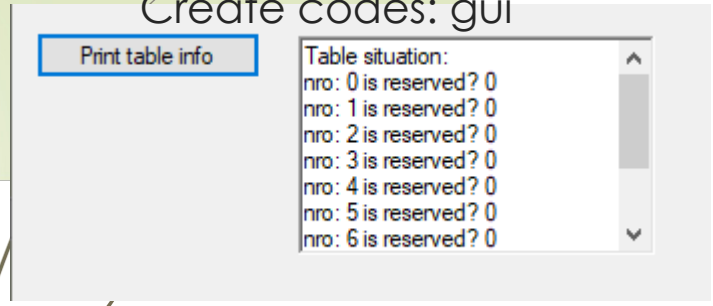
Create codes: gui




```
Restaurant rest;
private void button2_Click(object sender, EventArgs e)
{
    String n = textBox2.Text;
    rest = new Restaurant(n);
    label3.Text = "Restaurant created and with 10 free tables!";
}
```

Visual Studio: OOP part 2

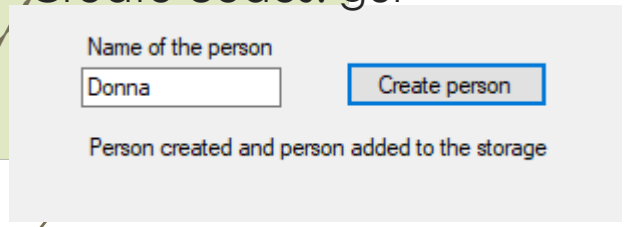
Create codes: gui




```
private void button3_Click(object sender, EventArgs e)
{
    richTextBox1.Text = rest.getTableInfo();
}
```

Visual Studio: OOP part 2

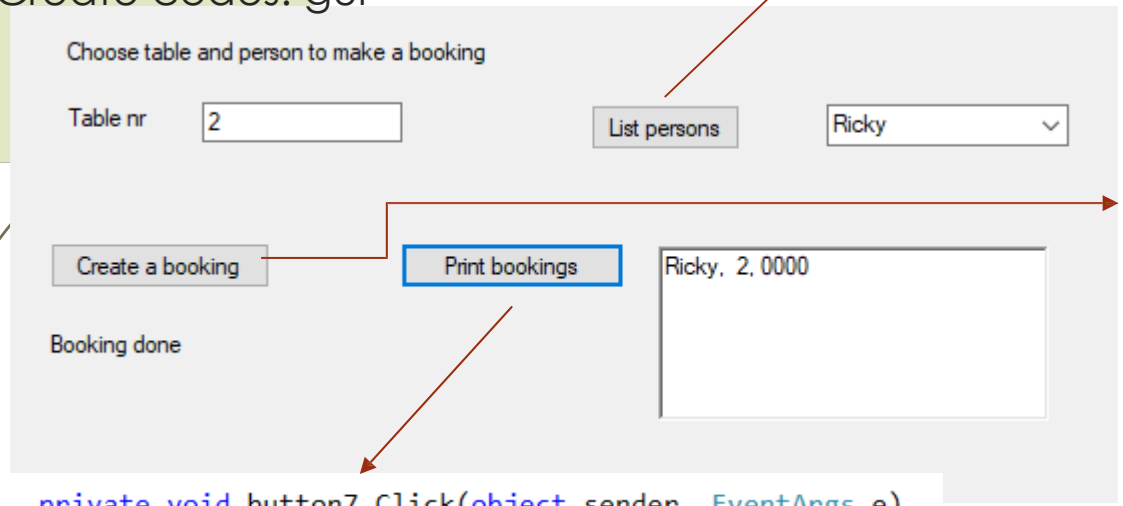
Create codes: gui




```
Person p;
private void button1_Click(object sender, EventArgs e)
{
    String n = textBox1.Text;
    p = new Person(n);
   String addInfo =  rest.addPersons(p);

    label2.Text = "Person created and " + addInfo;
}
```

Visual Studio: OOP part 2

Create codes: gui



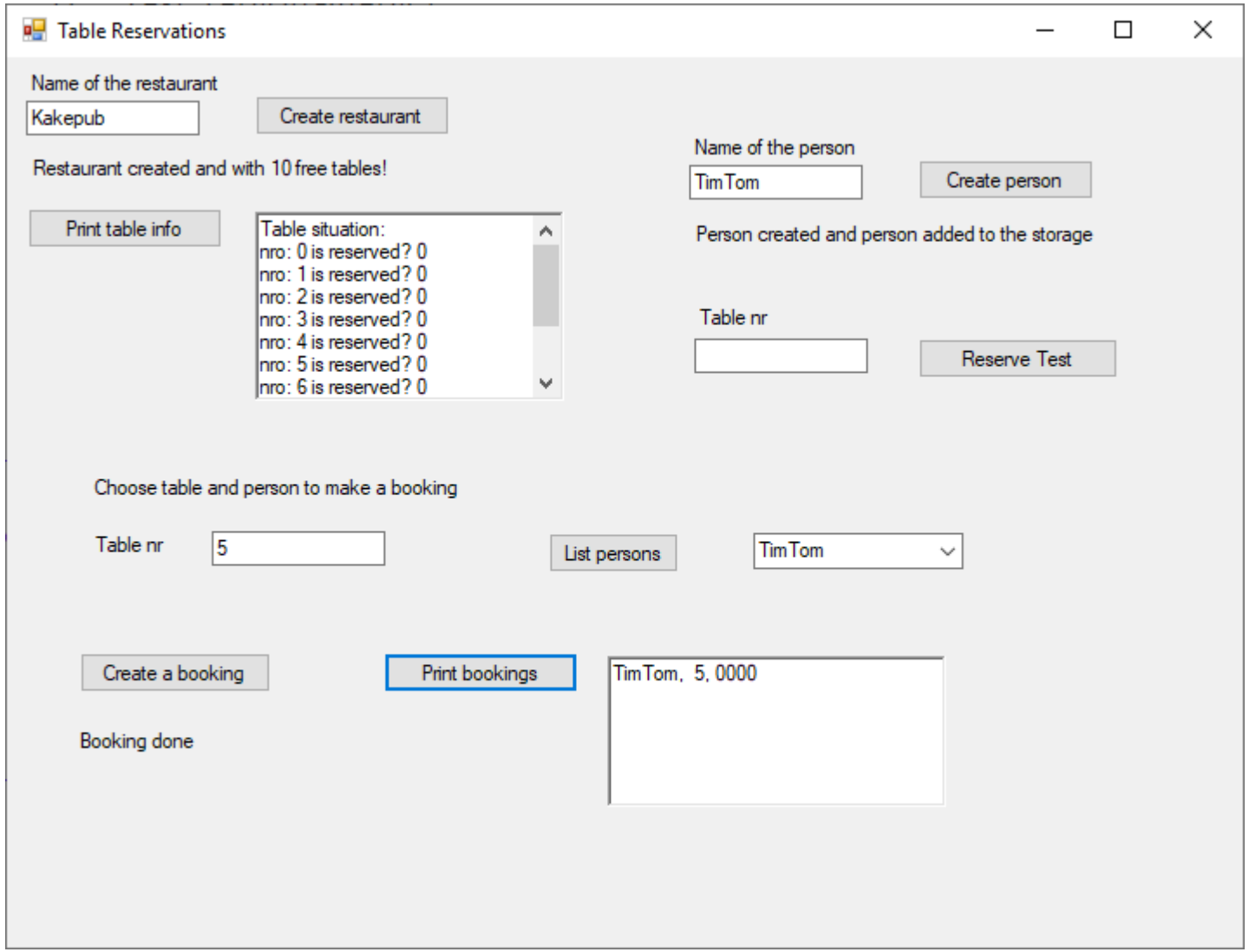

```
private void button5_Click(object sender, EventArgs e)
{
    foreach (Person p in rest.persons)
    {
        String name = p.getName();
        comboBox1.Items.Add(name);
    }
}
```

```
Booking bb;
Person pp;
Table tt;
private void button6_Click(object sender, EventArgs e)
{
    String personName = comboBox1.Text;
    String tableNr = textBox4.Text;
    int nr = Convert.ToInt16(tableNr);

    String ok = rest.reserveTable(nr);
    if (ok.Equals("Table can not be reserved!"))
        label8.Text = "Table can not be reserved!";
    else
    {
        tt = rest.returnTable(nr);
        pp = rest.getPerson(personName);

        bb = new Booking();
        bb.makeBooking(pp, tt);

        label8.Text = rest.addBooking(bb);

    }


}
```

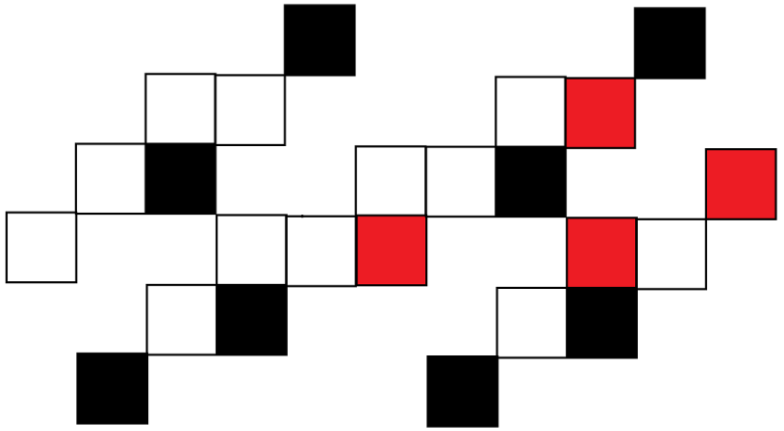
```
private void button7_Click(object sender, EventArgs e)
{
    foreach (Booking b in rest.bookings)
    {
        String info = b.getBookingInfo();
        richTextBox2.AppendText(info);
    }
}
```

Visual Studio: OOP part 2

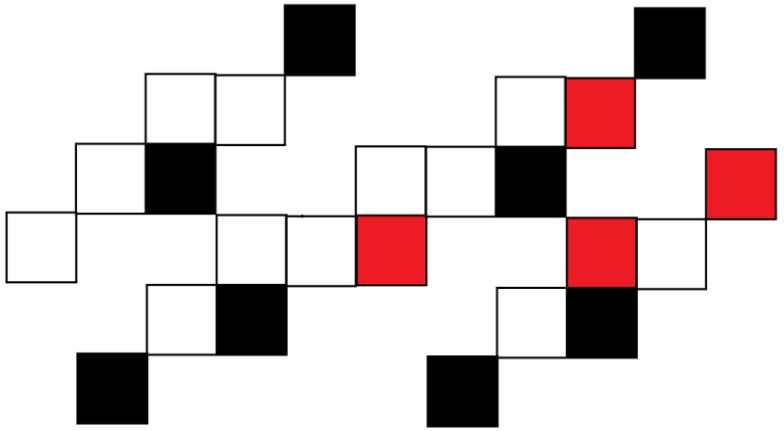
Try it! Make it better!



Visual Studio: OOP part 3



Visual Studio: OOP part 3
BlackJack



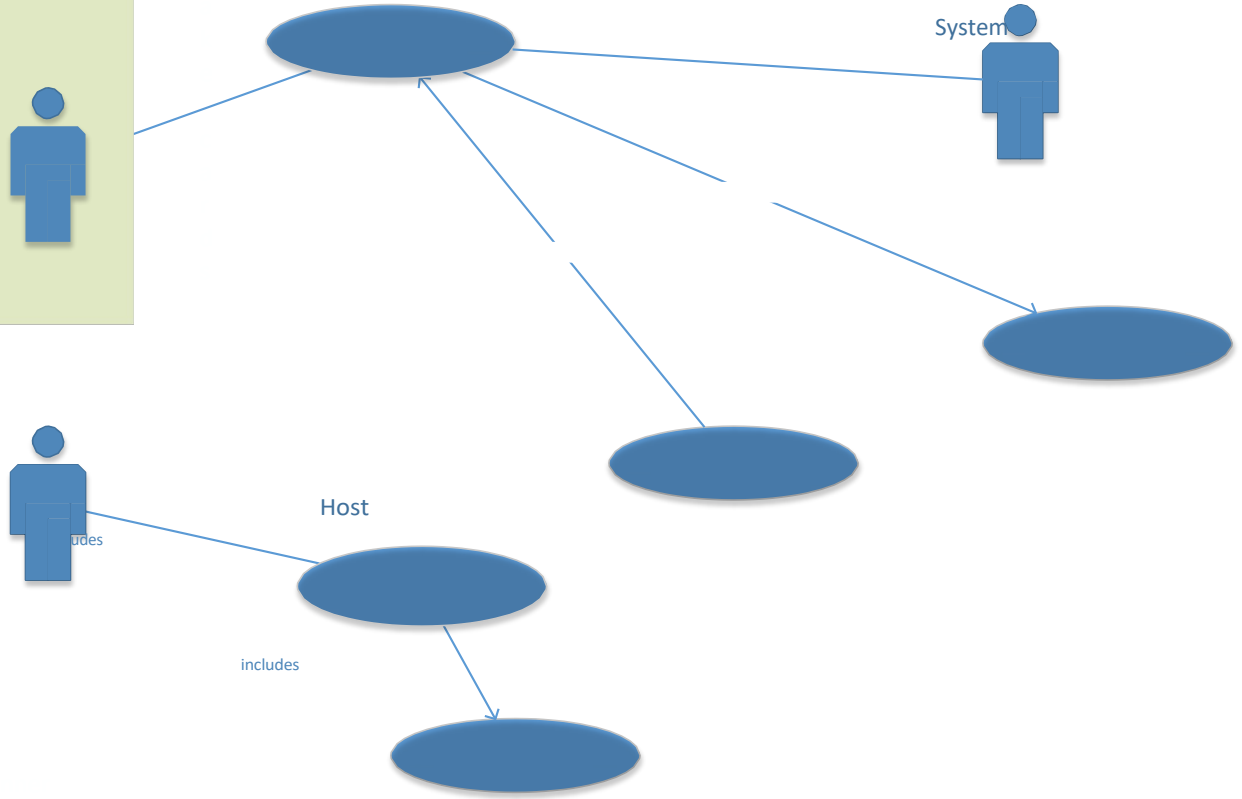
BlackJack Card Game

Introduction

Equally well known as Twenty-One. **T**he 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.

Use case diagram



Player

Visual Studio: OOP part 3 Use case example definition

Use case documentation	
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 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. 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.

Visual Studio: OOP part 3 Basic level design

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 ar class candidates:

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

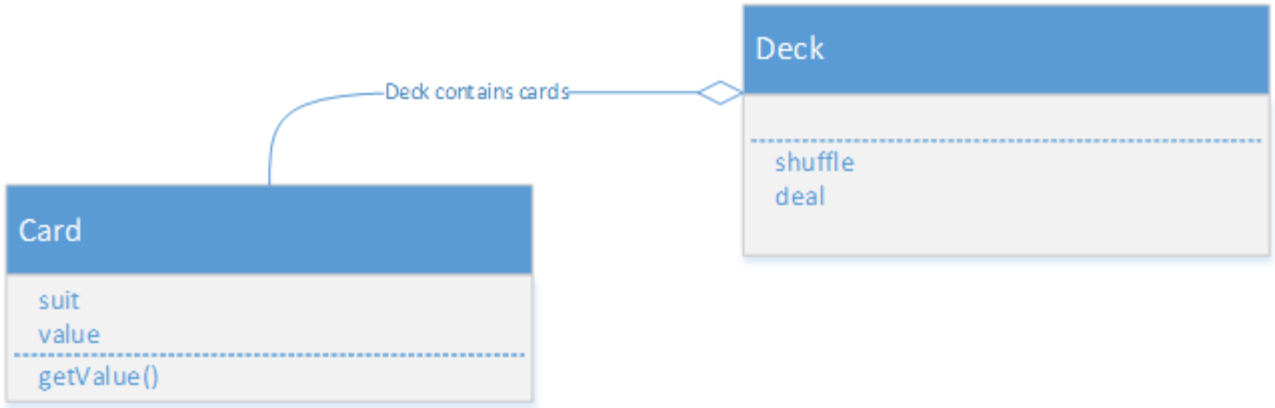
Shown on a form. Is points are 21, player is winner and message is shown.

If points are over 21 host has won.

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

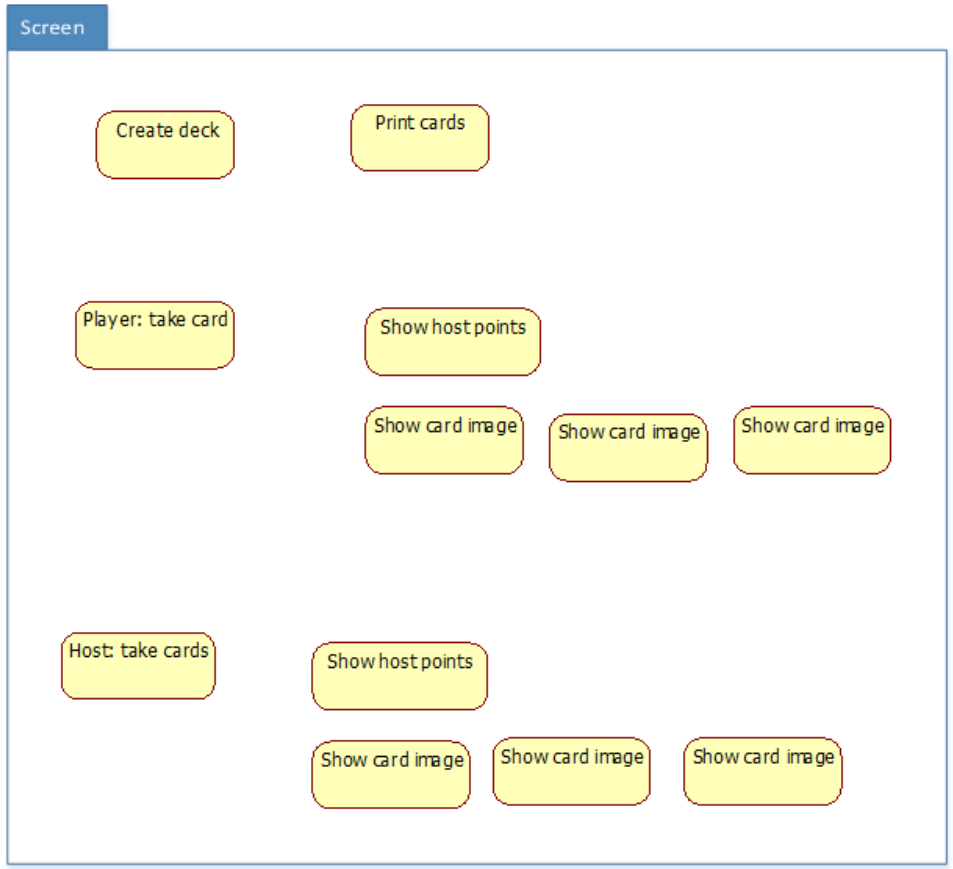
Visual Studio: OOP part 3 Basic level design

Class diagram



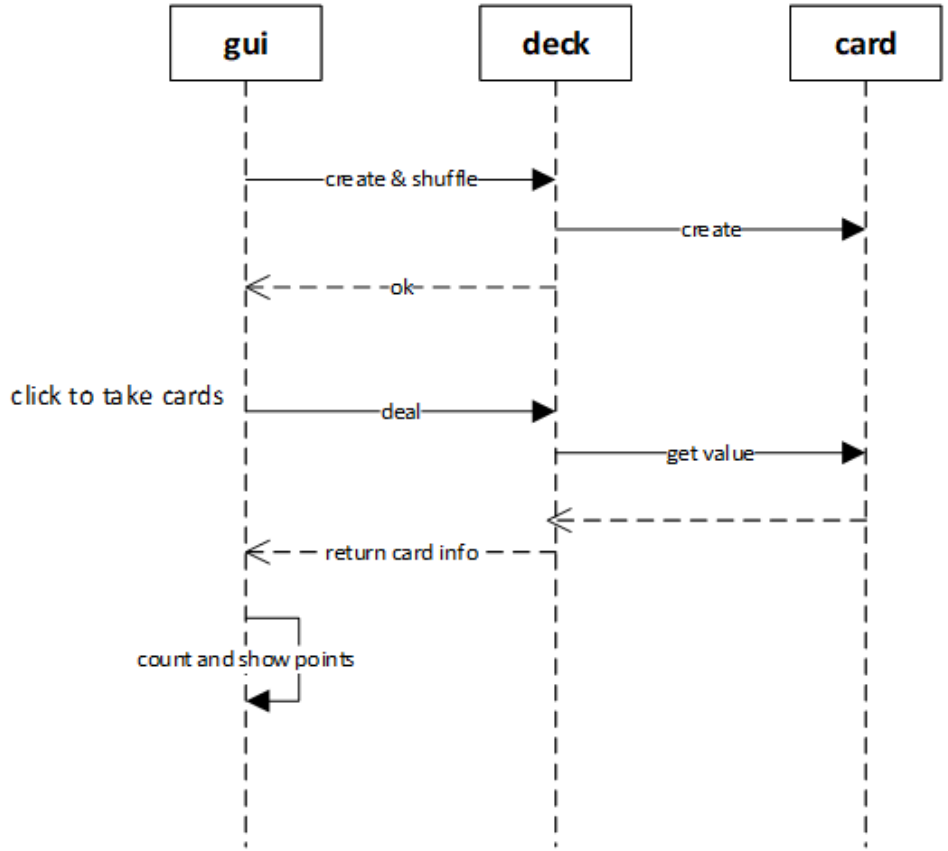
Visual Studio: OOP part 3 Basic level design

Gui prototype

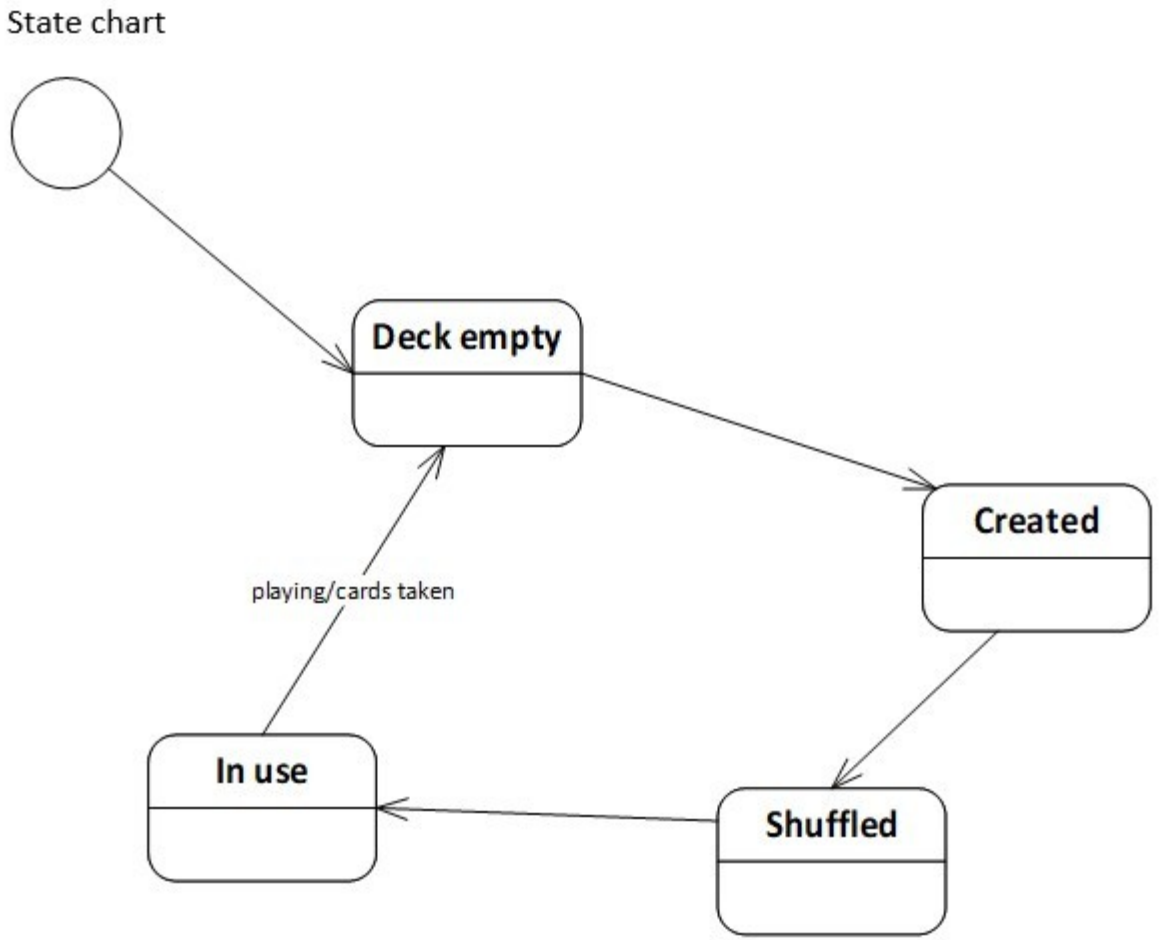


Visual Studio: OOP part 3 Basic level design

Seguence diagram

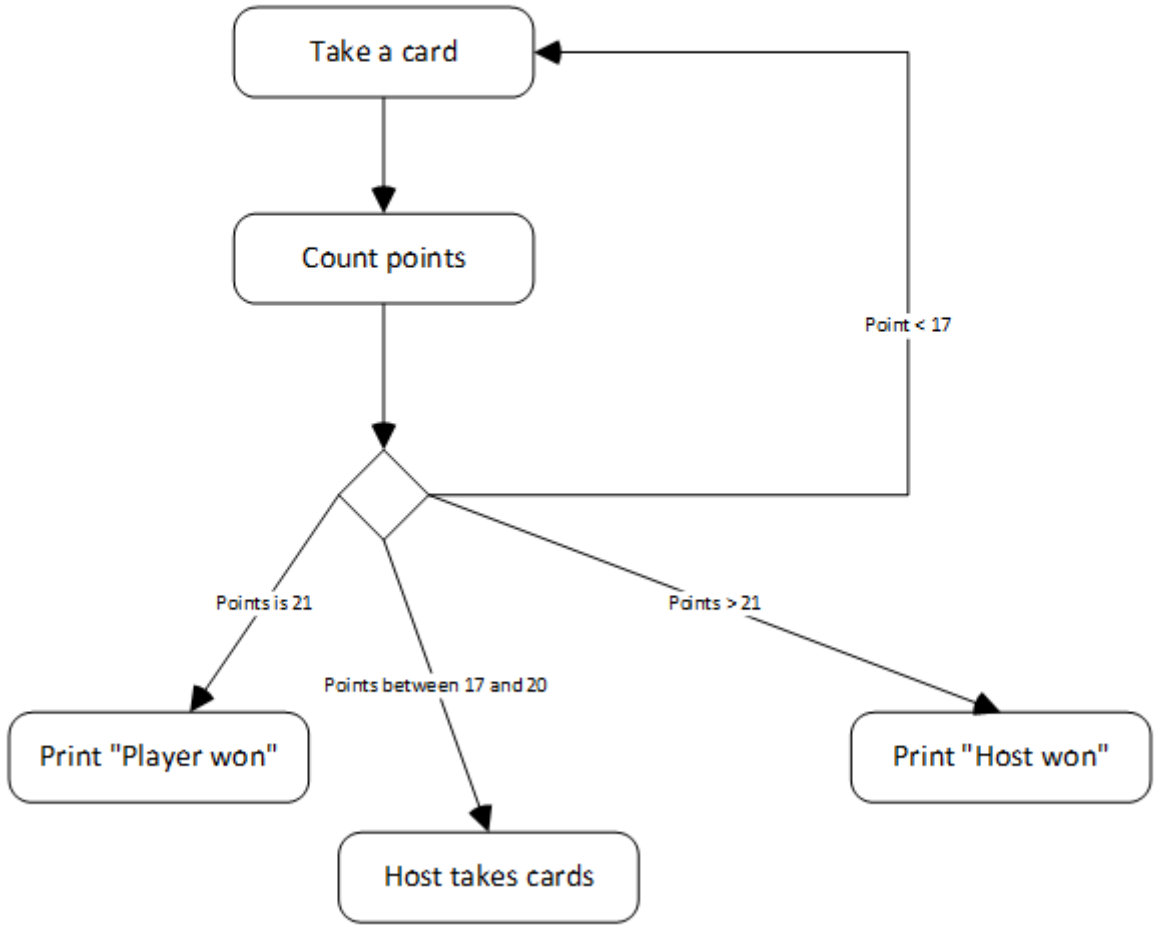


Visual Studio: OOP part 3 Basic level design



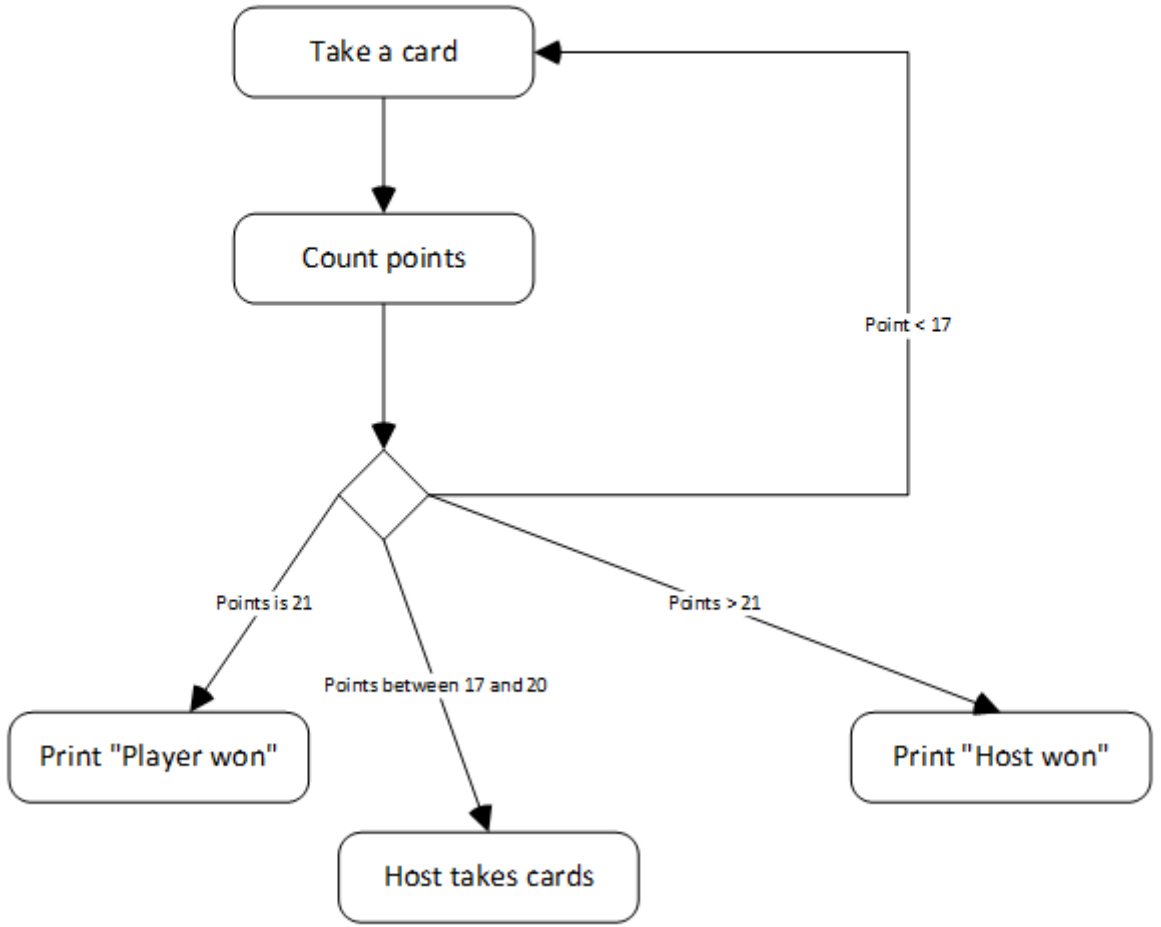
Visual Studio: OOP part 3 Basic level design

Activity diagram

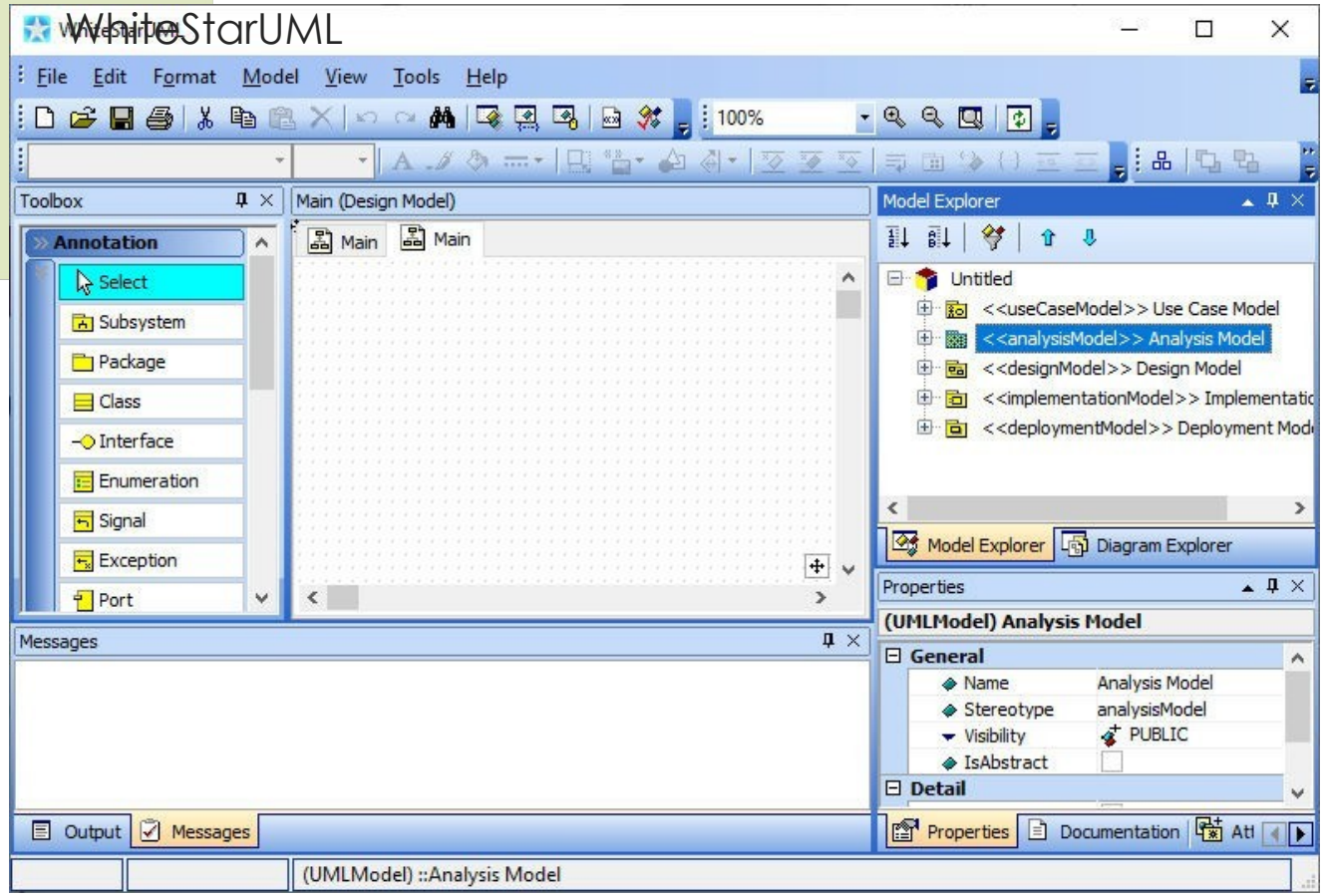


Visual Studio: OOP part 3 Basic level design

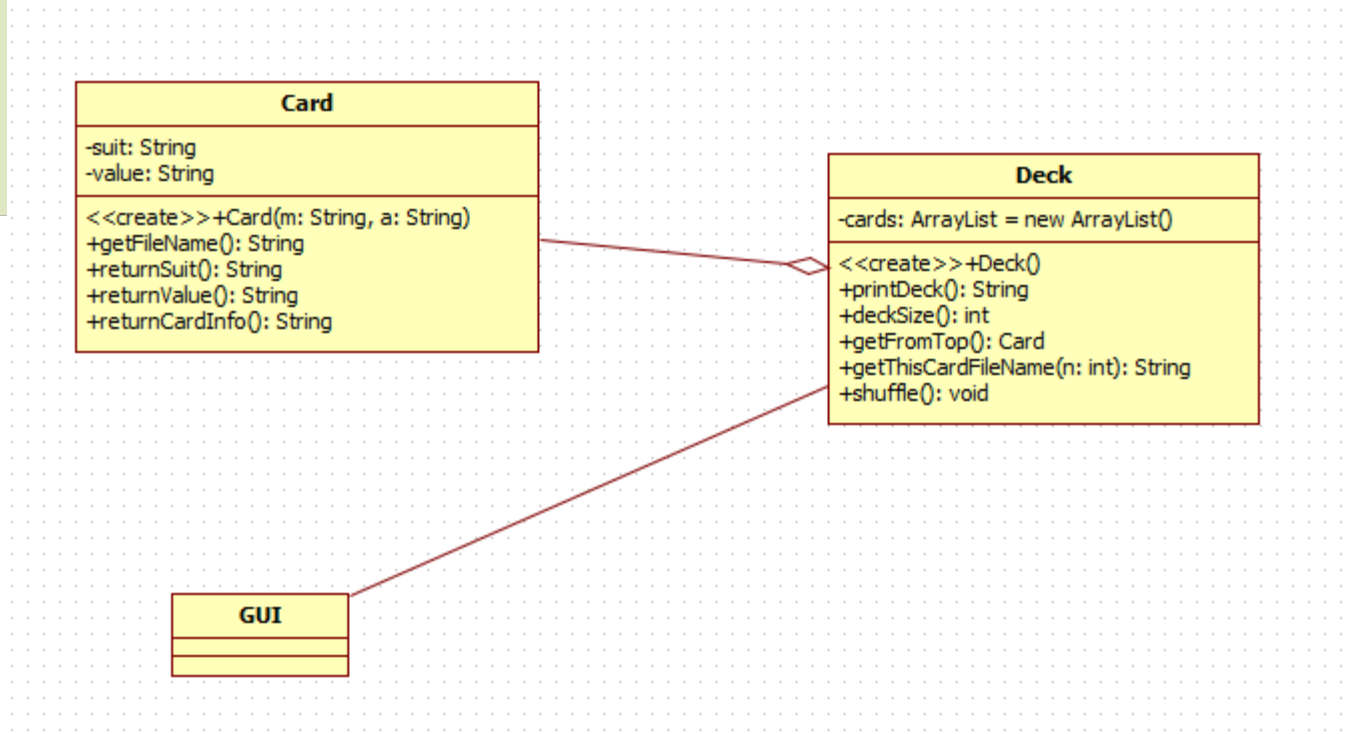
Activity diagram



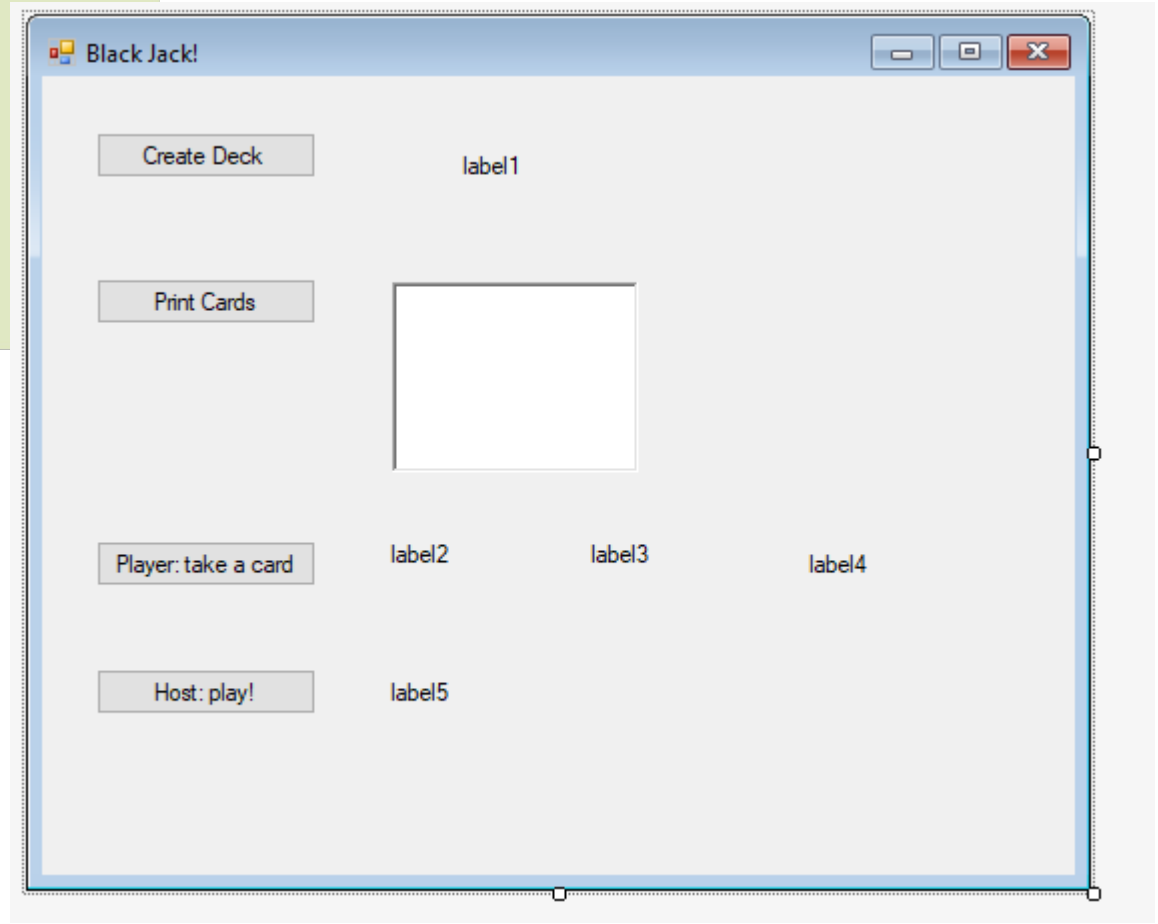
Visual Studio:
OOP part 3 Tool
for modelling:
WhiteStarUML



Visual Studio: OOP part 3
Tool for modelling:
WhiteStarUML

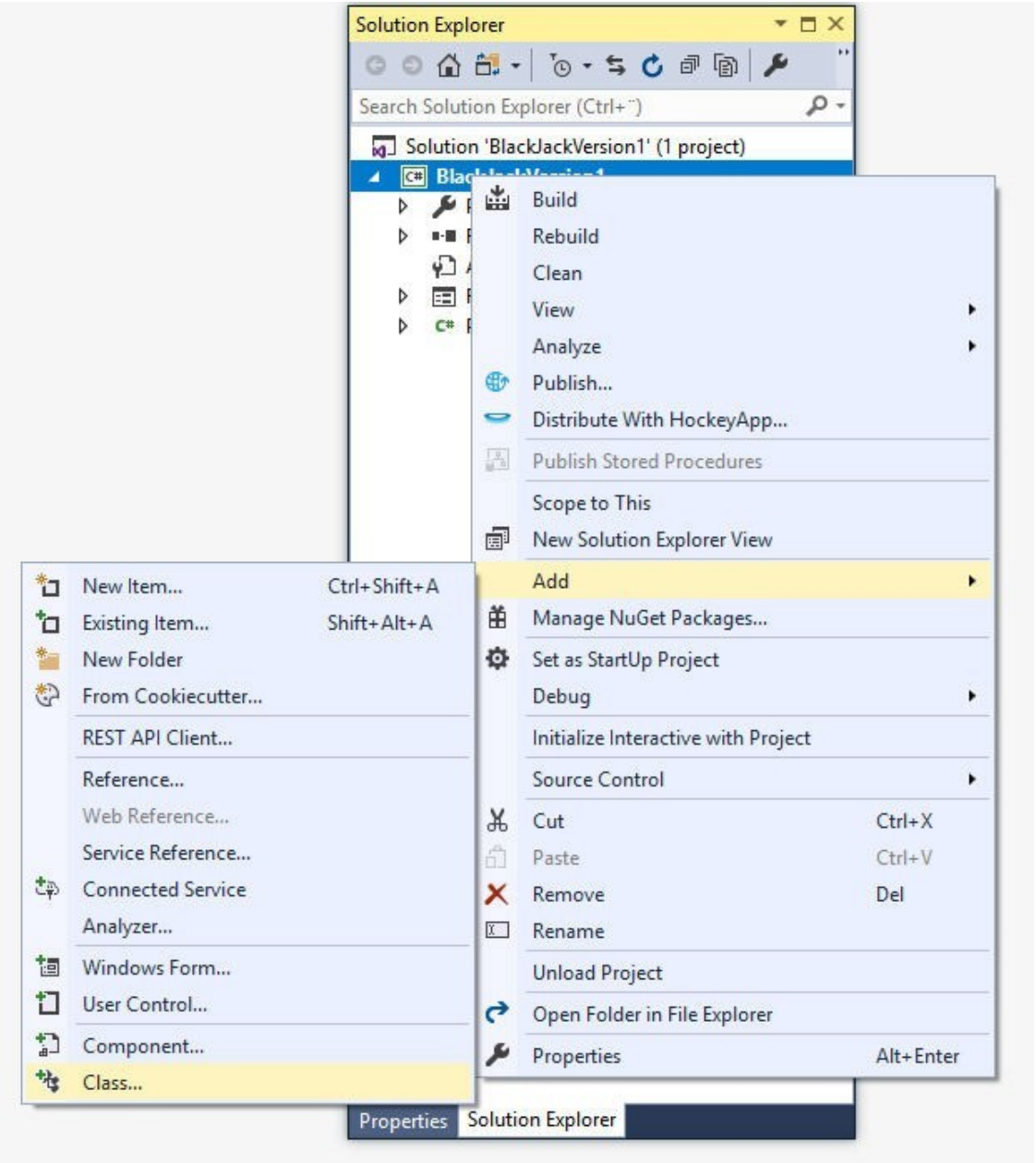


Visual Studio: OOP part 3
Create a new Visual Studio Project with this kind of gui



Visual Studio: OOP part 3
Create codes

Add classes via solution explorer if class frames have been created by modelling tool



Visual Studio: OOP part 3

Classes: Card

```
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;
    }
}
```

Visual Studio: OOP part 3

Classes: Deck: creating

```
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++;
              }
      }
```

Visual Studio: OOP part 3

Classes: Deck: printing

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

Visual Studio: OOP part 3

Classes: Deck: shuffling

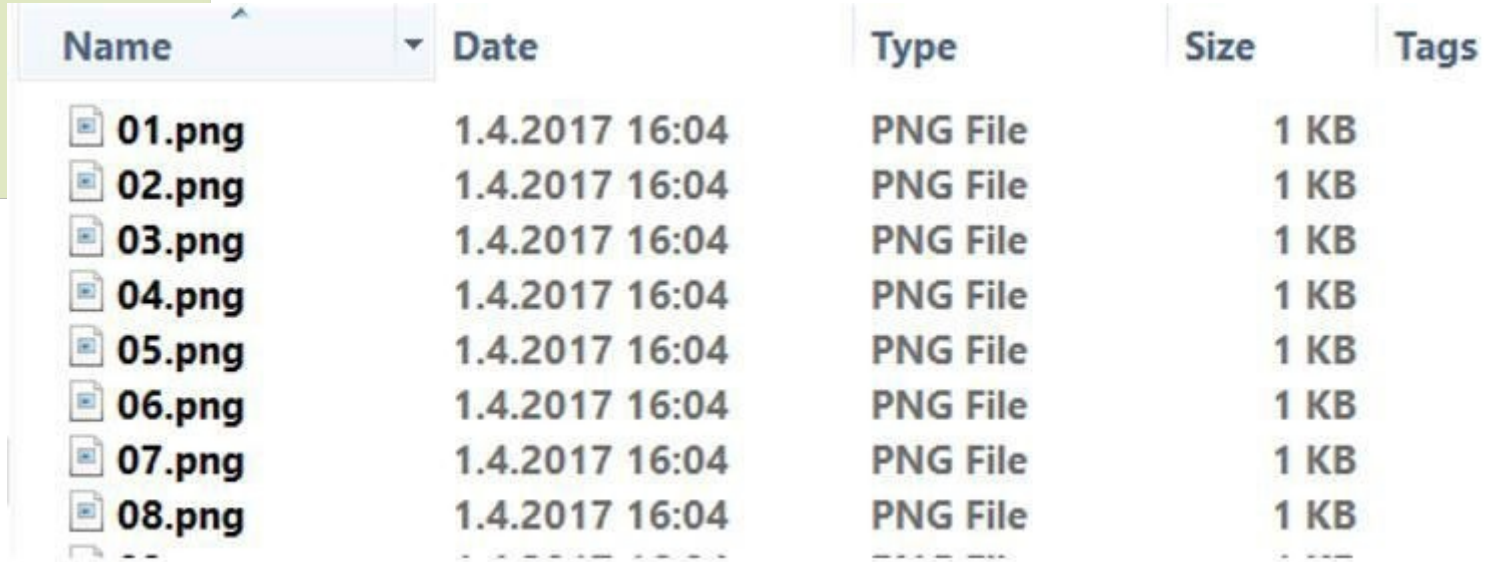
```
public void shuffle()
{
    Random rr = new Random();
    for (int i = 0; i < 1000; i++)
    {
        int x = rr.Next(0, 52);
        int y = rr.Next(0, 52);
        Card temp = (Card)cards[x];
        cards[x] = (Card)cards[y];
        cards[y] = temp;
    }
}
```

Visual Studio: OOP part 3

Classes: Deck: getting data

```
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();
 }
```

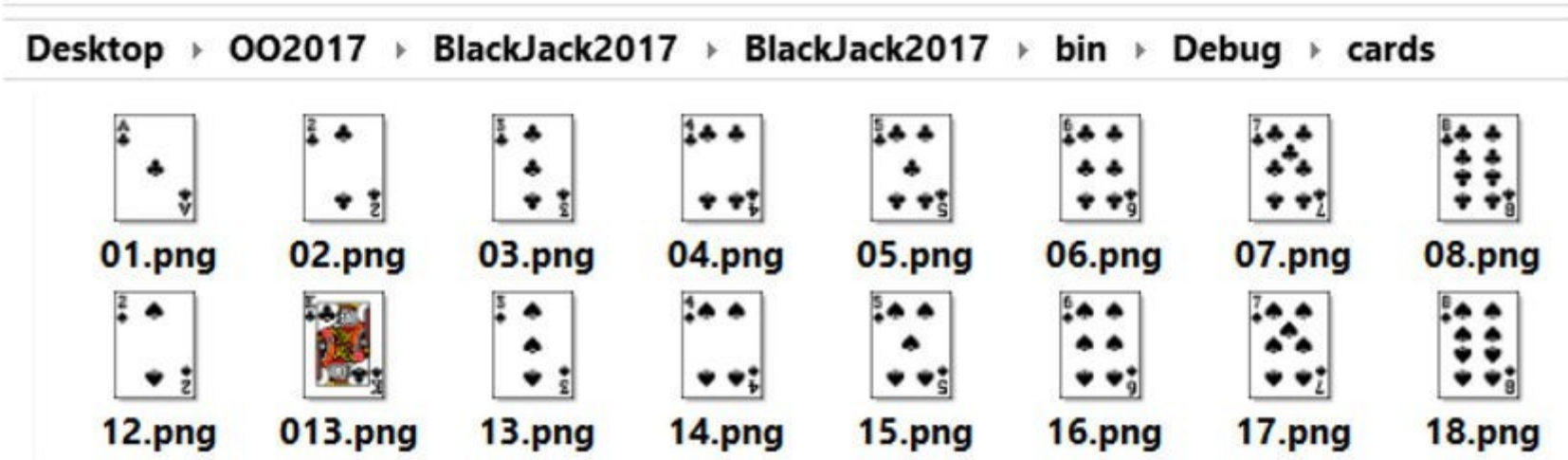
Visual Studio: OOP part 3:Card images



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	

Visual Studio: OOP part 3:Card images

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



Visual Studio: OOP part 3

Special features

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;
 }
```

Visual Studio: OOP part 3

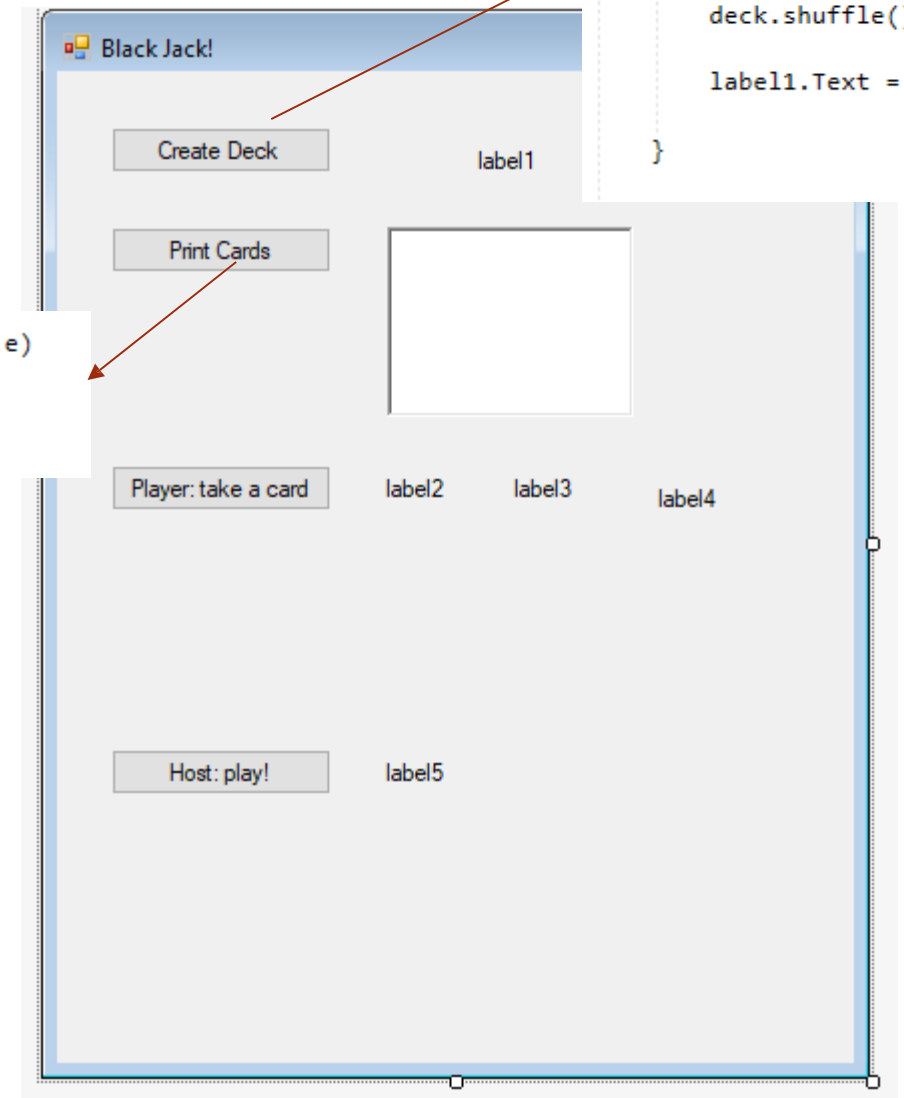
Special features

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);

      }
```

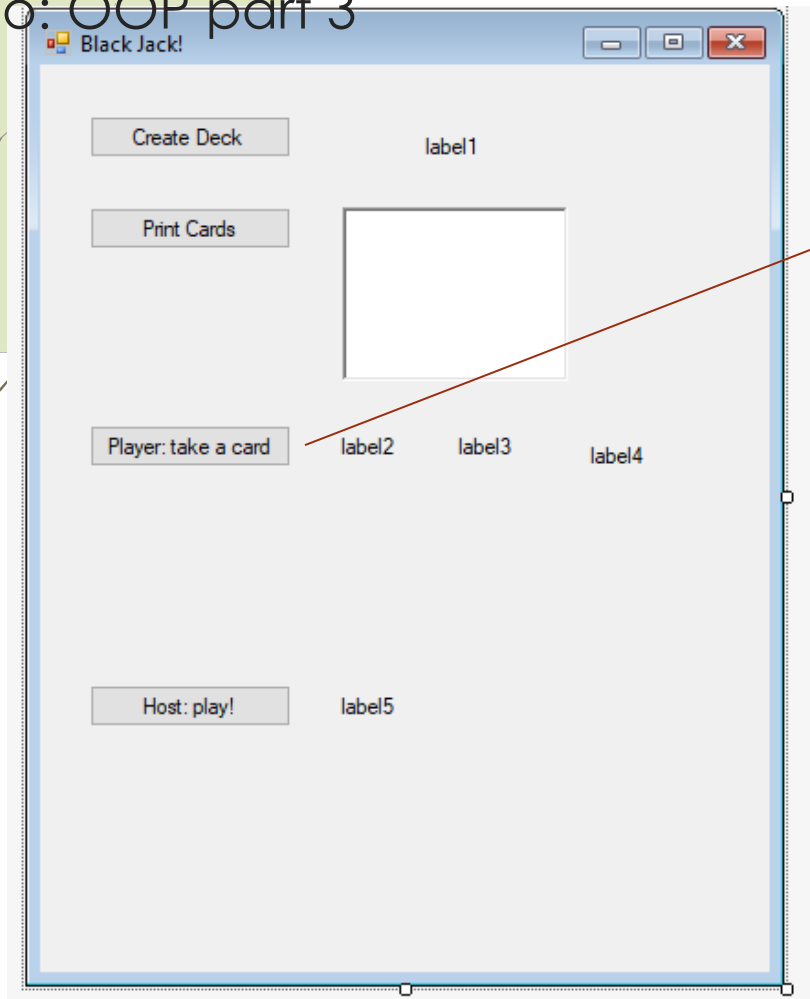
Visual Studio: OOP part 3

```
Deck deck;
private void button1_Click(object sender, EventArgs e)
{

    deck = new Deck();
    deck.shuffle();

    label1.Text = "Deck ok";

}
```



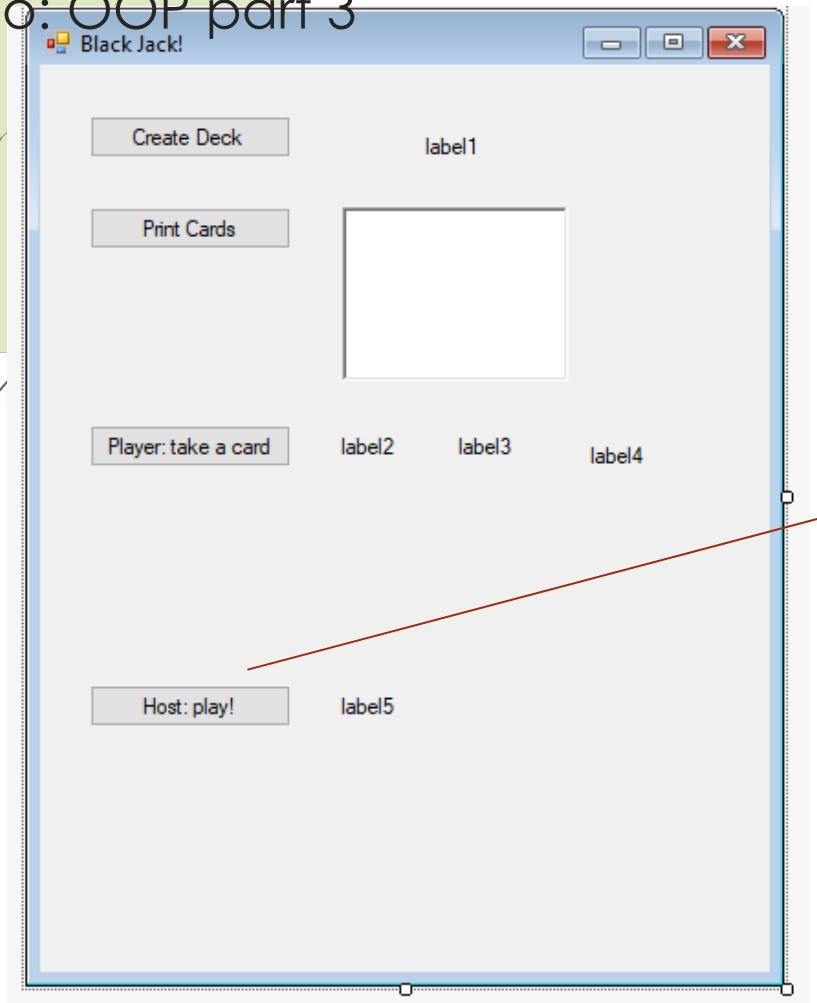

```
private void button2_Click(object sender, EventArgs e)
{
    richTextBox1.Text = deck.printDeck();
}
```

Visual Studio: OOP part 3



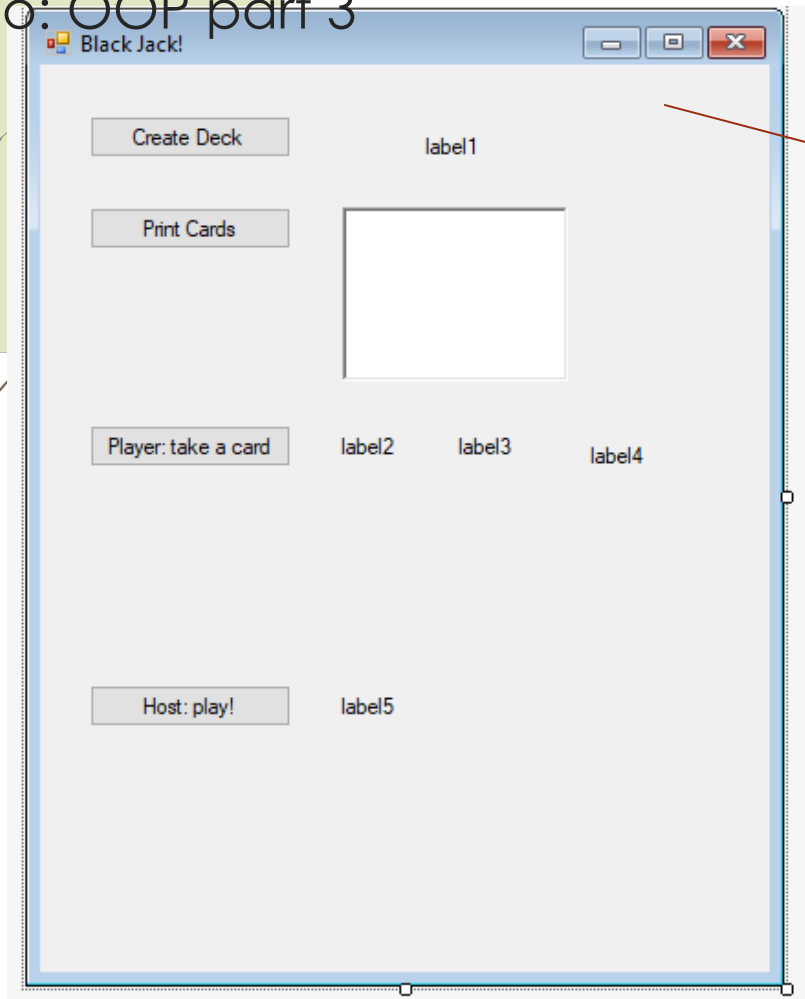

```
String path = "cards\\";
int points1 = 0;
int count1 = -1;
private void button3_Click(object sender, EventArgs e)
{
    count1++;
    Card taken = null;
    try
    {
        taken = deck.getFromTop();
    }
    catch (Exception ex)
    {
        label3.Text = "deck is empty!!  " + ex.Message;
    }
    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++;
}
```

Visual Studio: OOP part 3




```
int count2;
int points2 = 0;
int place = 0;
private void button4_Click(object sender, EventArgs e)
{
    place++;
    count2++;
    Card taken = deck.getFromTop();
    points2 += Convert.ToInt16(taken.returnValue());
    label5.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)
        label5.Text = saveToFile(0);
    else if (points2 > 21)
        label5.Text = saveToFile(1);
}
```

Visual Studio: OOP part 3




```
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;
    }
}
```

Visual Studio: OOP part 3: saving to file

```
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);

}
```

Visual Studio: OOP part 3

Try it! Make it better!

What can be added to the next version?

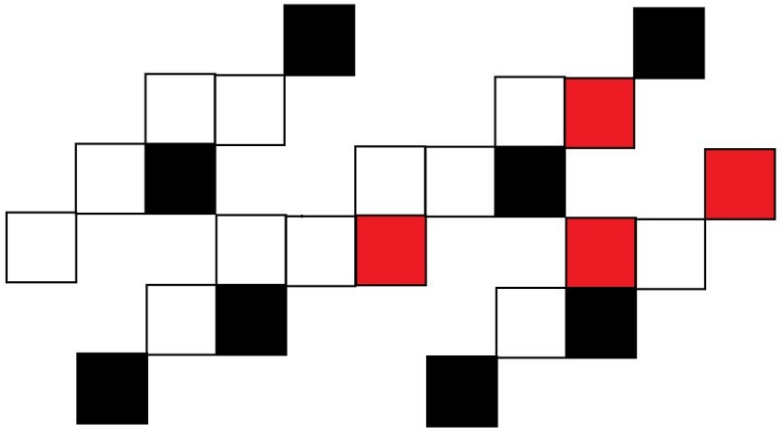
Betting

Starting a new round

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

And so on.

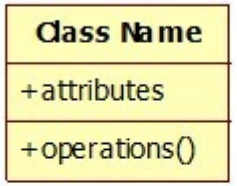
Visual Studio: OOP part 5



Visual Studio: OOP part 5

Class Diagrams

UML notation



Access spesifiers (visibility of attributes and operations)

private -

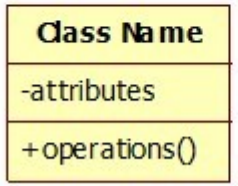
public +

protected #

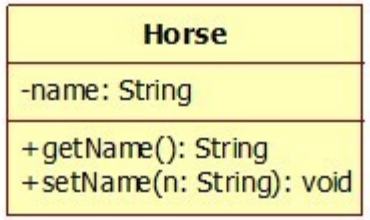
Visual Studio: OOP part 5

Class Diagrams

The main rule:
Attributes are private
Operations are public



Example: Horse



Visual Studio: OOP part 5

Class Diagrams: relationships between objects

Association

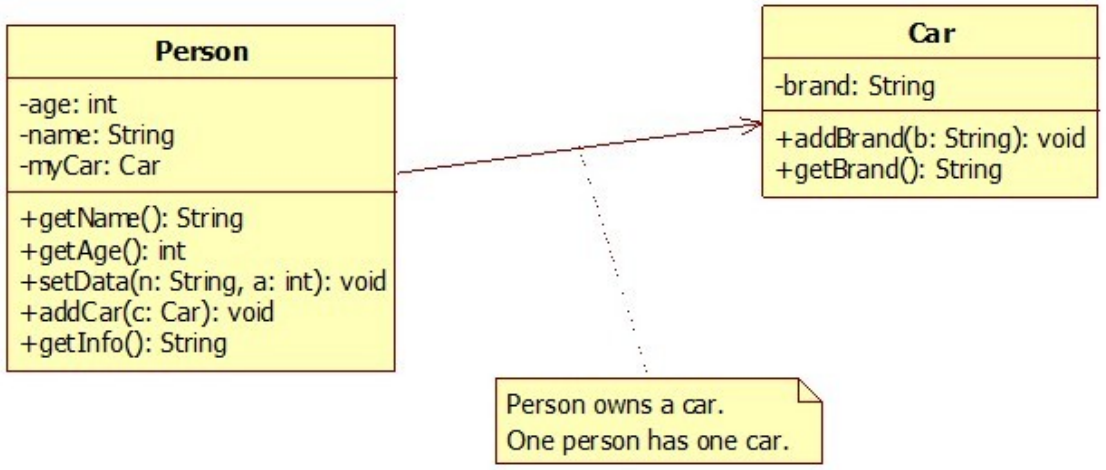
Objects communicate with each other.

They need each other, they need know each other.

Example

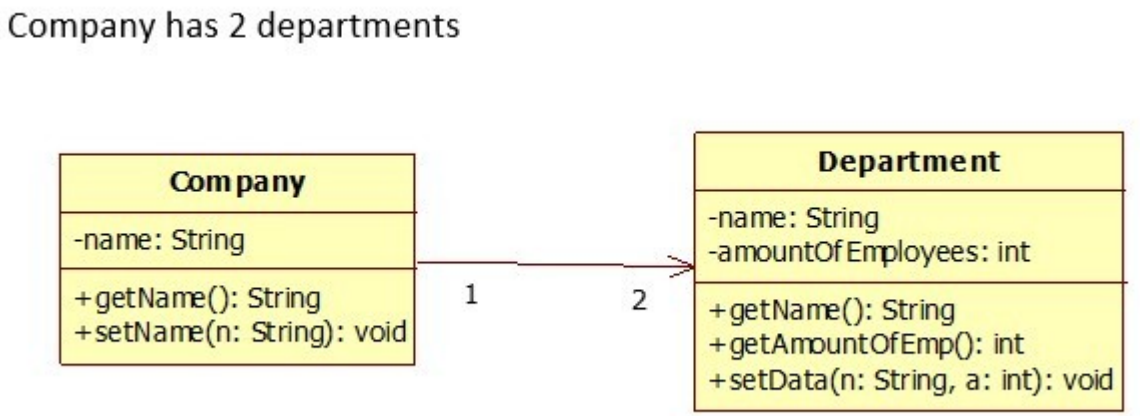
Person owns a car.

With UML:



Visual Studio: OOP part 5

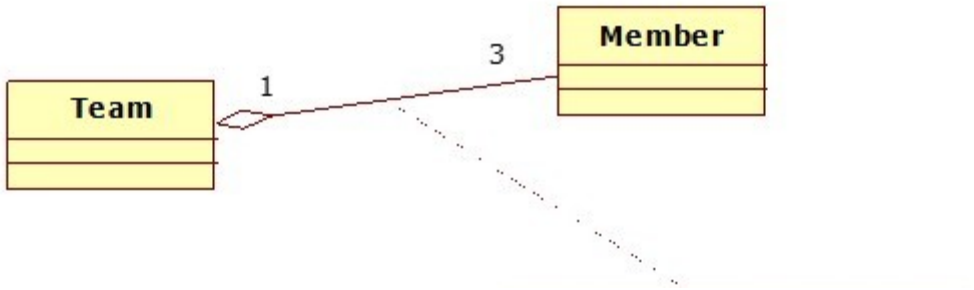
Class Diagrams: relationships between objects



Visual Studio: OOP part 5

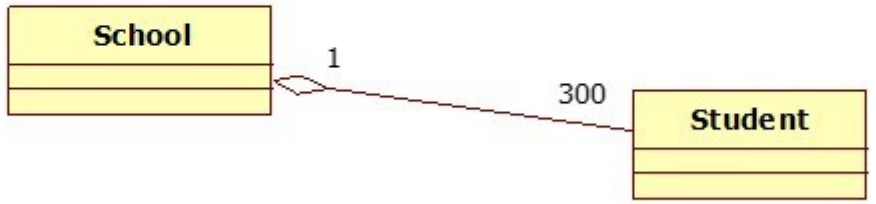
Class Diagrams: relationships between objects

Team has 3 members



Visual Studio: OOP part 5

School has 300 students



Visual Studio: OOP part 5

Class Diagrams: relationships between objects

Composition
"Is part of"/"Contains"

House has 3 rooms.

When the house is created, all the rooms are created at the same.

Visual Studio: OOP part 5

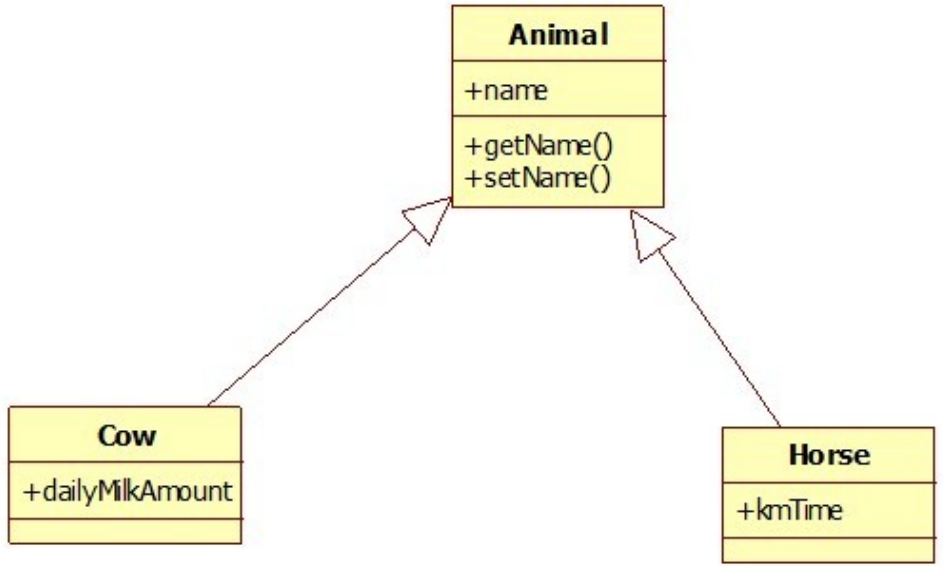
Inheritance, generalization

"Is"

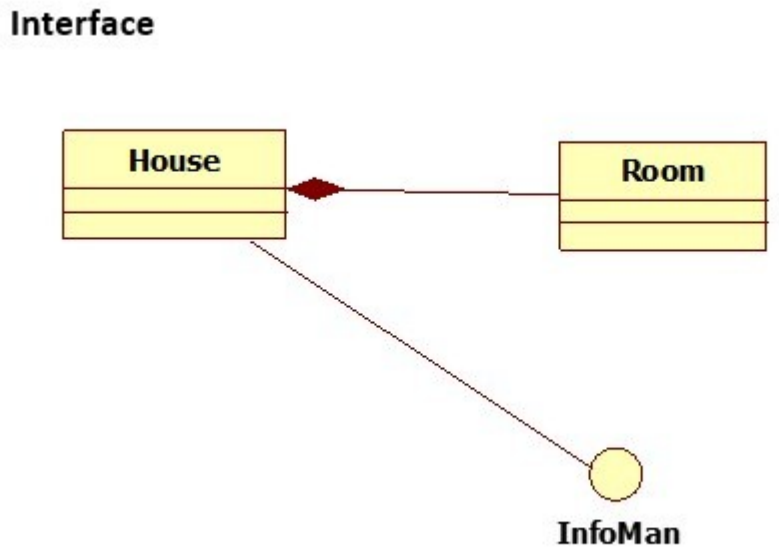
base class (mother class)

sub class (child class)

Cow and horse are animals.

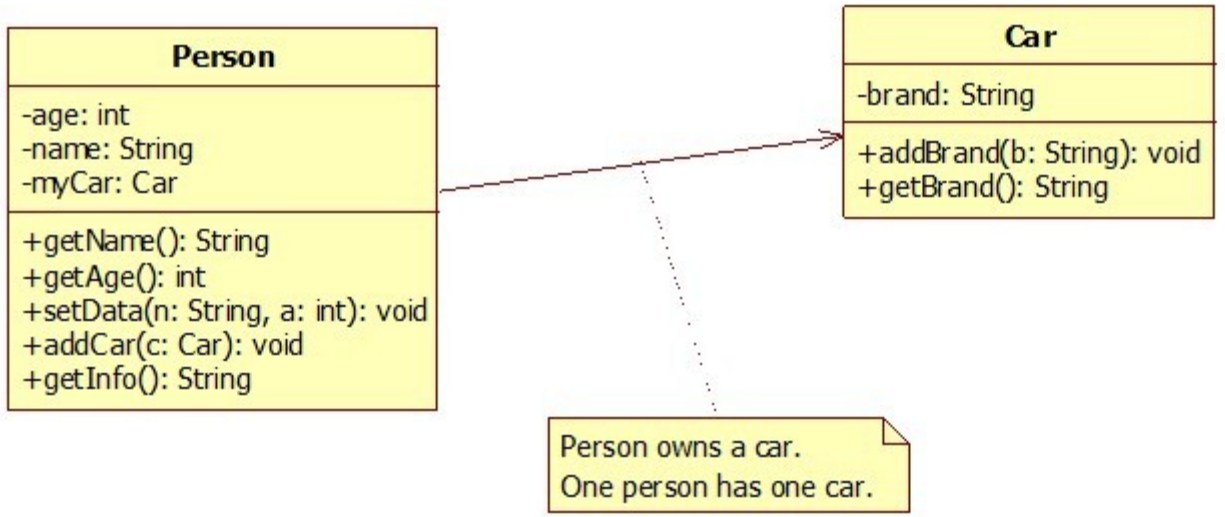


Visual Studio: OOP part 5



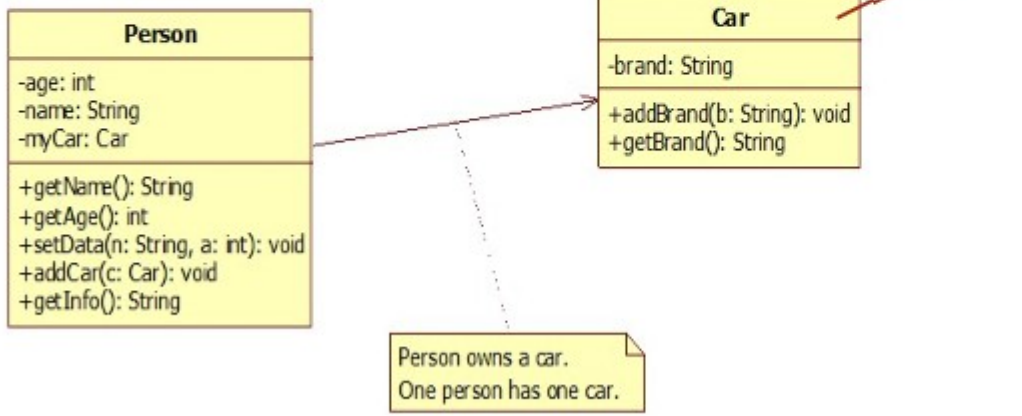
Visual Studio: OOP part 5

Coded examples

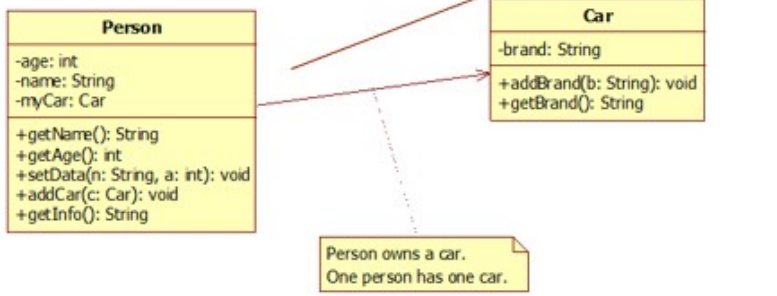


Visual Studio: OOP part 5

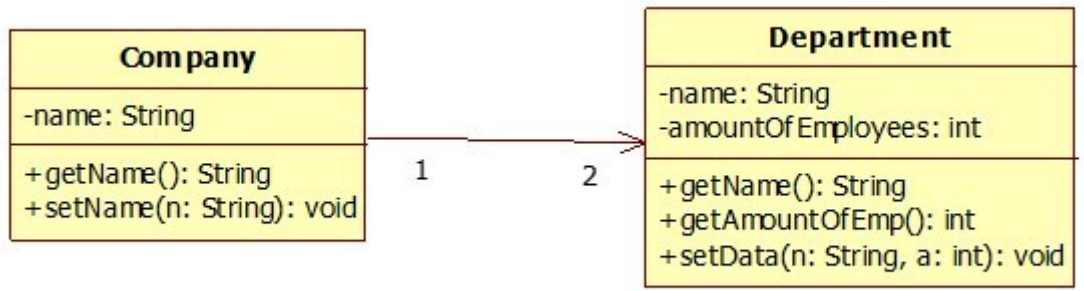
Coded examples




```
class Car
   {
        private String brand;

        public void addBrand(String b)
        {
            brand = b;
        }

        public String getBrand()
        {
            return brand;
        }
   }
```

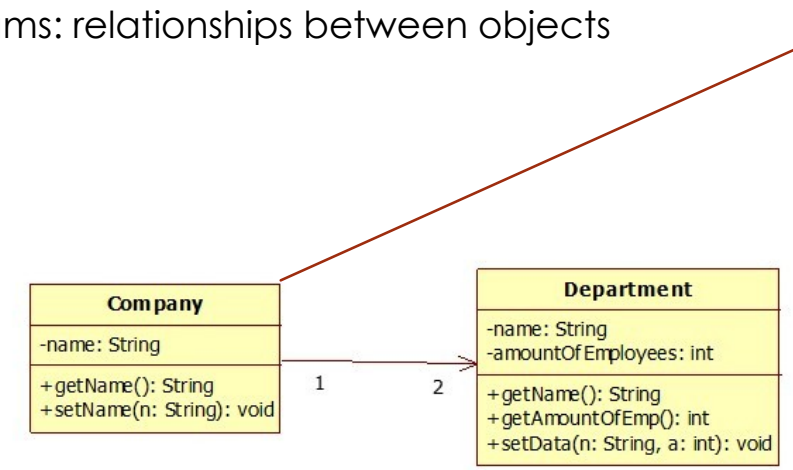



```java
class Person
 {
     private int age;
     private String name;
     private Car myCar;
     public void addCar(Car c)
     {
         myCar = c;
     }
     public String getInfo()
     {
         String info = "Name is " + name;
         info = info + " and age is " + age;
         info = info + " and person's car is " + myCar.getBrand();
         return info;
     }
     public String getName()
     {
         return name;
     }
     public int getAge()
     {
         return age;
     }
     public void setData(String n, int a)
     {
         name = n;
         age = a;
     }
 }
```


Company
-name: String
+getName(): String
+setName(n: String): void
1
2
Department
-name: String
-amountOfEmployees: int
+getName(): String
+getAmountOfEmp(): int
+setData(n: String, a: int): void

Visual Studio: OOP part 5

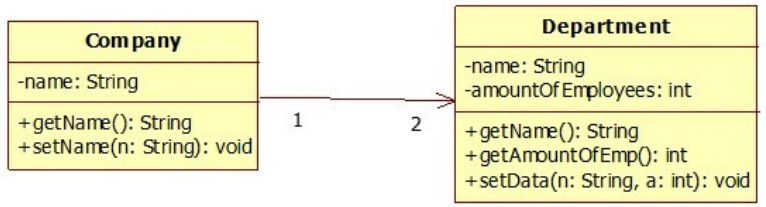
Class Diagrams: relationships between objects




```
class Company
    {
        private String name;
        public String getName()
        {
            return name;
        }
        public void setName(String n)
        {
            name = n;
        }

}
```

Visual Studio: OOP part 5

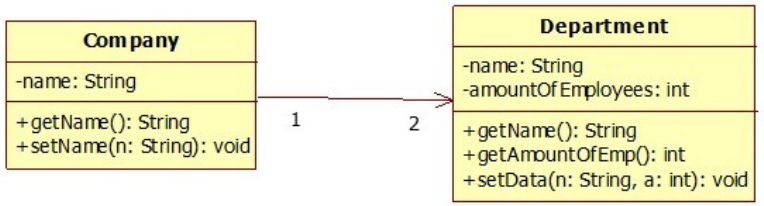
Class Diagrams: relationships between objects




```
class Department
    {
        private String name;
        private int amountOfEmployees;
        public String getName()
        {
            return name;
        }
        public int getAmountOfEmp()
        {
            return amountOfEmployees;
        }
        public void setData(String n, int a)
        {
            name = n;
            amountOfEmployees = a;
        }
}
```

Visual Studio: OOP part 5

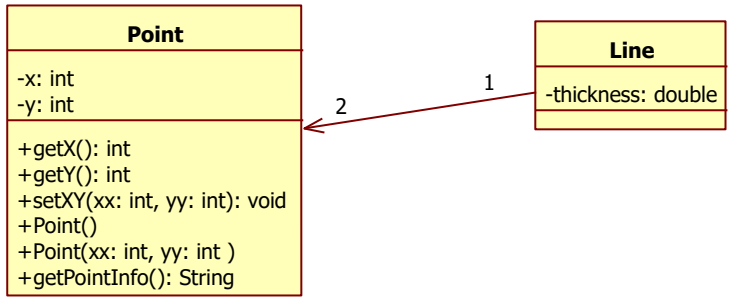
Class Diagrams: relationships between objects
Coded examples

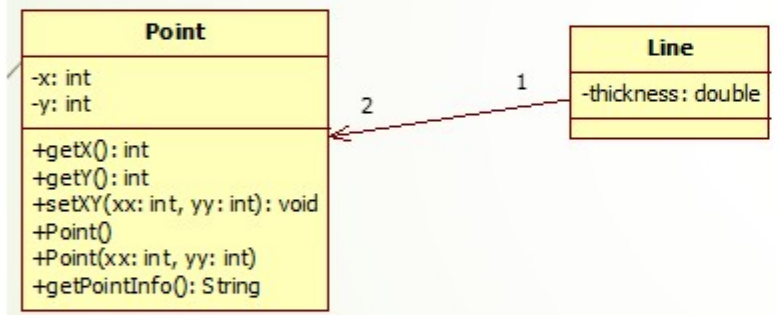



```
class Company
    {
        private String name;
        private Department dep1;
        private Department dep2;
        public void addDepartmentsToCompany(Department d1, Department d2)
        {
            dep1 = d1;
            dep2 = d2;
        }
        public String getCompanyInfo()
        {
            String info = "Company's name is " + name;
            info += "and it has departments " + dep1.getName();
            info += " and " + dep2.getName();
            return info;
        }
        public String getName()
        {
            return name;
        }
        public void setName(String n)
        {
            name = n;
        }
}
```

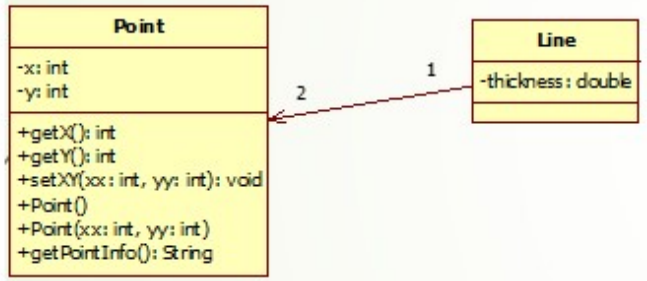
Visual Studio: OOP part 5

Class Diagrams: relationships
between objects
Coded examples



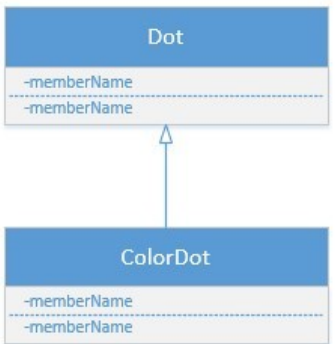



```java
class Point
    {
        private int x;
        private int y;
        public Point()
        {
            x = 0; y = 0;
        }
        public Point(int xx, int yy)
        {
            x = xx; y = yy;
        }
        public int getX()
        {
            return x;
        }
        public int getY()
        {
            return y;
        }
        public void setXY(int xx, int yy)
        {
            x = xx; y = yy;
        }
        public String getPointInfo()
        {
            String info = "(" + x + "," + y + ")";
            return info;
        }
    }
```




```
class Line
    {
        private Point startingPoint;
        private Point endingPoint;

        private int thickness;

        public void addPoints(Point p1, Point p2)
        {
            startingPoint = p1;
            endingPoint = p2;
        }
        public String getLineInfo()
        {
            String info = "Line's thickness is " + thickness;
            info = info + " and starting point is " + startingPoint.getPointInfo();
            info = info + " and ending point is " + endingPoint.getPointInfo();
            return info;
        }
        public void setThickness(int t)
        {
            thickness = t;
        }
        public int getThickness()
        {
            return thickness;
        }
    }
```

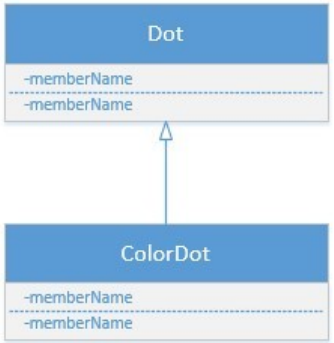
Coded examples



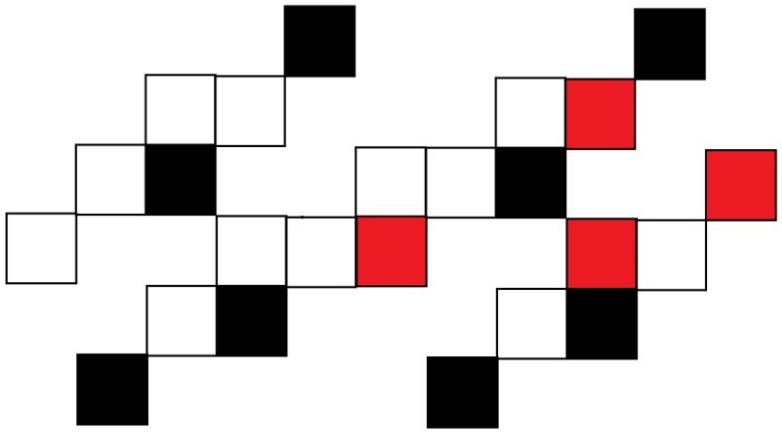

```
public class Dot
{
  protected double x;
  protected double y;
  public Dot()
  {
    x = 0;
    y = 0;
  }
  public Dot(double xx, double yy)
  {
    x = xx;
    y = yy;
  }
  public void setPoint(double xx, double yy)
  {
    x = xx;
    y = yy;
  }
  public double getX()
  {
     return x;
  }
  public double getY()
  {
     return y;
  }
 }
```

Visual Studio: OOP part 5

Class Diagrams: relationships between objects
Coded examples




```
public class ColorDot : Dot
{
    String color;
    public ColorDot()
    {
      color = "Unknown";
    }
    public ColorDot(String c)
    {
      color = c;
    }
    public ColorDot(String c, double xx, double yy)
    {
      x = xx;
      y = yy;
      color = c;
    }
    public String getInfo()
    {
      String temp = "Color = " + color + " and coordinates are: " + x + "," + y;
      return temp;
    }
}
```

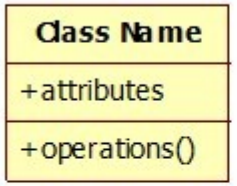


Visual Studio: OOP part 6

Visual Studio: OOP part 6

Class Diagrams

UML notation

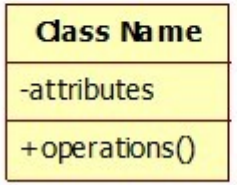


Access spesifiers (visibility of attributes and operations)
private -
public +
protected #

Visual Studio: OOP part 6

Class Diagrams

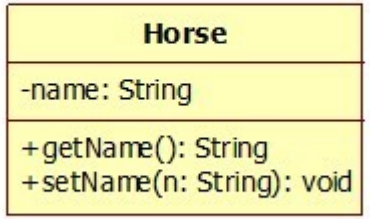
The main rule:
Attributes are private
Operations are public



Visual Studio: OOP part 6

Class Diagrams

Example: Horse



Visual Studio: OOP part 6

Class Diagrams: relationships between objects

Association

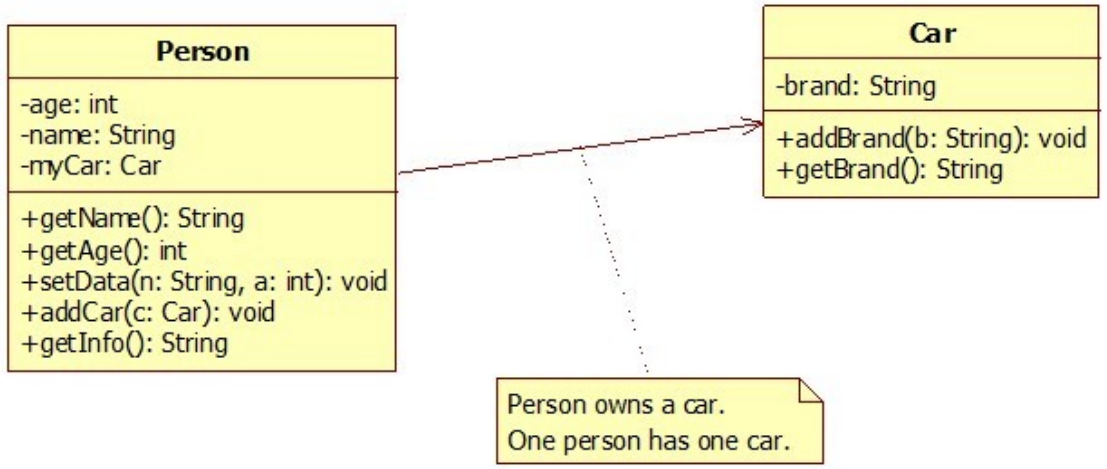
Objects communicate with each other.

They need each other, they need know each other.

Example

Person owns a car.

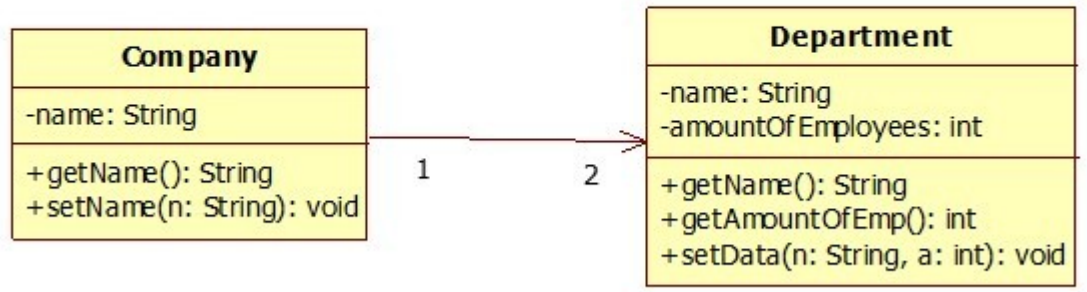
With UML:



Visual Studio: OOP part 6

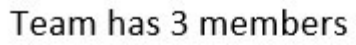
Class Diagrams: relationships between objects

Company has 2 departments



Visual Studio: OOP part 6

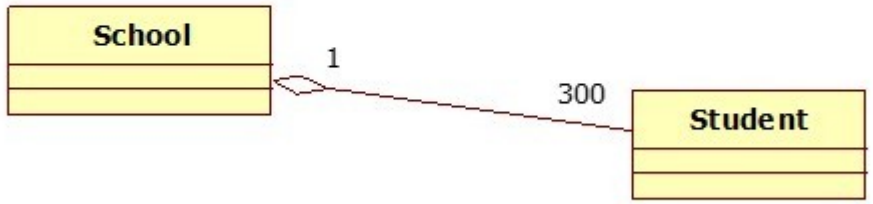
Class Diagrams: relationships between objects



Visual Studio: OOP part 5

Class Diagrams: relationships between objects

School has 300 students



Visual Studio: OOP part 6

Class Diagrams: relationships between objects

Composition
"Is part of"/"Contains"

Visual Studio: OOP part 6

Class Diagrams: relationships between objects

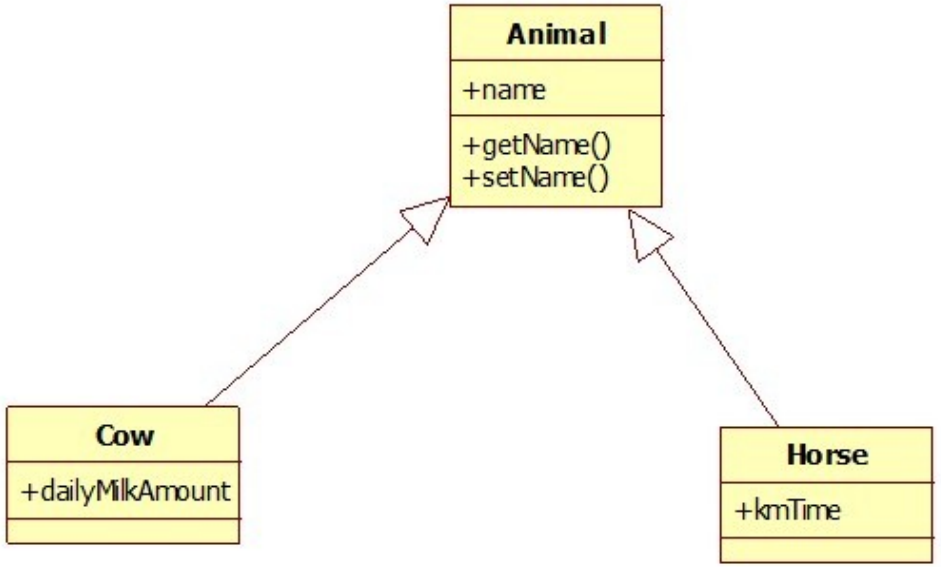
Inheritance, generalization

"Is"

base class (mother class)

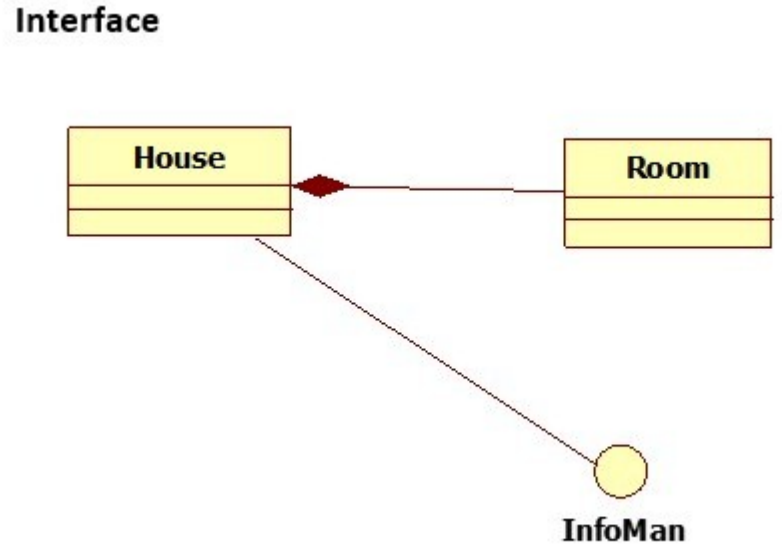
sub class (child class)

Cow and horse are animals.



Visual Studio: OOP part 6

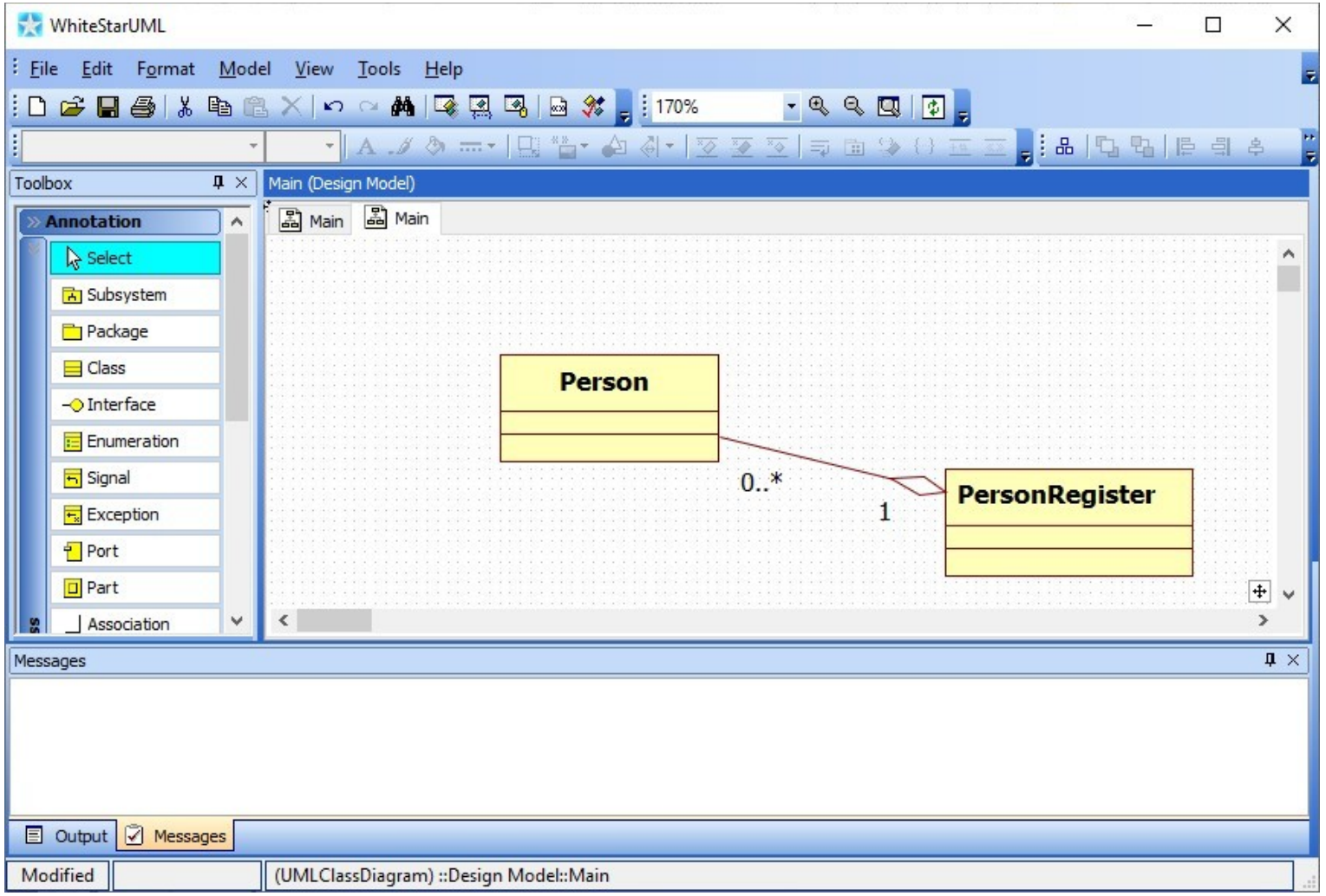
Class Diagrams: relationships between objects



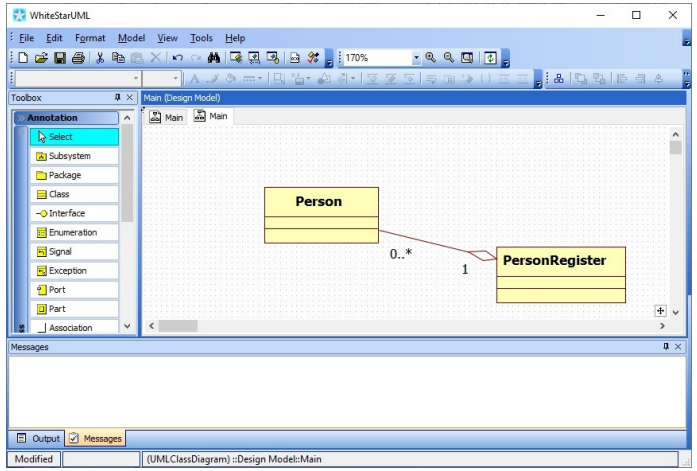
Visual Studio: OOP part 6

Class Diagrams: relationships between objects Coded example: Personregister

WhiteStarUML tool in use now



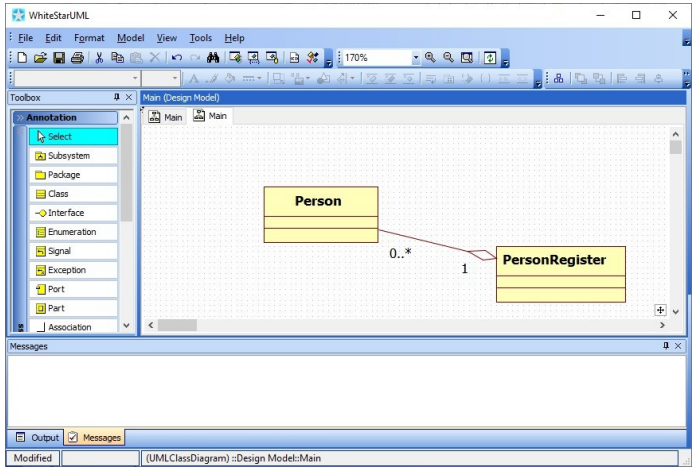
Visual Studio: OOP part 6 Codes




```
class Person
{
    public String name;
    public String hometown;
}
```

Visual Studio: OOP

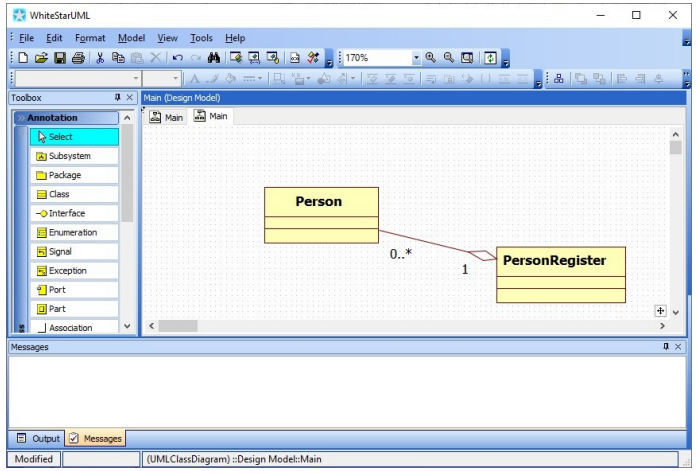
Codes




```
class PersonRegister
{
    ArrayList persons = new ArrayList();
    public void addPerson(Person p)
    {
        persons.Add(p);
    }
    public String  getPersonsList()
    {
        String all = "";
        foreach (Person pp in persons)
        {
            all += pp.name + ", " + pp.hometown + "\n";
        }
        return all;
    }
}
```

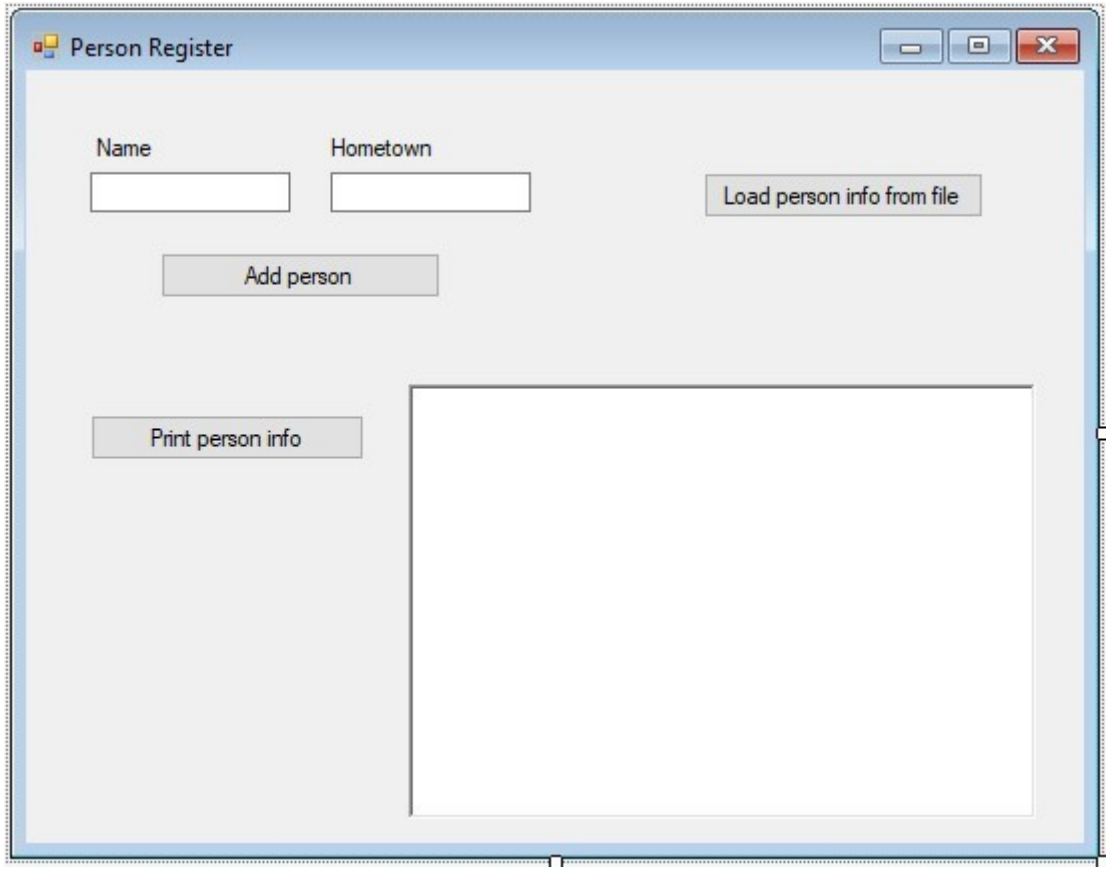
part 6

Visual Studio: OOP part 6 Codes



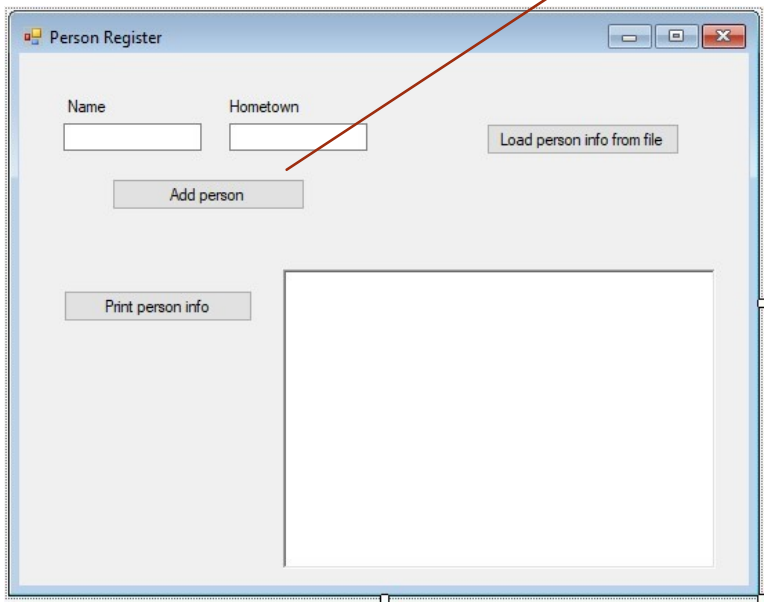
Visual Studio: OOP part 6

GUI



Visual Studio: OOP part 6

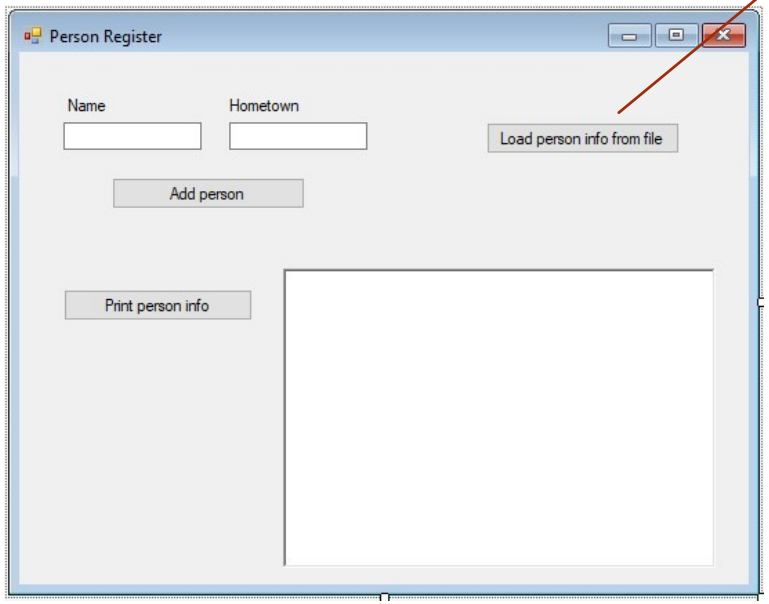
GUI




```
string path = @"info.txt";
private void button1_Click(object sender, EventArgs e)
{
    Person temp = new Person();
    temp.name = textBox1.Text;
    temp.hometown = textBox2.Text;
    String info = temp.name + "," + temp.hometown;
    using (StreamWriter sw = File.AppendText(path))
    {
        sw.WriteLine(info);
    }
    pr.addPerson(temp);
    label4.Text = "Person info added to register...";
}
```

Visual Studio: OOP part 6

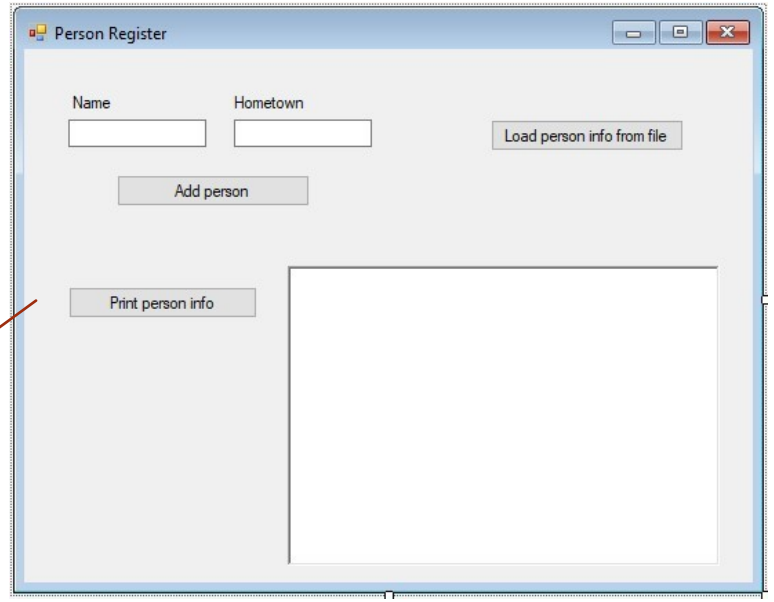
GUI




```
private void button2_Click(object sender, EventArgs e)
{
    using (StreamReader sr = File.OpenText(path))
    {
        string s = "";
        while ((s = sr.ReadLine()) != null)
        {
            String[] words = s.Split(',');
            Person temp = new Person();
            temp.name = words[0];
            temp.hometown = words[1];
            pr.addPerson(temp);
        }
    }
    label3.Text = "Person data read from file...";
}
```

Visual Studio: OOP part 6

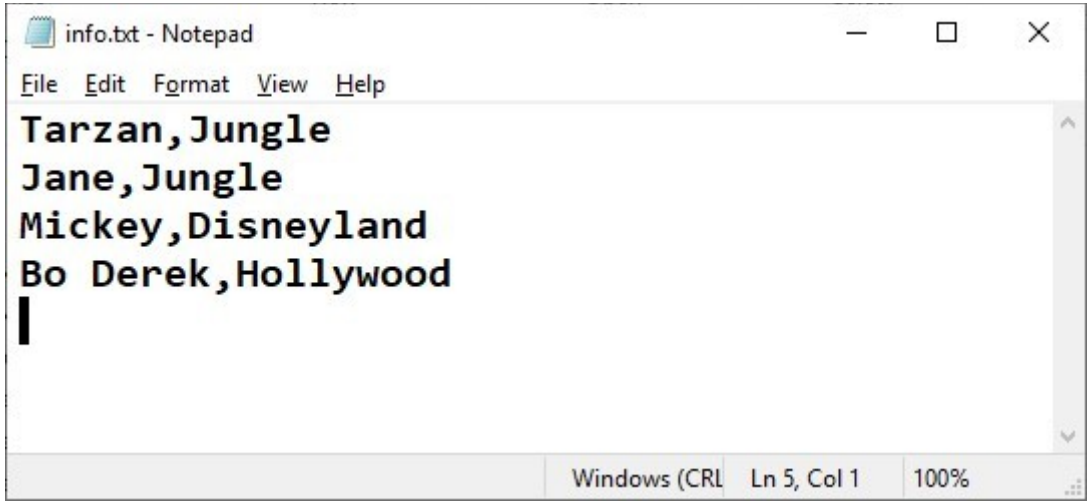
GUI




```
private void button3_Click(object sender, EventArgs e)
{
    richTextBox1.Clear();
    richTextBox1.Text = pr.getPersonsList();
}
```

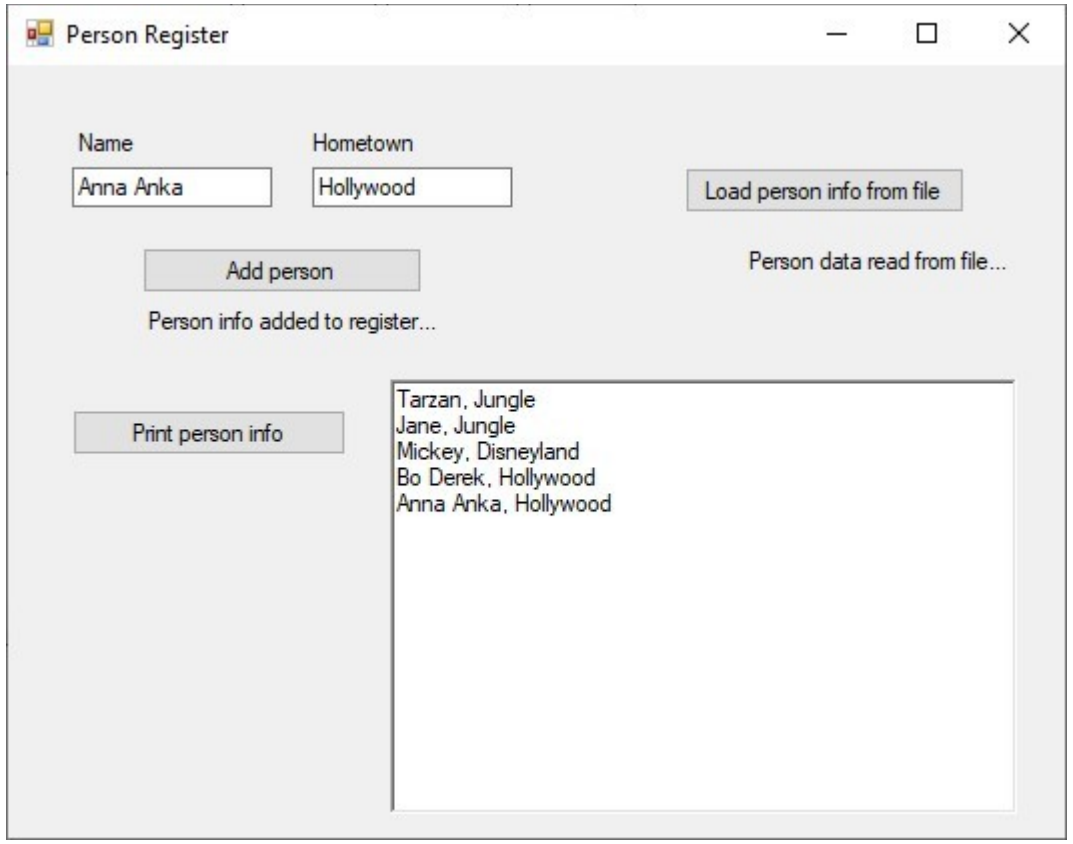
Visual Studio: OOP part 6

Textfile as a register



Visual Studio: OOP part 6

Test:


Person Register
Name
Hometown
Anna Anka
Hollywood
Load person info from file
Add person
Person data read from file...
Person info added to register...
Print person info
Tarzan, Jungle
Jane, Jungle
Mickey, Disneyland
Bo Derek, Hollywood
Anna Anka, Hollywood

UML & OOP

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:

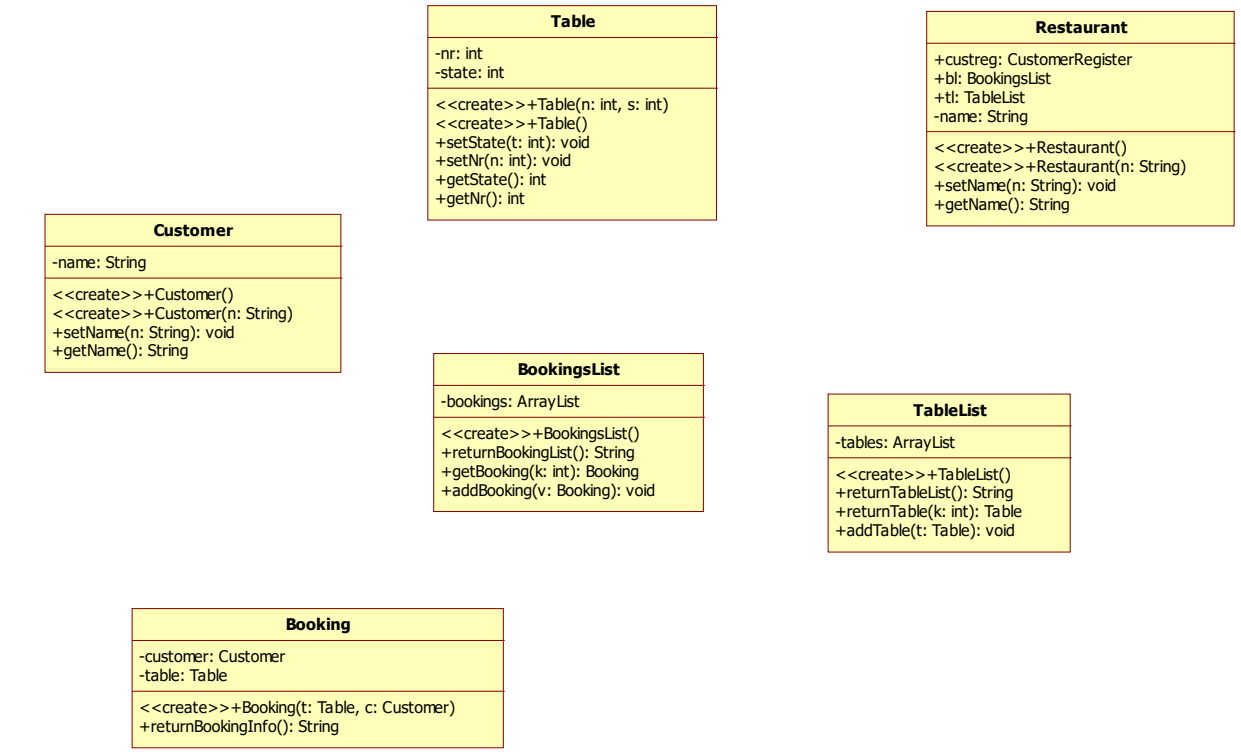
Customer

Restaurant

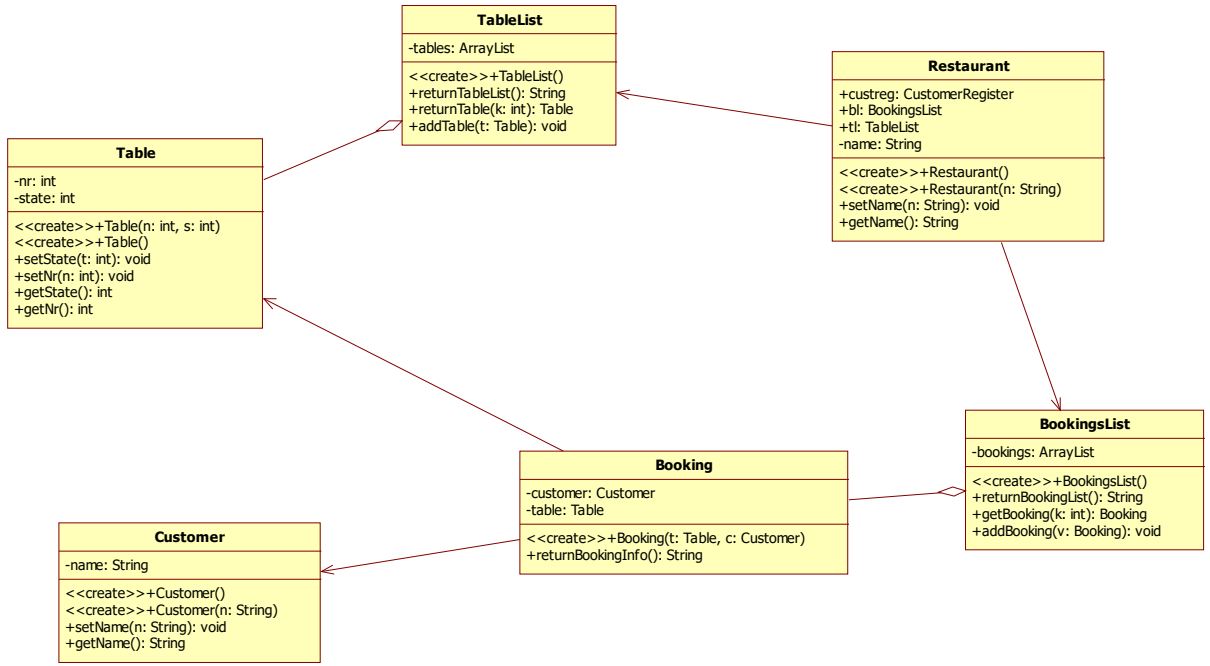
Table

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);
        }
    }
```

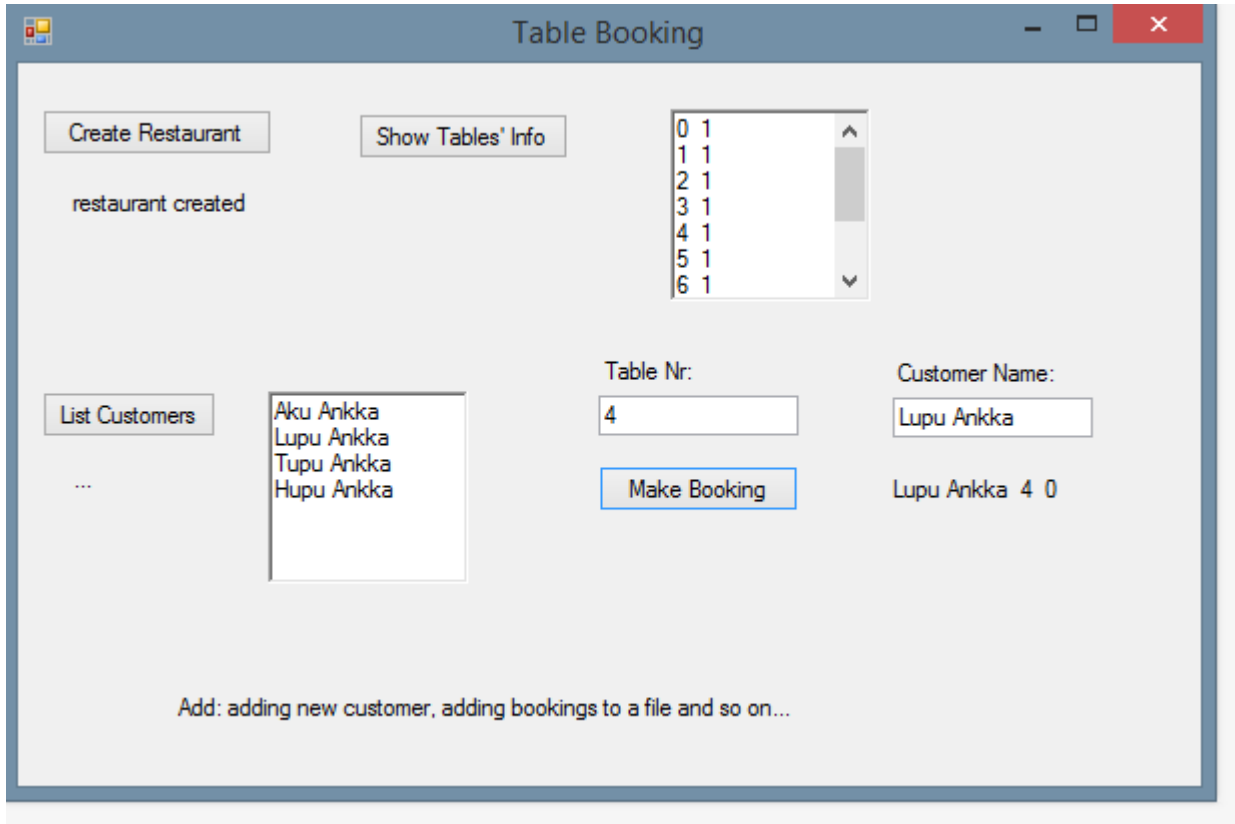
```
class TableList
    {
        private ArrayList tables;
        public  TableList()
        {
            tables = new ArrayList();
            for (int k = 0; k < 10; k++)
            {
                Table t = new Table(k, 1);
                tables.Add(t);
            }


         }



        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



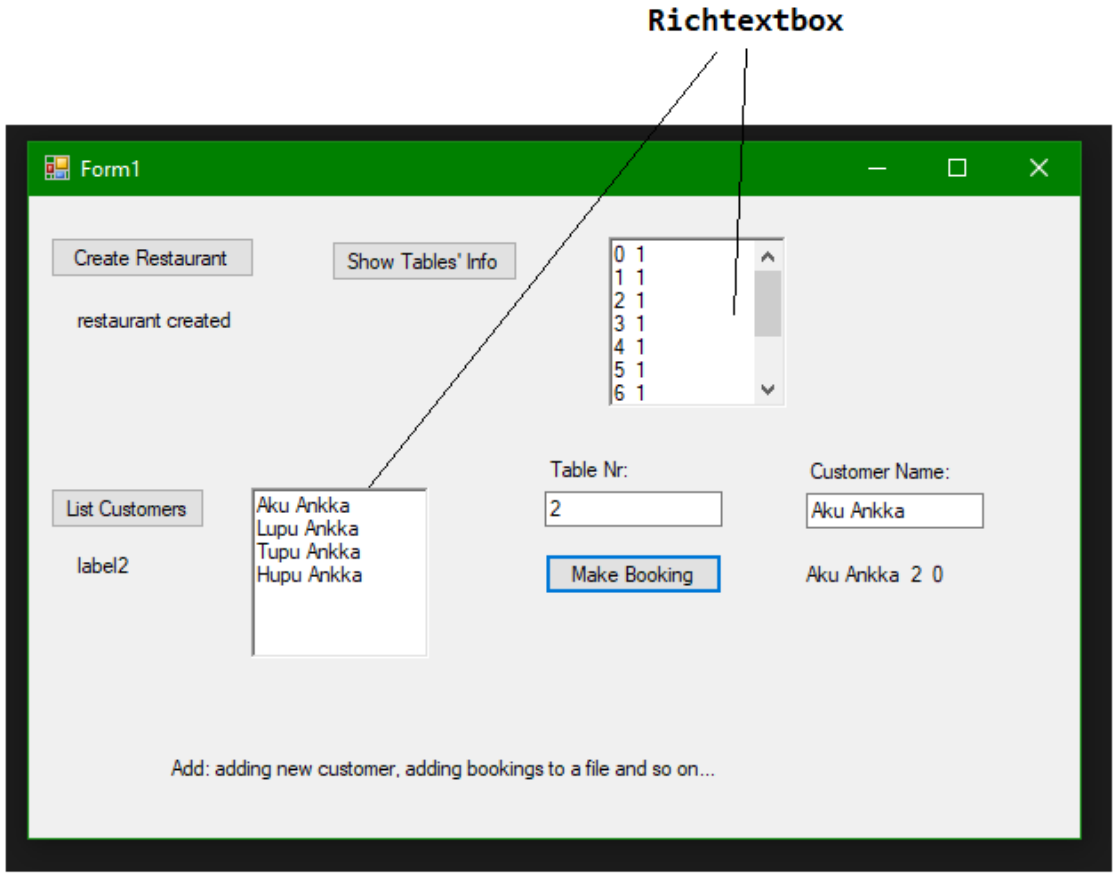
Create the app!

Make it more versatile and personal!

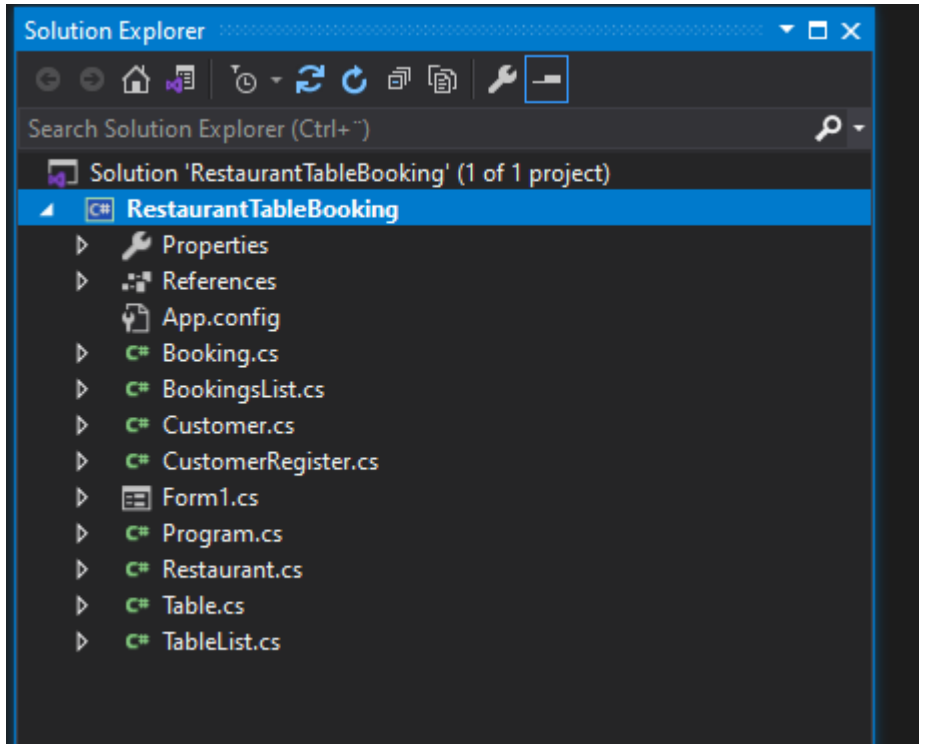
Restaurant

GUI

Create this

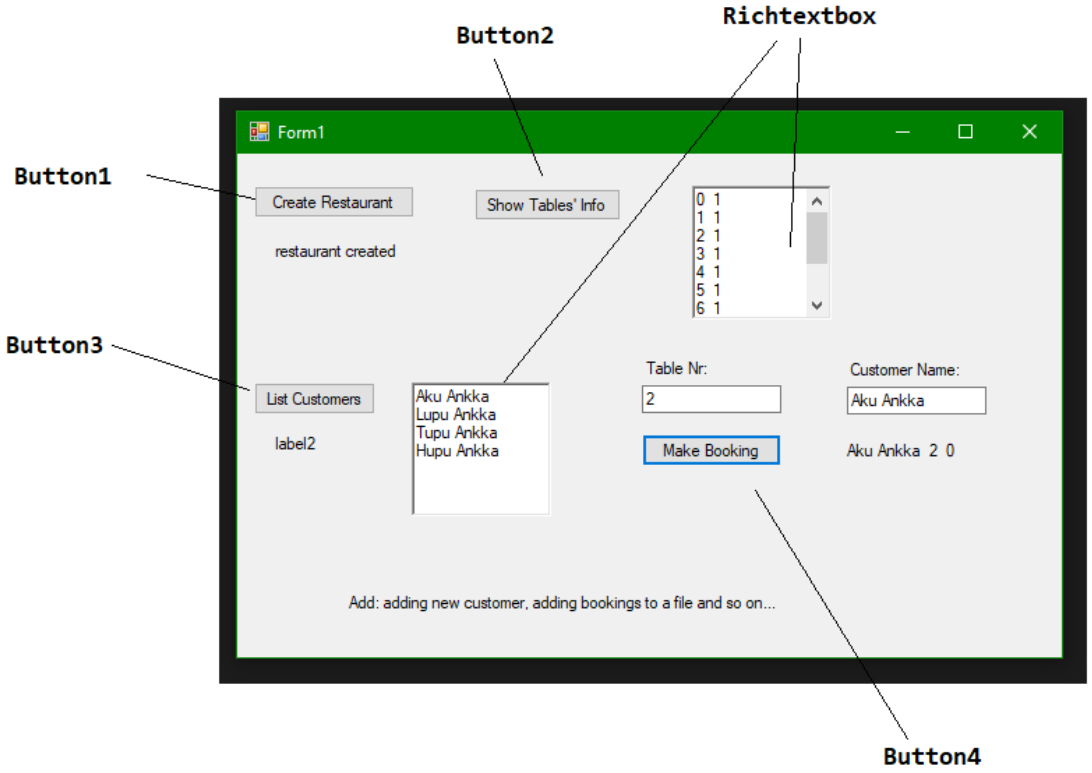


Add these classes to the project (take from console project)



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.

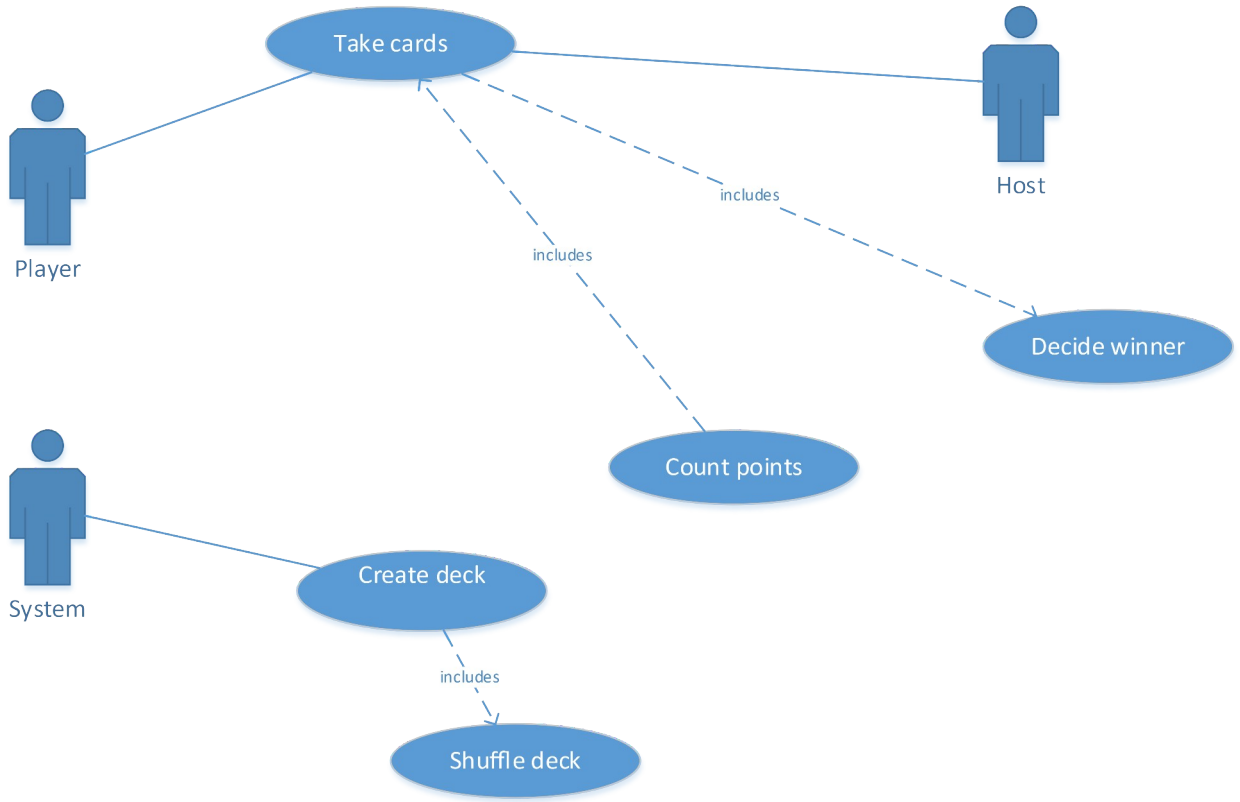
Betting is possible if wanted.

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:

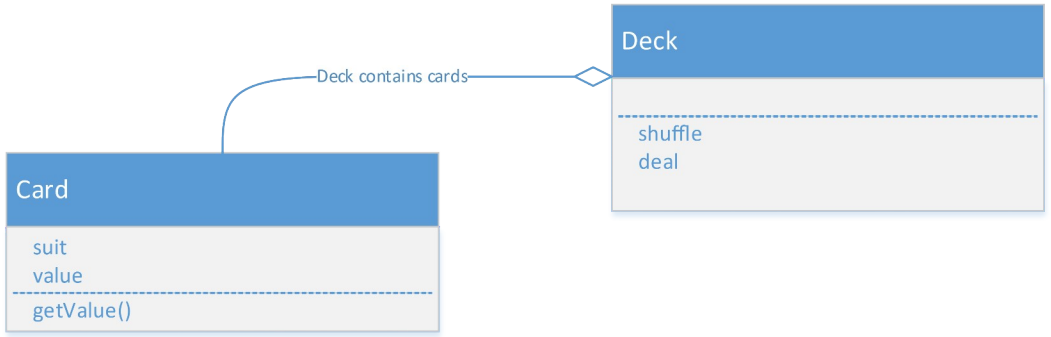
Player takes cards from the deck one by one by clicking a button. Points and card images are

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.

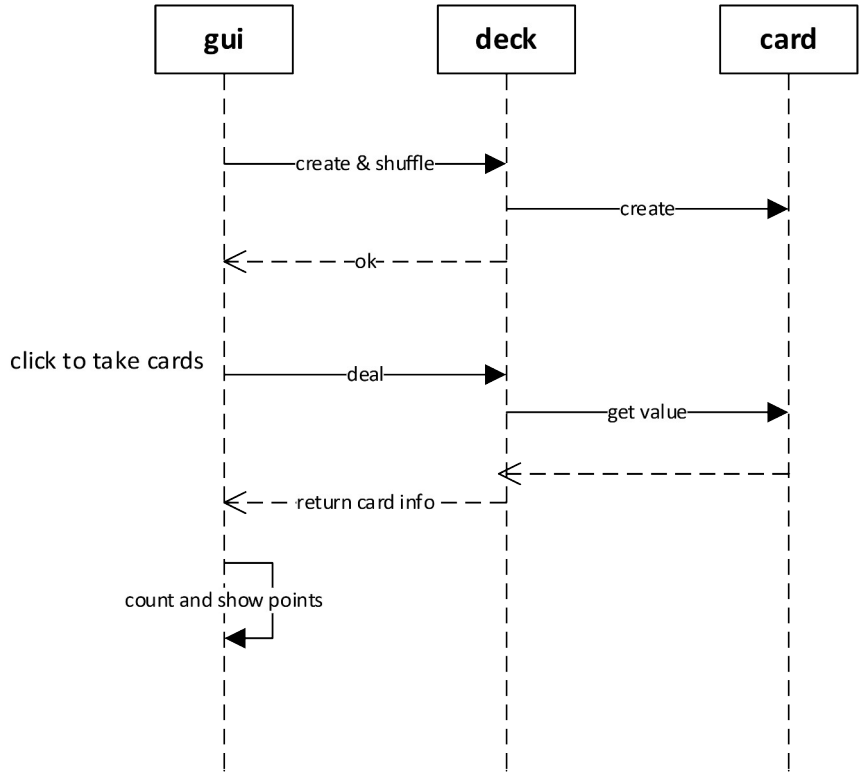
Class diagram



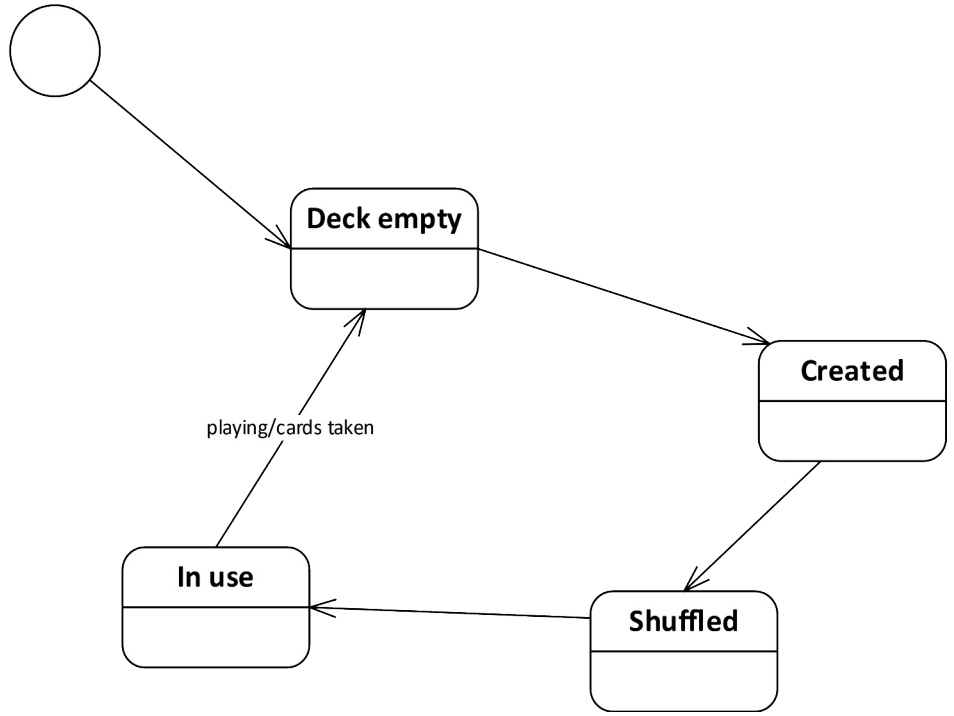
Gui prototype


Screen
Create deck
Print cards
Player: take card
Show host points
Show card image
Show card image
Show card image
Host: take cards
Show host points
Show card image
Show card image
Show card image

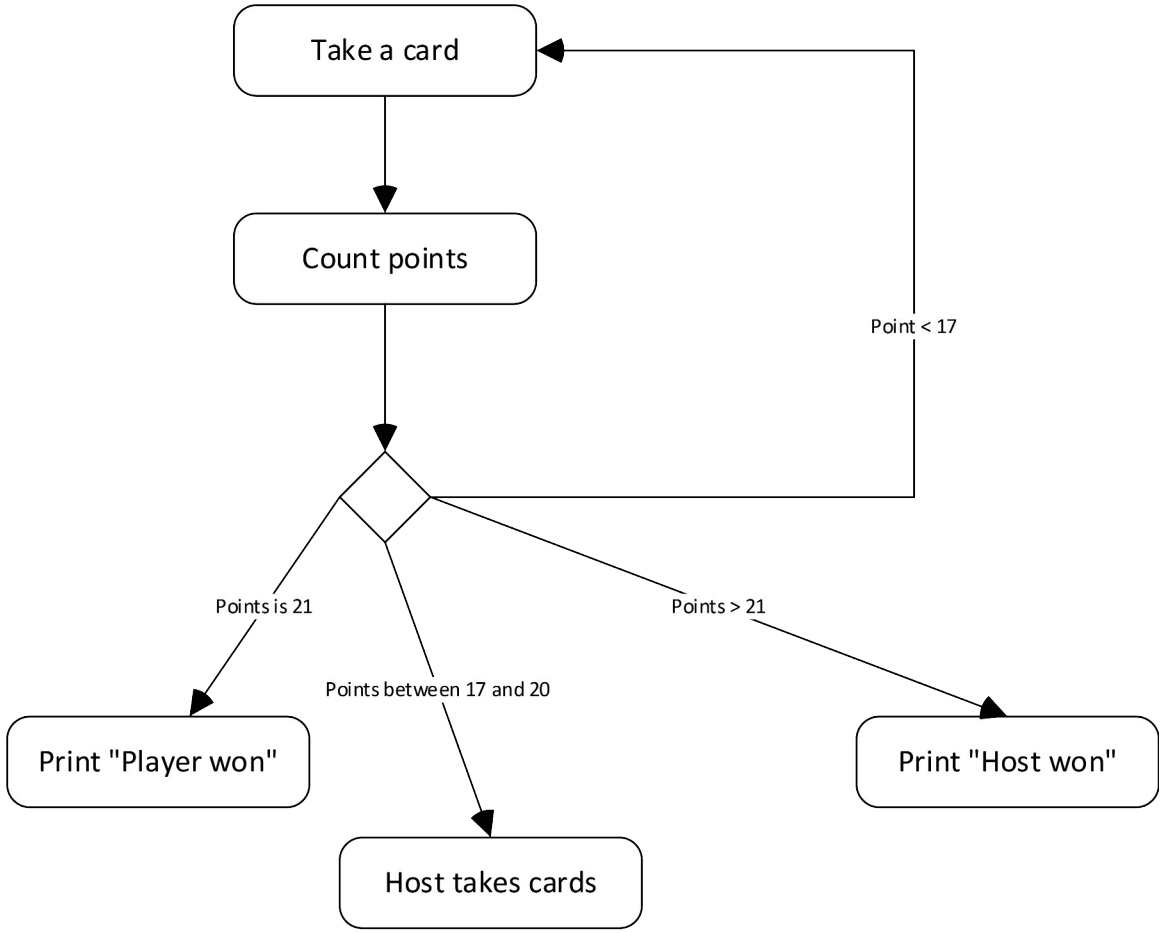
Seguence diagram



State chart



Activity diagram


Take a card
Count points
Point < 17
Points is 21
Points between 17 and 20
Points > 21
Print "Player won"
Host takes cards
Print "Host won"

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;
        }
    }
```

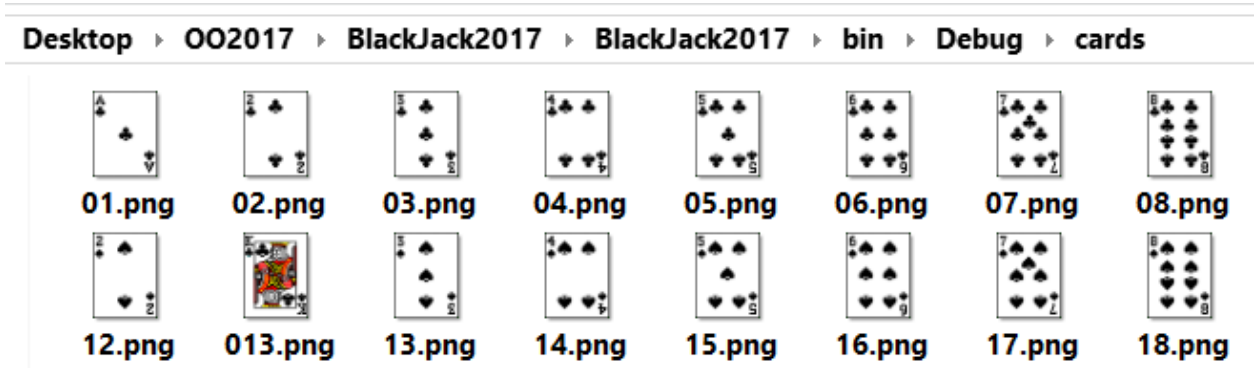
```
class Deck
  {
      ArrayList cards = new ArrayList();

      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;

            }
        }
    }
```

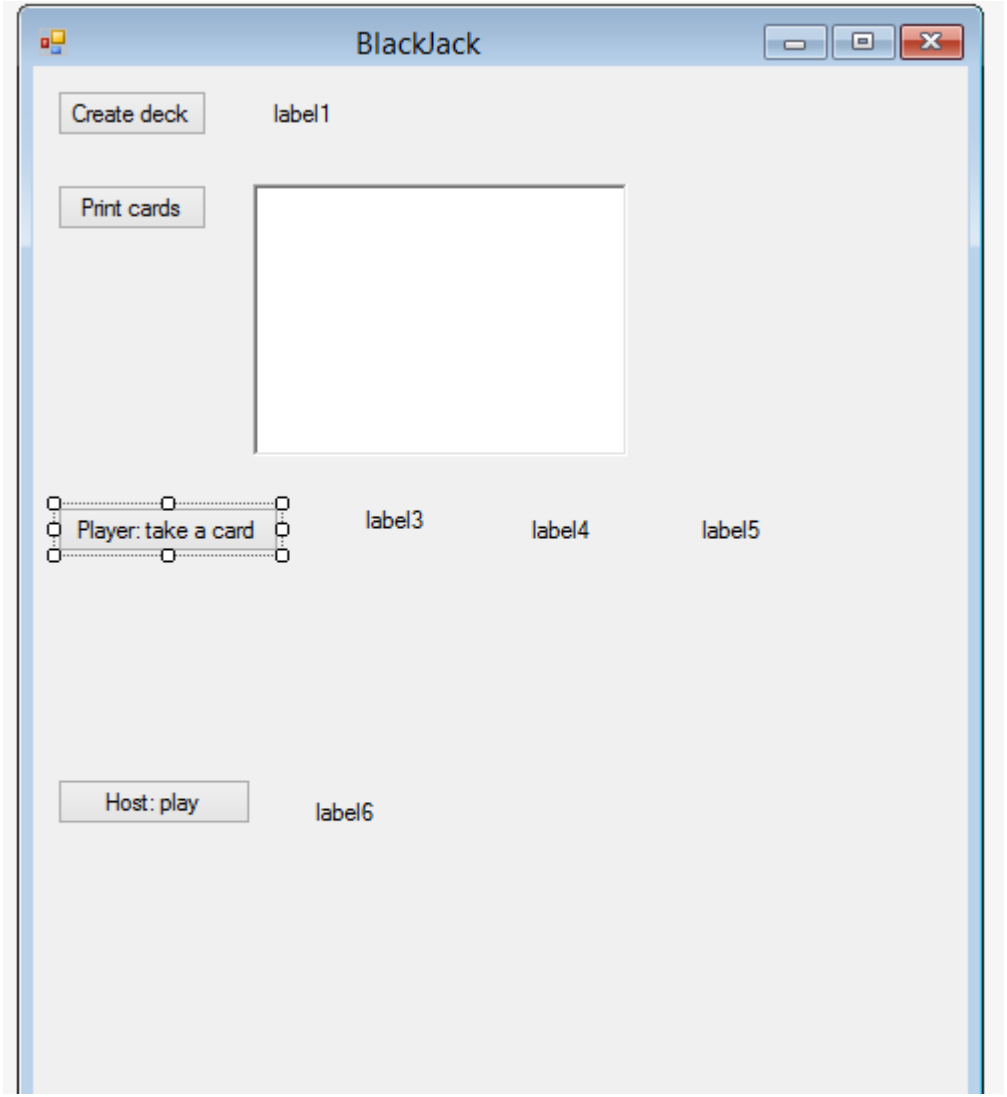
Cards

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

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



Final gui


BlackJack
Create deck
label1
Print cards
Player: take a card
label3
label4
label5
Host: play
label6

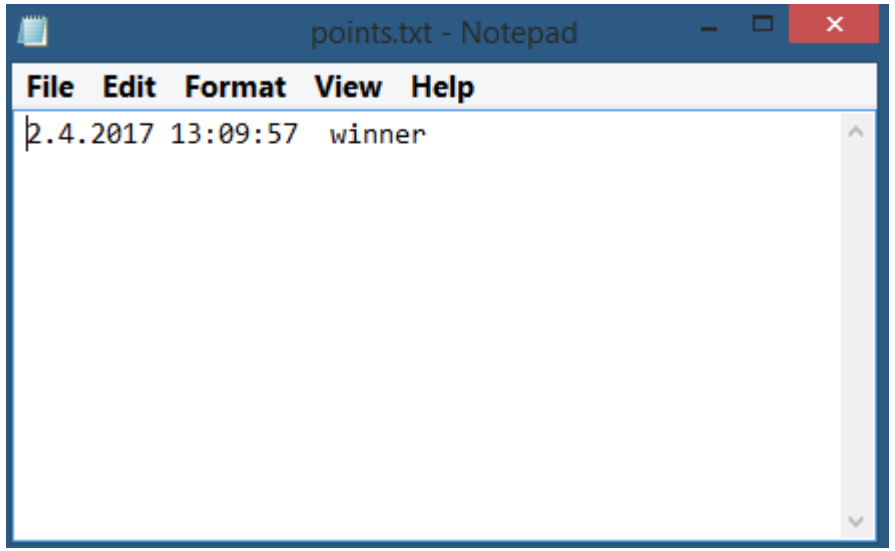
Using of exceptions

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:


BlackJack
Create deck
Deck ok
Print cards
Spade King
Club 7
Spade Jack
Diamond Jack
Spade 2
Club 10
Heart King
Diamond Queen
Club 5
Diamond 10
Player: take a card
20
cards\07.png 50 Host's turn!
Host: play
Player won!

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);

        }
    }
}
```