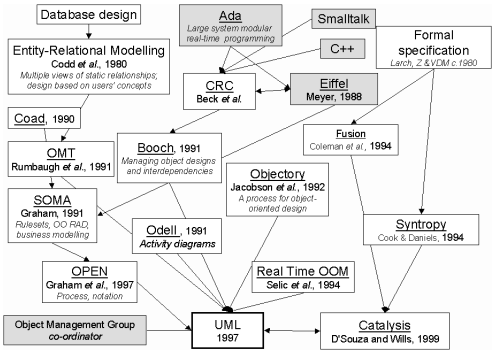
UML

Įvadas į Universalią Modeliavimo Kalbą

Ištakos

- Programų kūrimas prasideda reikalavimų analize bei projektavimu
- Įvairiems projekto aspektams išreikšti reikalinga kalba
- Šiuo metu tai **U**niversal **M**odelling **L**anguage
- Ideologai: *Grady Booch, James Rumbaugh, Ivar Jacobson.*
- Sukurta 1990-siais inicijavus **OMG.**

Ištakos: (http://uml.tutorials.trireme.com/)



UML – tai

- Grafinių diagramų "kalba", progr. sistemų specifikavimui, vizualizacijai, konstravimui, dokumentavimui.
- Sudėtinga, (ypač) turtinga žymėjimais, lanksti, plečiama, visuotinai pripažinta.
- Oficiali svetainė: www.omg.org
- Remiama firmų: *Oracle, IBM, Microsoft, Rational Software*
- Įgalina projektavimo automatizaciją

UML technologijos privalumai

- Supaprastėja komunikacija, visi kalba ta pačia kalba ⇨ iššvaistoma mažiau laiko
- Reikalavimai lengviau apibrėžiami ir dokumentuojami ⇨ mažiau "pamirštų" vietų
- Vartotojai įtraukiami į programos kūrimą nuo pat pradžių ⇨ mažiau perdarymų pabaigoje
- Priemonė išsaugoti sukauptas žinias firmoje, net jei žmonės ją palieka
- Sutaupo laiko susipažįstant su jau sukurtomis sistemomis

Šaltinis: "Baltijos programinė įranga"

Pagrindiniai diagramų tipai

Struktūrinės

- Klasių
- Objektų
- Paketų
- Komponentų
- Realizavimo (*Deployment)*

Elgsenos

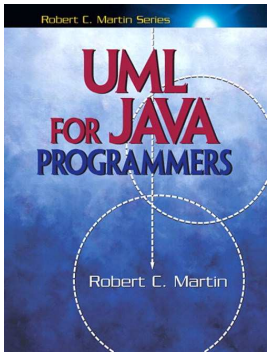
- Veiklos (*activity*)
- Būsenų *(statechart)*
- Bendradarbiavimo *(collaboration)*
- Sekų *(sequence)*
- Panaudojimo (*Use-Case*)

Klasių diagramos

- Apibrėžia statines sistemos ypatybes
- Nusako klasių sudedamasias dalis
- Atspindi klasių tarpusavio sąryšius
- Glaudžiai susijusios su objektų diagr-mis
- Gali vaizduoti klasių priskyrimą paketams

UML Class Diagrams for Java Programmers

Date: Sep 24, 2004
By Robert Martin.
Sample Chapter is provided courtesy of Prentice Hall PTR.



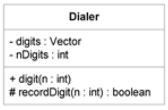
The Basics Classes

http://www.informit.com/articles/printerfriendly.asp?p=336264

Paprasčiausias pavidalas


```
public class Dialer
{
}
```

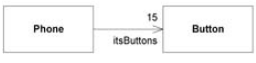
Trys poskyriai




```
public class Dialer
{
  private Vector digits;
  int nDigits;
  public void digit(int n);
  protected boolean recordDigit(int n);
}
```

Asociacija

1:15




```
public class Phone
{
  private Button itsButtons[15];
}
```

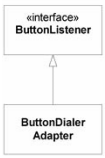
1: daug


```
public class Phonebook
{
  private Vector itsPnos;
}
```

Paveldėjimas


```
public class Employee
{
  ...
}

public class SalariedEmployee extends Employee
{
  ...
}
```

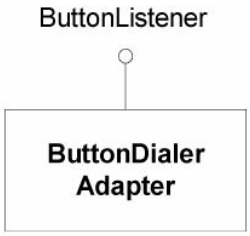
Realizacija




```
interface ButtonListener
{
  ...
}

public class ButtonDialerAdapter
  implements ButtonListener
{
  ...
}
```

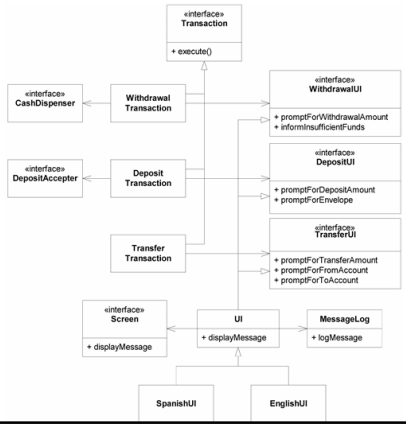
Pastaba: iš tiesų žymima **punktyrine** linija

Alternatyvi realizacijos diagrama



Pavyz-dys:

ATM

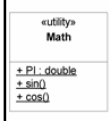


Galima `UI.java` realizacija

```
public class UI
        implements  WithdrawalUI, DepositUI, TransferUI {
        private Screen itsScreen;
        private MessageLog itsMessageLog;
        public void displayMessage(String message)  {
                itsMessageLog.logMessage(message);
                itsScreen.displayMessage(message);
        }
}
```

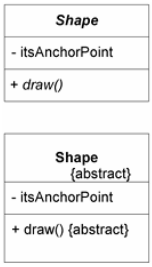
Stereotipas

- **«interface»**
- **«utility»** (klasė tik iš statinių elementų)
- Vartotojo apibrėžti: «persistent», «C-API», «struct», or «function»




```
public class Math
{
  public static final double PI =
                              3.14159265358979323;

  public static double sin(double theta){...}
  public static double cos(double theta){...}
}
```

Abstrakčiosios klasės



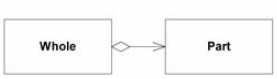

```
public abstract class Shape
{
  private Point itsAnchorPoint;
  public abstract void draw();
}
```

Savybės (*properties*)

Shape

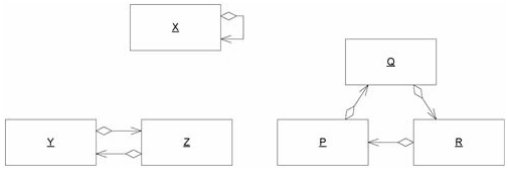
{author=Martin,
date=20020429,
file=shape.java,
private}

Agregacija (visuma-dalis)



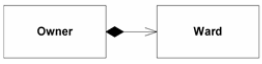

```
public class Whole
{
  private Part itsPart;
}
```

Keli objektai negali būti vienas kito dalimi



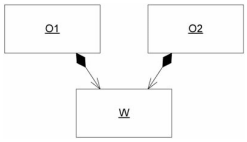
Kompozicija

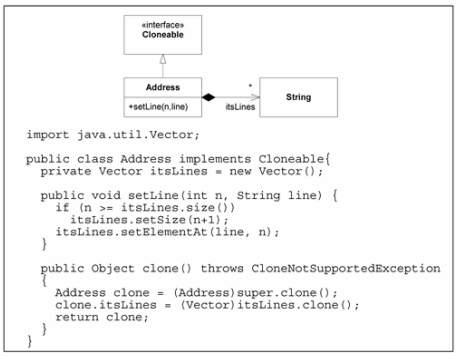
- stipresnis agregacijos atvejis: dalis negali priklausyti keliems savininkams




```
public class Owner
{
  private Ward itsWard;
}
```

Neteisinga objektų diagrama:

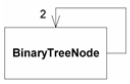





```
import java.util.Vector;

public class Address implements Cloneable{
  private Vector itsLines = new Vector();

  public void setLine(int n, String line) {
    if (n >= itsLines.size())
      itsLines.setSize(n+1);
    itsLines.setElementAt(line, n);
  }

  public Object clone() throws CloneNotSupportedException
  {
    Address clone = (Address)super.clone();
    clone.itsLines = (Vector)itsLines.clone();
    return clone;
  }
}
```

Kompozicijoje- **gili** kopija

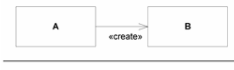
Ryšio kartotinumas (*multiplicity*)




```
public class BinaryTreeNode
{
  private BinaryTreeNode leftNode;
  private BinaryTreeNode rightNode;
}
```

• Digit.	The **exact** number of elements.
• * or 0..*	Zero to many.
• 0..1	Zero or one. In Java this is often implemented with a reference that can be null.
• 1..*	One to many.
• 3..5	Three to five.
• 0, 2..5, 9..*	Silly, but legal.

Asociacijos stereotipai




```
public class A {
  public B makeB() {
    return new B();
  }
}
```

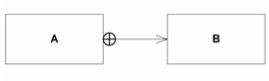

```
public class A {
  public void f() {
    B b = new B();
    // use b
  }
}
```


```
public class A {
  public void f(B b) {
    // use b;
  }
}
```


```
public class A {
  private B itsB;
  public void f() {
    itsB.f();
  }
}
```

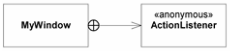
Vidinės klasės

Inner class.



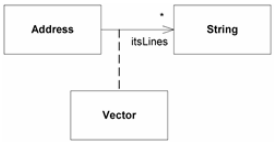

```
public class A {
  private class B {
    ...
  }
}
```

Anonymous inner class.




```
public class Window {
  public void f() {
    ActionListener l =
      new ActionListener() {
        // implementation
      };
  }
}
```

Asociacijos klasės



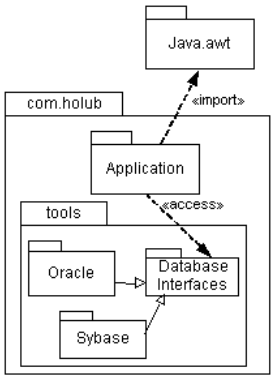

```
public class Address {
  private Vector itsLines;
};
```

Asociacijos kvalifikatorius


```
public class LoginServlet {
  private String empid;
  public String getName() {
    Employee e = DB.getEmp(empid);
    return e.getName();
  }
}
```

Pagal: http://www.informit.com/articles/printerfriendly.asp?p=336264

Paketų diagramos



Šaltinis: Allen I. Holub (http://www.holub.com).

Objektų diagramos

Object modelling symbols

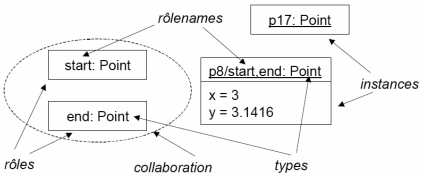


Figure 8.1 Instances, rôles and collaborations.

Šaltinis: Ian Graham or Alan Wills @ trireme.com

Pabaiga