


# Tehnici Avansate de Ingineria Programării


Introducere – 26 Mai 2014

Adrian Iftene  
adiftene@info.uaic.ro

# Cuprins

- ▶ Conținutul cursului
  - ▶ Laboratoarele
  - ▶ Proiectul
  - ▶ Examenul
  - ▶ Notarea
  - ▶ Protocolul de comunicare
  - ▶ Bibliografie
  - ▶ Conținut – pe scurt
- 


# Conținut IP (Ce ar trebui să știți...)

- ▶ Ingineria programării (Software engineering)
  - ▶ Modele de proiectare (Design models)
  - ▶ Ingineria cerințelor (Requirements identification)
  - ▶ Diagrame UML (UML diagrams)
  - ▶ Design patterns
  - ▶ Testare și debug (Testing and debugging)
  - ▶ Întreținere (Maintenance)
  - ▶ Metrice software (Software metrics)
  - ▶ Drepturi de autor (Author rights)
- 

# Conținut TAIP

- ▶ Design orientat obiect clase: *recapitulare GRASP și nivel mediu: GOF*, nivel ridicat: stiluri arhitecturale (șabloane), SOA, principii de design orientat obiect
- ▶ Dezvoltarea și mentenanța sistemelor: dezvoltare agilă condusă de model, design condus de domeniu, dezvoltare condusă de teste, refactorizare
- ▶ Modelare, modelarea afacerilor: BPMN, limbaje specifice domeniu (DSL), cadre de lucru: Eclipse Modeling Framework, Open Architecture Ware (OAW)

# Laboratoarele IP (Ce ar trebui să știți...)

- ▶ Diagrame UML
  - ▶ Design Patterns
  - ▶ Unit testing
  - ▶ Java, C++, C#, OOP (coding style)
  - ▶ Comunicare, Planificare
  - ▶ Evaluare, Buget, Negociere
- 

# Laboratoarele TAIP

- ▶ AOP, QoS, SOA, MOP
- ▶ Refactorizare: îmbunătățirea arhitecturii codului existent
- ▶ Testare automată
- ▶ Aplicarea șabloanelor de proiectare avansate
- ▶ Se negociază punctajele pe echipă, membru,...
- ▶ Nu există limită superioară pentru punctaj
- ▶ EXISTĂ limită inferioară pentru punctajele laboratoarelor și proiectului
- ▶ **Important:** Faceți legătura cu laboratoarele de Tehnologii Java!!!

# Proiectul


- ▶ Lucrul în echipă (3–4–5–6 persoane) + 1, 2 coordonatori
- ▶ Va presupune:
  - Realizarea proiectului de cercetare urmând pașii specifici din ingineria software
  - Documentare (ce au făcut alții: care sunt cele mai importante nume în domeniu, ce au făcut ei, tool-uri de referință, site-uri de referință, + / -)
  - Modelarea folosind diagrame UML
  - Implementare (modul principal, interfață, AOP)
  - Testare automată
  - Evaluare, Comparatie cu alte sisteme, statistici Documentație, Publicare articol

# Examenul

- ▶ Fără documentație 35 minute
- ▶ Întrebări grilă
- ▶ Accentul se va pune pe înțelegerea noțiunilor prezentate teoretic la curs și folosite practic la laborator

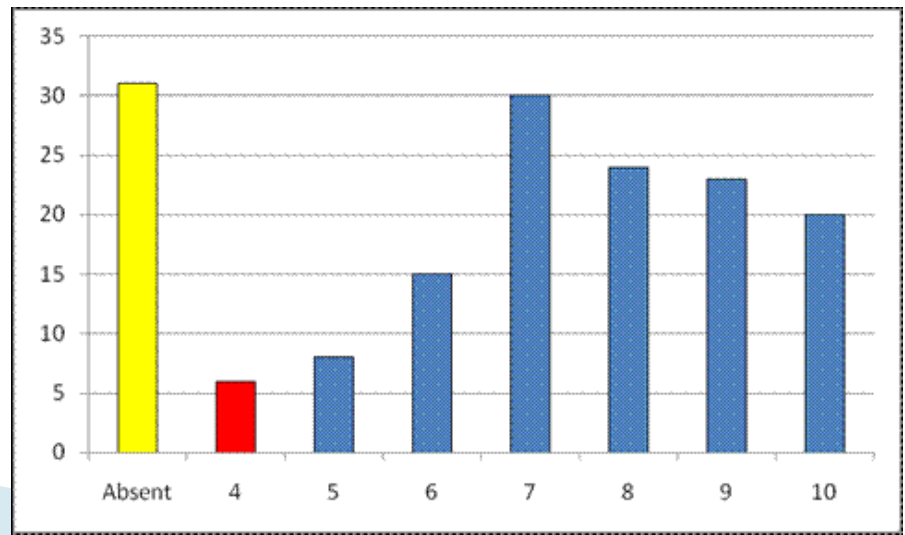


# Notarea (1)

- ▶ **Notă Laborator** – obținută în primele 7 laboratoare (teme săptămânale, lucrul în echipă) ~ *finalizare implementare*
  - ▶ **Notă Proiect** – obținută în ultimele 6 laboratoare (un proiect de echipă în care fiecare își va aduce contribuția) ~ finalizare componentă cercetare
  - ▶ **Notă Examen** – 35 minute, subiecte grilă, accentul va cădea pe înțelegerea noțiunilor parcurse
- 

# Notarea (2)

- ▶ **Nota Finală** = (Notă\_Laborator + **Notă\_Proiect** + 2 \* Notă\_Examen) / 4 / *Curba lui Gauss*
- ▶ Condiții de promovare
  - Notă\_laborator > 40 % din Notă\_Max\_Laborator
  - **Notă\_proiect > 60 % din Notă\_Max\_Proiect**
  - Notă\_examen > 40 % din Notă\_Max\_Examen
  - (Notă\_Max\_Laborator + Notă\_Max\_Proiect = Notă\_Max\_Examen)



# Conținut TAIP – pe scurt

- ▶ SWEBOK: locul și rolul ingineriei programării, arii tematice, discipline înrudite
- ▶ Dezvoltarea și mentenanța sistemelor: dezvoltare agilă condusă de model, șabloane de arhitectură a aplicațiilor de întreprindere, dezvoltare condusă de teste, refactorizare: cod, arhitectură
- ▶ Design orientat obiect clase: SOA, principii de design orientat obiect
- ▶ Modelare, modelarea afacerilor: BPMN, limbaje specifice domeniu (DSL), cadre de lucru: Eclipse Modeling Framework, Open Architecture Ware (OAW)

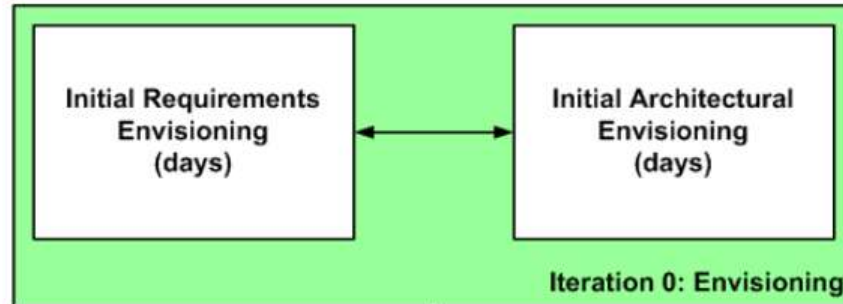
# Model driven development

- ▶ Model-driven development (MDD) – software methodology focused on creating models close to a specific field than informatics concepts
- ▶ Model-driven architecture (MDA) is the best known initiative of MDD and was launched by the group OMG (Object Management Group) in 2001

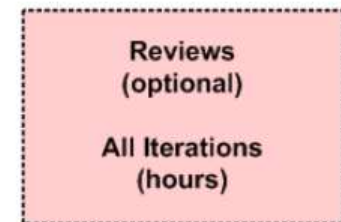
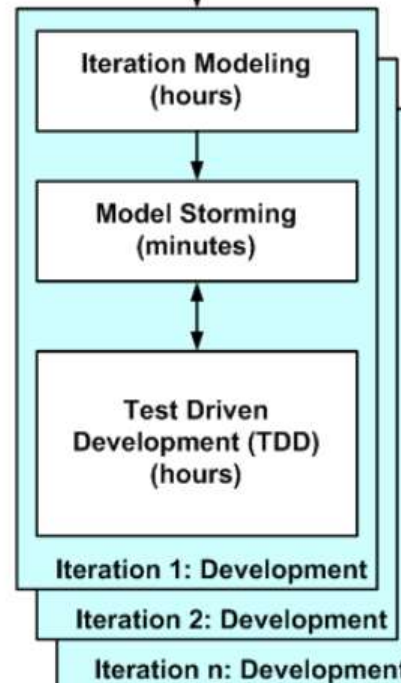
# Agile model driven development

## ► AMDD is agile version of MDD

- Identify the high-level scope
- Identify initial "requirements stack"
- Identify an architectural vision



- Modeling is part of iteration planning effort
- Need to model enough to give good estimates
- Need to plan the work for the iteration
- Work through specific issues on a JIT manner
- Stakeholders actively participate
- Requirements evolve throughout project
- Model just enough for now, you can always come back later
- Develop working software via a test-first approach
- Details captured in the form of executable specifications



# Test driven development

## ▶ Test–Driven Development – TDD

### TDD steps:

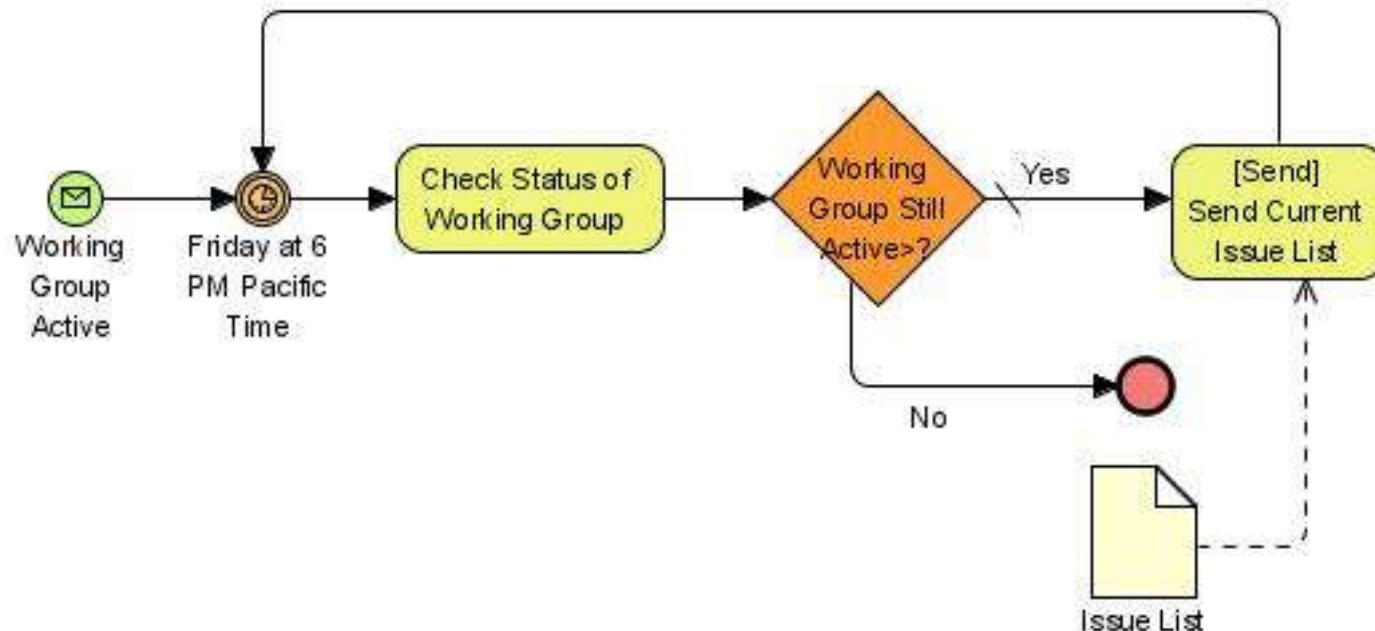
1. Add a test.
2. Execut tests; the new test will fail.
3. Add functional code such that pass all test.
4. Run tests again.
  - If the test fails, go to 3.
  - If the tests pass successfully, we can continue with other functionality
5. Refactoring code (functional and testing)

# Modeling

- ▶ IBM Rational Rose Modeler
- ▶ BPMN
- ▶ Domain specific languages (DSL)
- ▶ Working frameworks:
  - Eclipse Modeling Framework
  - Open Architecture Ware (OAW)

# BPMN

- ▶ Business Process Modelling Notation (BPMN) is a graphical representation for specifying business processes in a workflow





# SOA

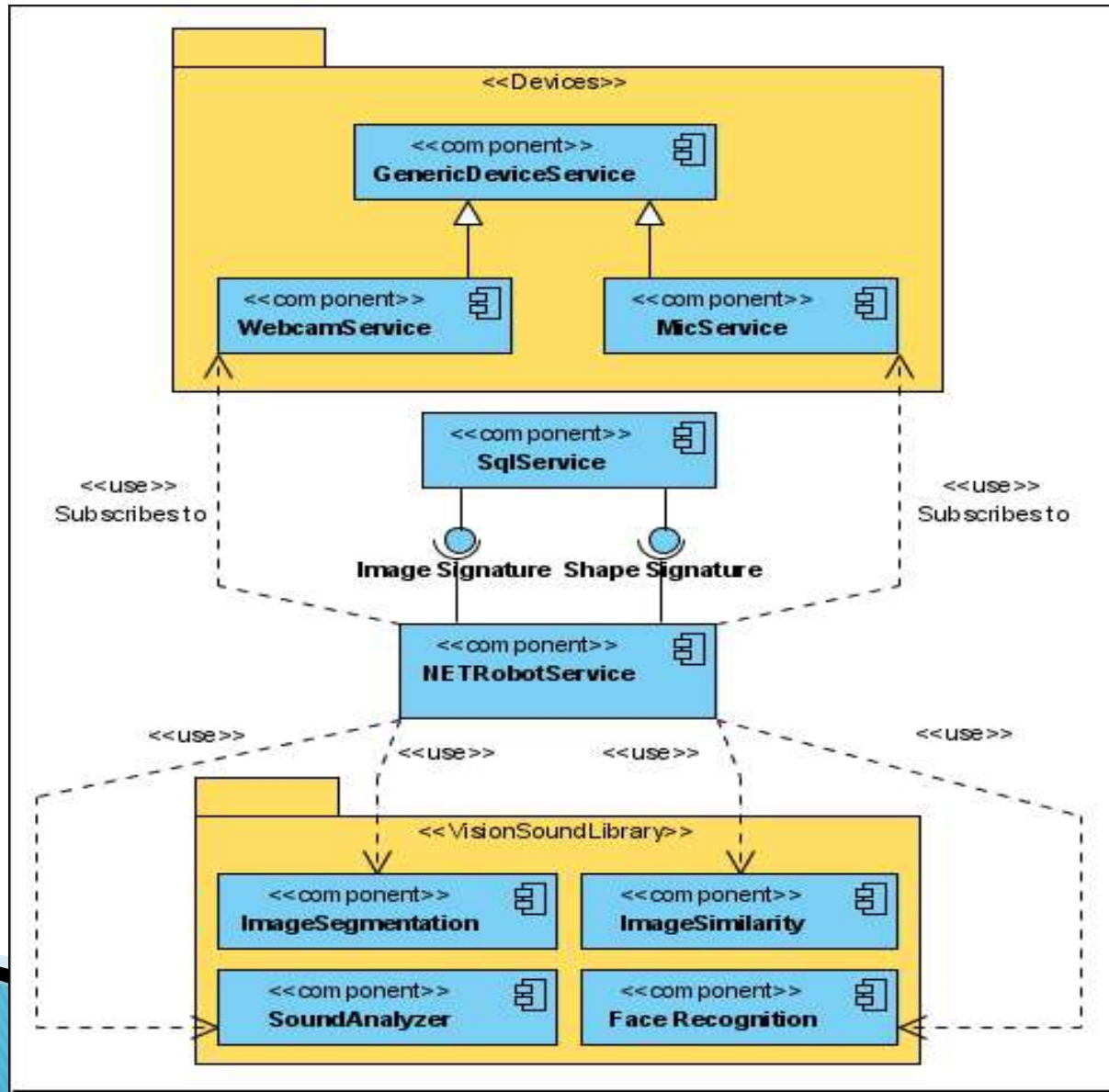
- ▶ SOA (Service Oriented Architecture) presupune distribuirea funcționalității aplicației în **unități mai mici**, distincte – numite **servicii** – care pot fi distribuite într-o rețea și pot fi **utilizate împreună** pentru a crea aplicații complexe
- ▶ **Serviciile** sunt unități funcționale independente, ce **rezolvă probleme punctuale** și pot fi **combinate** pentru a rezolva probleme complexe.
- ▶ De asemenea pot fi **reutilizate** în aplicații diferite

# SOA – Exemple

## ▶ *Exemple de servicii:*

- completarea unei cereri online pentru crearea unui cont
- vizualizarea unui extras de cont
- efectuarea unei comenzi de bilet de avion online
- Pentru un robot: servicii pentru văz, auzit, deplasat

# SOA - .NetROBOT - Tudor D.



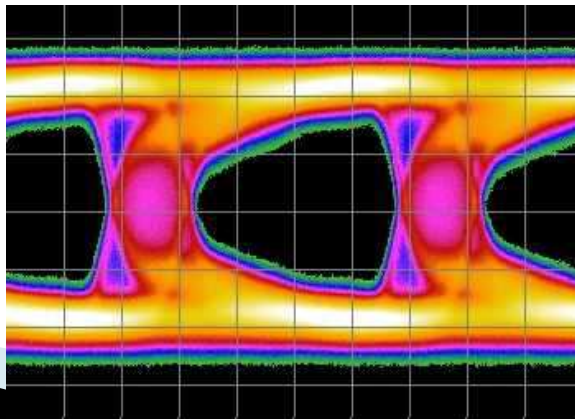
# Quality of service – Definition

- ▶ Quality of service (QoS) is the ability to provide **different priority** to different **applications, users, or data flows**, or to **guarantee a certain level of performance** to a data flow
- ▶ QoS refers to **resource reservation control mechanisms** rather than the achieved service quality
- ▶ QoS enables you to provide better service to certain flows



# QoS – Examples

- ▶ Real-time streaming multimedia applications:
  - voice over IP, online games, network support systems
  - IP-TV, cellular data communication
  - Videoconferencing, circuit emulation service
  - Industrial control systems (used for RT control of machinery)
- ▶ In these cases a required **bit rate**, **delay**, **jitter** (the deviation in or displacement of some aspect of the pulses in a high-frequency digital signal), **packet dropping probability** and/or **bit error rate** may be guaranteed

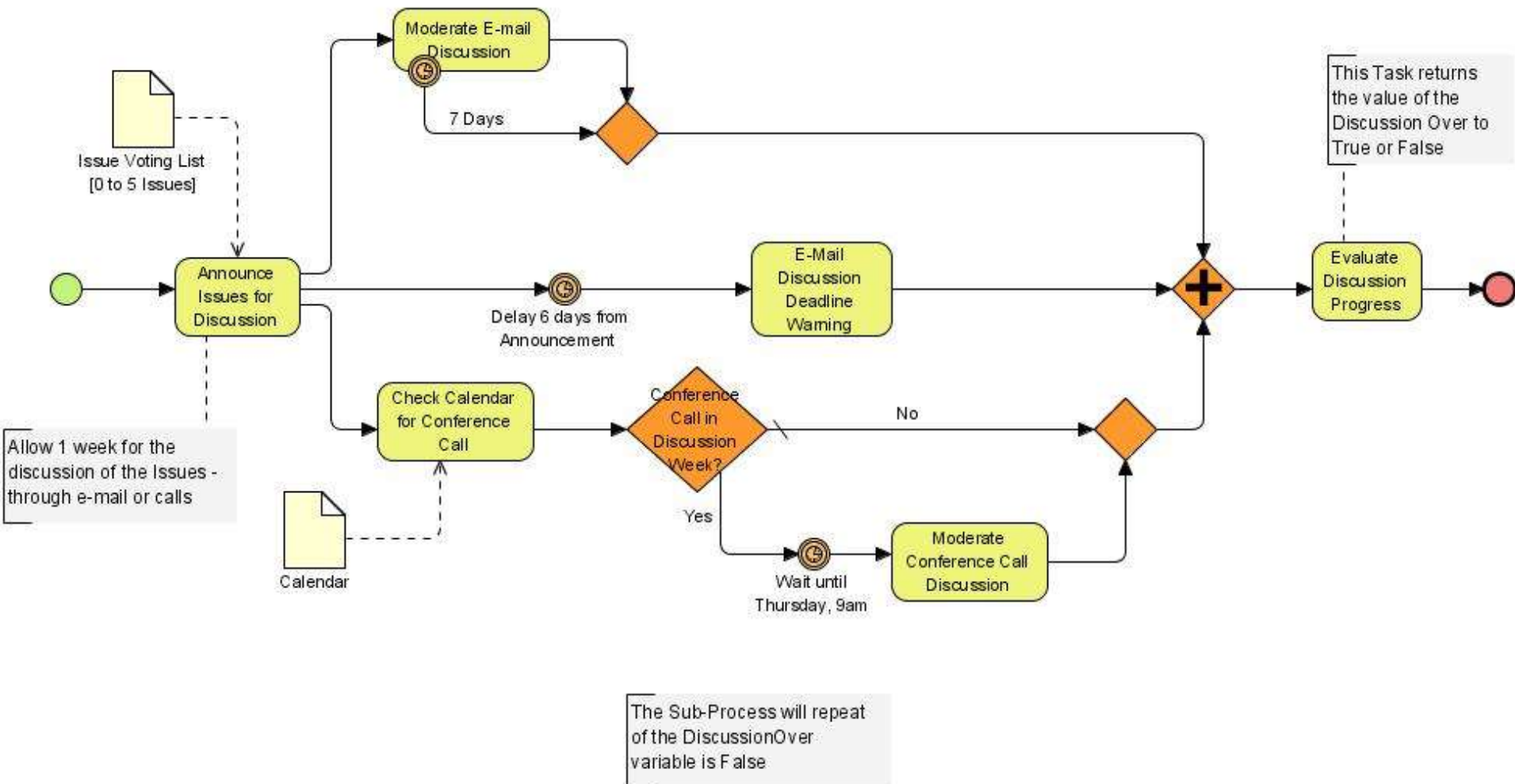


# QoS – Context

- ▶ Quality of service guarantees are important if the **network capacity is insufficient** or if we require a fixed bit rate and are delay sensitive
- ▶ **Where?** Computer networking, telecommunication networks
- ▶ **How?** A network or protocol that supports QoS may agree on a **traffic contract** with the application software and reserve capacity in the network nodes
- ▶ **Example:** it can monitor the data rate and delay, and dynamically control **scheduling priorities** in the network nodes => the most important data gets through the network as quickly as possible

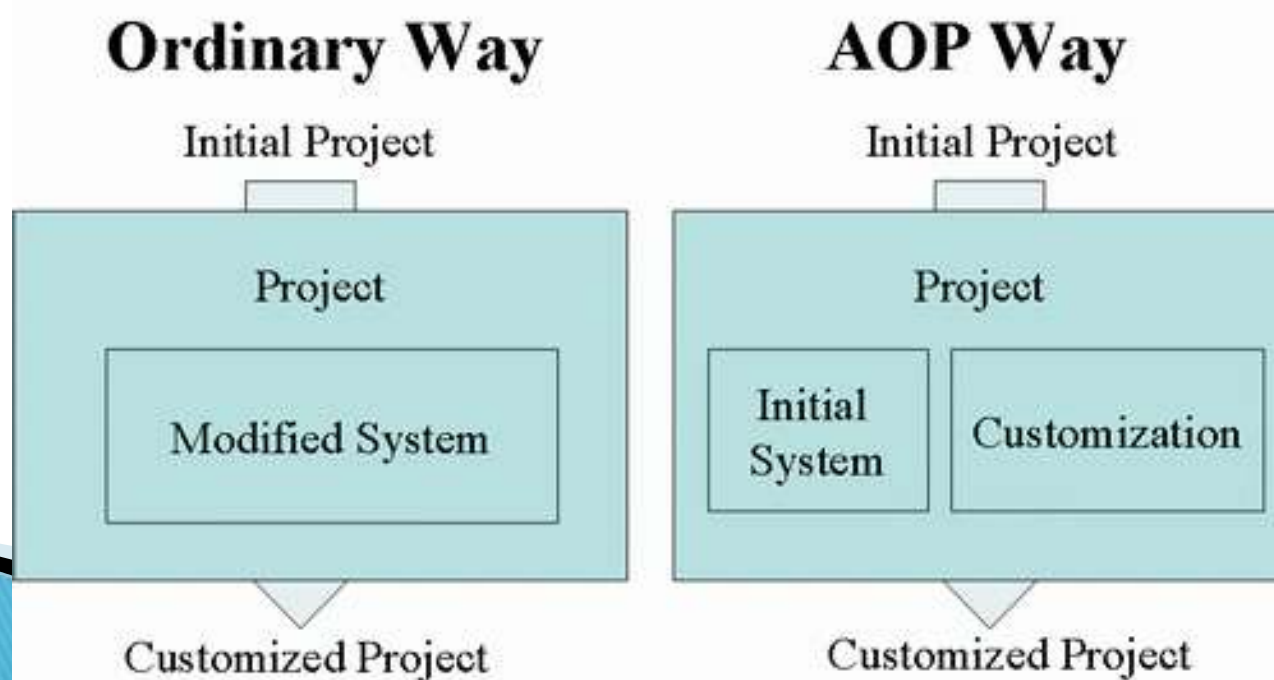


# BPMN – Example



# Aspect-oriented programming

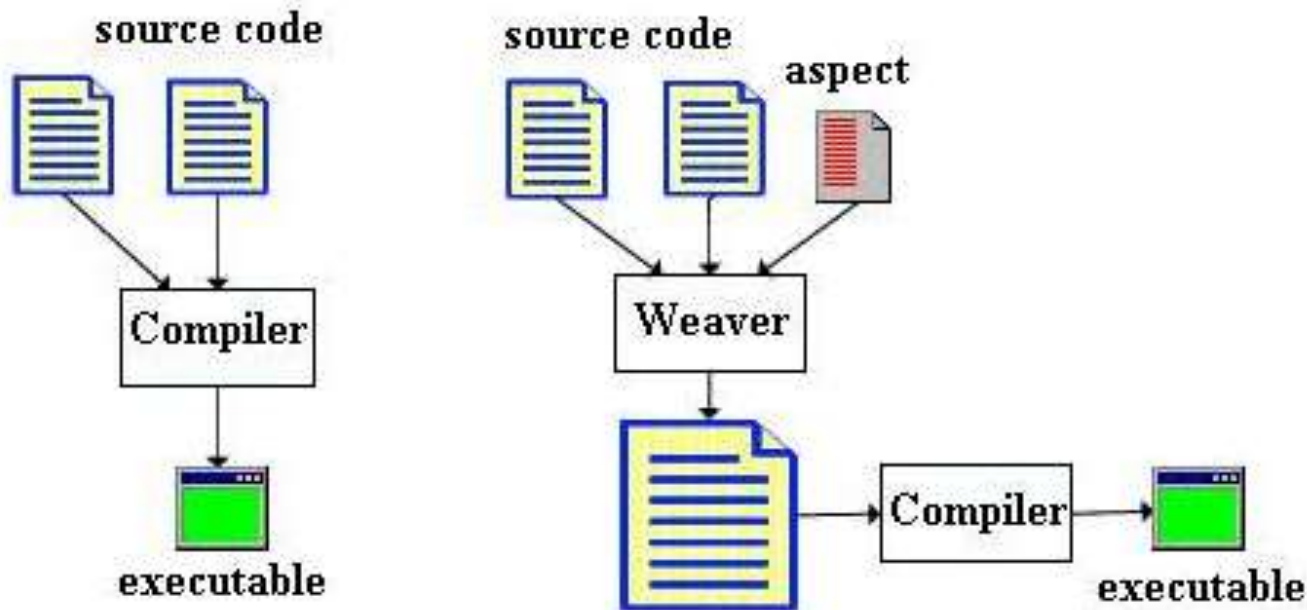
- ▶ AOP is a programming paradigm which **isolates secondary or supporting functions** from the main program's business logic
- ▶ AOP **increases modularity** by allowing the separation of **cross-cutting concerns**
- ▶ AOP includes programming techniques and tools that support the modularization of concerns at the level of the source code





# AOP – Basic Terminology

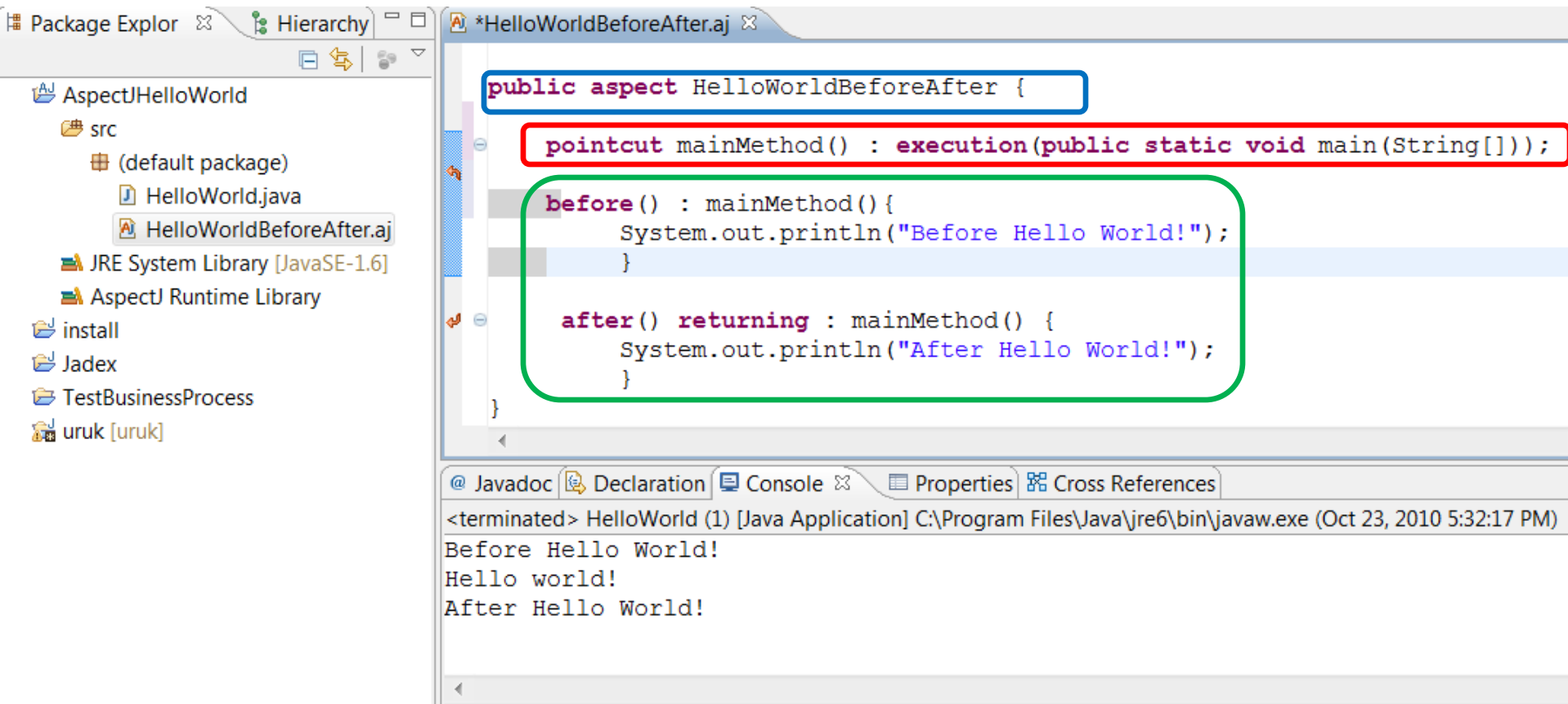
- ▶ **Cross-cutting concerns** – aspects of a program which affect other concerns
- ▶ **Advice** – additional code
- ▶ **Pointcut** – point where additional code is executed
- ▶ **Aspect** – the combination of the **pointcut** and the **advice**



# Limbaje orientate pe aspect

- ▶ Exemple: **AspectJ**, CaesarJ, CLOS, Compose, JAsCo, ObjectTeams

# AOP – AspectJ – Hello World!



Package Explorer

- AspectJHelloWorld
  - src
    - (default package)
      - HelloWorld.java
      - HelloWorldBeforeAfter.aj
  - JRE System Library [JavaSE-1.6]
  - AspectJ Runtime Library
- install
- Jadex
- TestBusinessProcess
- uruk [uruk]

```
public aspect HelloWorldBeforeAfter {  
    pointcut mainMethod() : execution(public static void main(String[]));  
    before() : mainMethod() {  
        System.out.println("Before Hello World!");  
    }  
    after() returning : mainMethod() {  
        System.out.println("After Hello World!");  
    }  
}
```

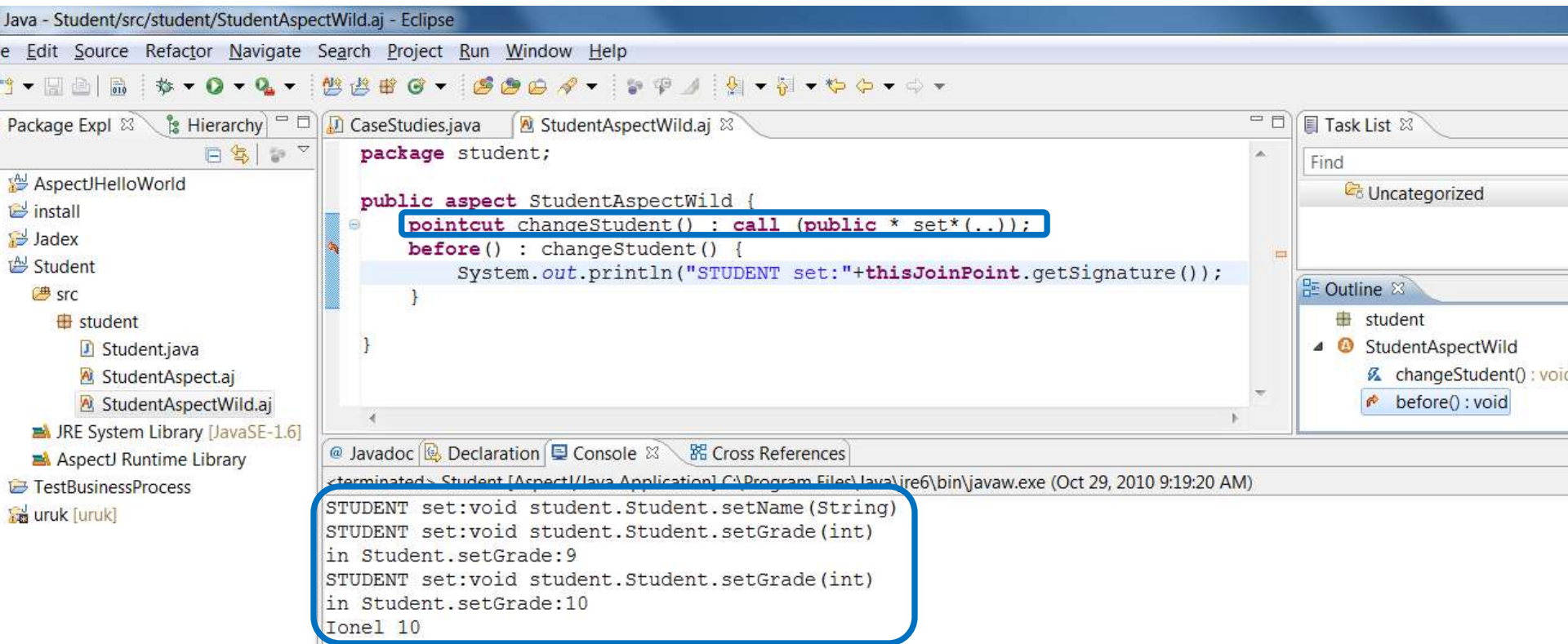
@ Javadoc Declaration Console Properties Cross References

<terminated> HelloWorld (1) [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 23, 2010 5:32:17 PM)  
Before Hello World!  
Hello world!  
After Hello World!

- ▶ aspect
- ▶ pointcut
- ▶ advice

# AOP – AspectJ – Example 2

- ▶ **Problem:** we want to know when something changes the student (*name* or *grade*)
- ▶ **Solution:** we add a pointcut for all “set” methods



# AOP – AspectJ – Example 3

- ▶ **Problem:** we want to trace our program execution
- ▶ **Solution:** we add a pointcut for all methods

The screenshot shows an IDE with three tabs: `CaseStudies.java`, `StudentAspectWild.aj`, and `TraceAspect.aj`. The `TraceAspect.aj` tab is active, showing the following code:

```
package student;

public aspect TraceAspect {

    //tracing
    pointcut trace () : call (* * (..)) && ! within(TraceAspect);
    before() : trace() {
        System.out.println("TRACE: "+thisJoinPoint.getSignature());
    }
}
```

The `pointcut trace ()` line is highlighted with a blue box. To the right, the `Task List` pane shows the `TraceAspect` with methods `trace() : void` and `before() : void`. Below the code editor, the `Console` pane shows the execution trace:

```
<terminated> Student [AspectJ/Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Oct 29, 2010 9:59:48 AM)
TRACE:void student.Student.setName(String)
TRACE:void student.Student.setGrade(int)
TRACE:StringBuilder java.lang.StringBuilder.append(int)
TRACE:String java.lang.StringBuilder.toString()
TRACE:void java.io.PrintStream.println(String)
in Student.setGrade:9
TRACE:int student.Student.getGrade()
TRACE:void student.Student.setGrade(int)
TRACE:StringBuilder java.lang.StringBuilder.append(int)
TRACE:String java.lang.StringBuilder.toString()
TRACE:void java.io.PrintStream.println(String)
in Student.setGrade:10
TRACE:void java.io.PrintStream.println(Object)
TRACE:String java.lang.String.valueOf(Object)
TRACE:StringBuilder java.lang.StringBuilder.append(String)
TRACE:StringBuilder java.lang.StringBuilder.append(int)
TRACE:StringBuilder java.lang.StringBuilder.append(String)
TRACE:String java.lang.StringBuilder.toString()
Ionel 10
```

Annotations on the console output:

- Two arrows point from the `set` and `println` labels to the first two lines of the trace: `TRACE:void student.Student.setName(String)` and `TRACE:void student.Student.setGrade(int)`.
- One arrow points from the `toString` label to the line `TRACE:String java.lang.StringBuilder.toString()`.

# Architectural degradation

- ▶ Rigid – hard to change
- ▶ Fragile – changes have undesirable effects
- ▶ Immobility – separation into components is difficult
- ▶ Viscous – things not running to properly
- ▶ Additional complexity
- ▶ Additional repetition
- ▶ Opacity – hard to understand

# Refactorizare (Refactoring)

- ▶ Schimbările succesive conduc la o structură sub-optimă a codului
  - Crește complexitatea
  - Scade claritatea
- ▶ Refactorizarea este o schimbare în structura internă a unui produs software cu scopul de a-l face mai ușor de înțeles și de modificat fără a-i schimba comportamentul observabil
- ▶ Rezultate:
  - Scăderea cuplării
  - Creșterea coeziunii

# Refactorizare – Când?

- ▶ Următoarele situații sunt semnale pentru necesitatea refactorizării:
  - Cod duplicat
  - Metode lungi
  - Clase mari
  - Liste lungi de parametri
  - Instrucțiuni *switch după tipul obiectelor* – Se recomandă polimorfismul
  - Generalitate speculativă – Ierarhie de clase în care subclasele au același comportament
  - Comunicare intensă între obiecte (cuplare puternică)
  - Înlănțuirea de mesaje



# Bibliografie

- ▶ Robert Cecil Martin: *Design Principles and Design Patterns*. [www.objectmentor.com](http://www.objectmentor.com).
- ▶ Robert Cecil Martin: *Agile Development. Principles, Patterns, and Practices*, Prentice–Hall, 2003

# Bibliografie

- ▶ Pagina cursului de IP Adrian Iftene  
<http://thor.info.uaic.ro/~adiftene/Scoala/2009/IP/>
- ▶ Pagina lui Ovidiu Gheorghies (a lucrat cu Adriana G.)  
<http://thor.info.uaic.ro/~ogh/ip/>
- ▶ Ian Sommerville: Software Engineering, Addison Wesley, 2001
- ▶ Craig Larman: Applying UML and Patterns, Addison Wesley, 2002
- ▶ Erich Gamma, Richard Helm, Ralph Johnson, John Vissides: Design Patterns, Elements of Reusable Object-Oriented Software, Addison Wesley, 1998
- ▶ Internet

# Links

- ▶ SOA: <http://www-01.ibm.com/software/solutions/soa/> ,  
<http://ro.wikipedia.org/wiki/SOA>
- ▶ SOA for the real world: <http://www.javaworld.com/javaworld/jw-11-2006/jw-1129-soa.html?page=1>
- ▶ Abstract Server  
<http://today.java.net/pub/a/today/2004/06/8/patterns.html>
- ▶ Agile Model Driven Development (AMDD)  
<http://www.agilemodeling.com/essays/amdd.htm>
- ▶ Florin Leon – IP Curs 11  
[http://eureka.cs.tuiasi.ro/~fleon/Curs\\_IP/IP11\\_Implementarea.pdf](http://eureka.cs.tuiasi.ro/~fleon/Curs_IP/IP11_Implementarea.pdf)
- ▶ OAW <http://www.openarchitectureware.org/>

Vă Mulțumesc!

Pentru prezență,  
răbdare,  
colaborare...

