Software Engineering I (02161)

Week 10

Assoc. Prof. Hubert Baumeister

DTU Compute
Technical University of Denmark

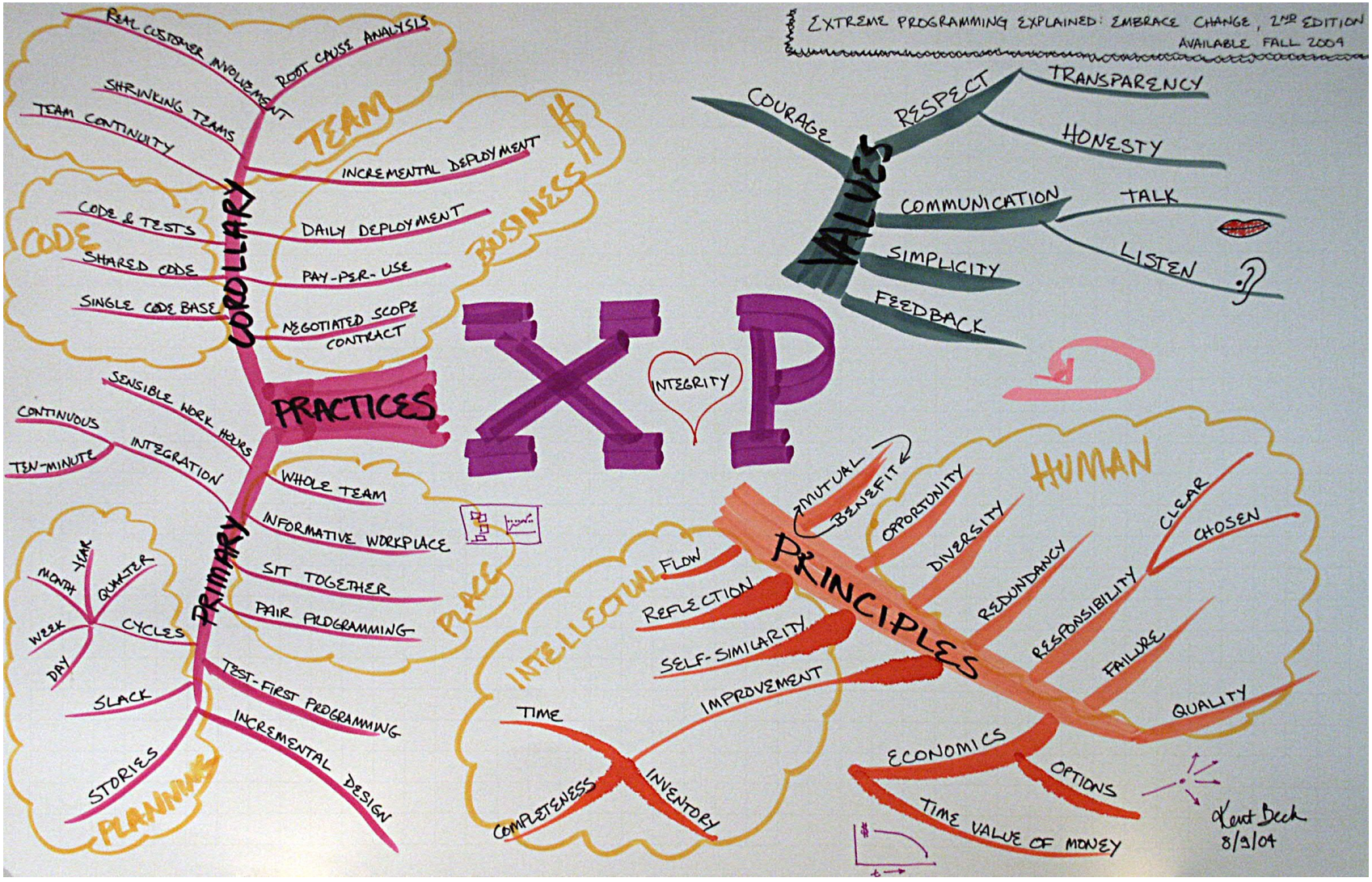
Spring 2017

Last Week

- Layered Architecture: Persistent Layer
- Software Development Processes
  - Waterfall
  - (Rational) Unified Process
  - Agile Processes: User story driven, travel light, Agile Manifesto

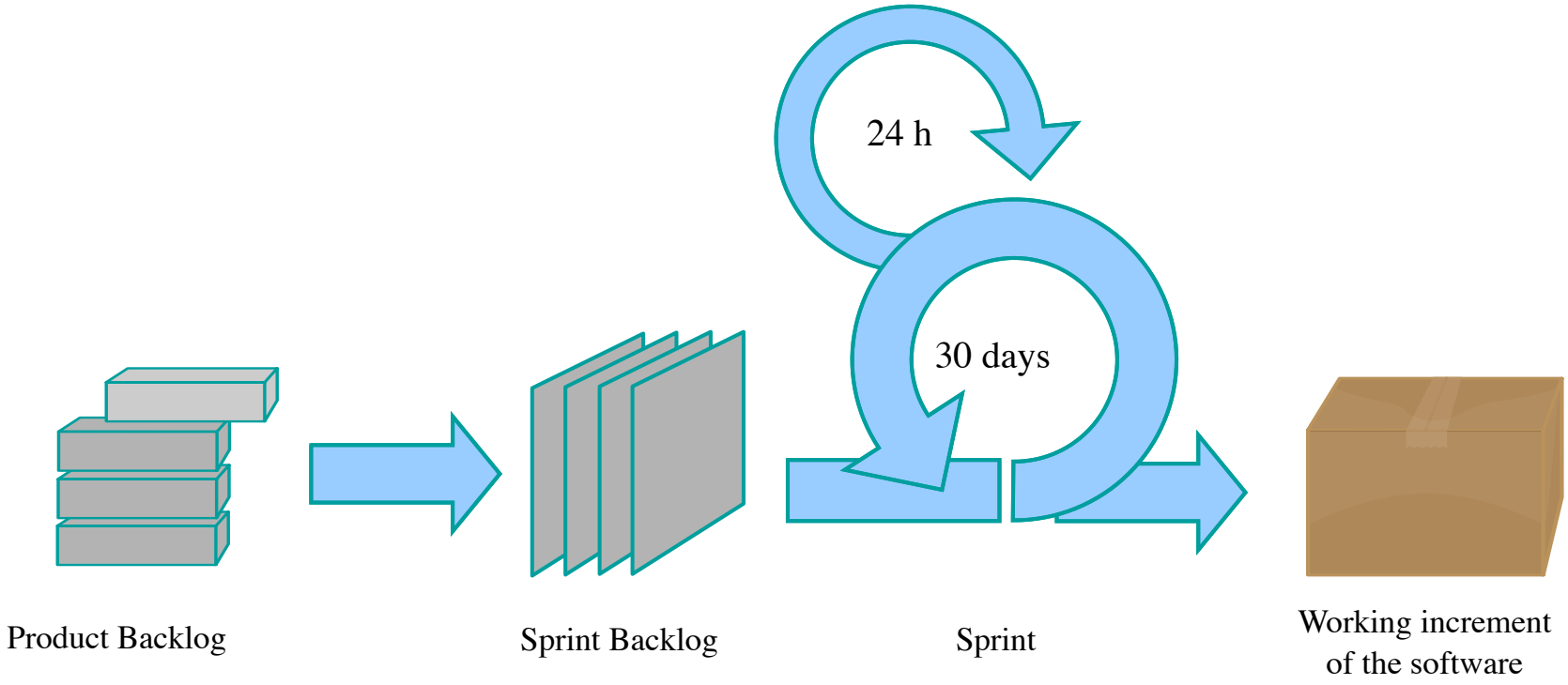
Contents

eXtreme Programming (XP)



Kent Beck, Extreme Programming 2nd ed.

Scrum



Wikipedia

- Robert Martin (Uncle Bob) about ”The Land that Scrum Forgot”
  `http://www.youtube.com/watch?v=hG4LH6P8Syk`
  - → History about agile methods, the agile manifesto, and Scrum and its relationshop to XP

Lean Software Development

- Lean Production:
  - Reduce the amount of *waste* in the production process
  - Generate *flow*
- Waste: resources used which do not produce value for the customer
  - time needed to fix bugs
  - time to change the system because it does not fit the customers requirements
  - time waiting for approval
  - ...

Cycle time

Cycle time

Time it takes to go through the process one time

$$cycle\_time = \frac{number\_of\_features}{feature\_implemantion\_rate}$$

- Example: Waterfall
  - Batch size = number_of_features in an iteration
  - Software: 250 features, feature_implementation_rate = 5 features/week
  - cycle_time = 250 f / (5 f/w) = 50 weeks
  - Overall time: 50 weeks

  → 1 cycle

Goal: Reducing the cycle time

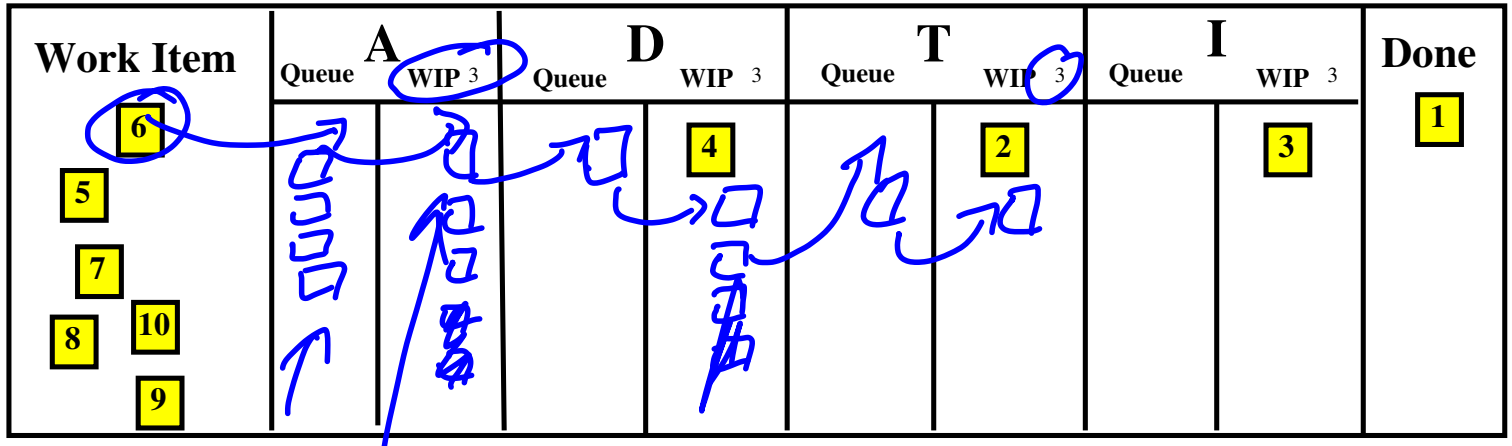
- Reduce batch size: 1 feature in an iteration
- Software: 250 features, feature_implementation_rate = 5 features/week

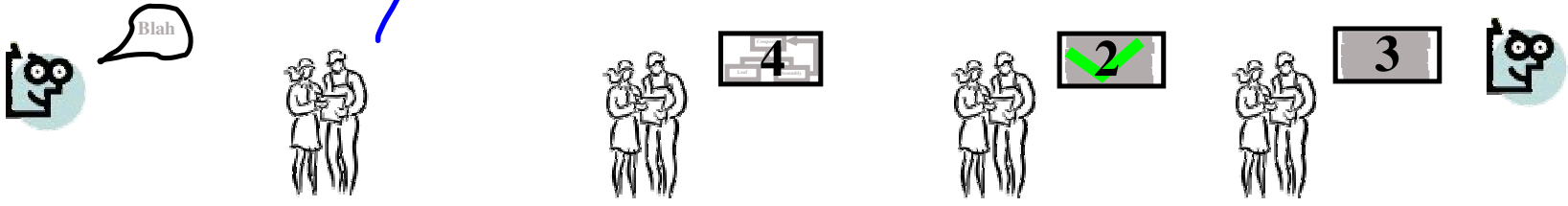
$$cycle\_time = \frac{number\_of\_features}{feature\_implemantion\_rate}$$

- Agile: cycle_time = 1 f / (5 f/w) = 1/5 week = 1 day = 8 h
- 250 cycles
- Advantages
  - Process adapts to changes in requirements
  - Process improvements and fine tuning

Generating flow using Pull and Kanban

WIP = Work in Progress Limit





Flow through Pull with Kanban

- Process controlling: local rules
- Load balancing: Kanban cards and *Work in Progress (WIP) limits*
- Integration in other processes

Figure from David Anderson `www.agilemanagement.net`

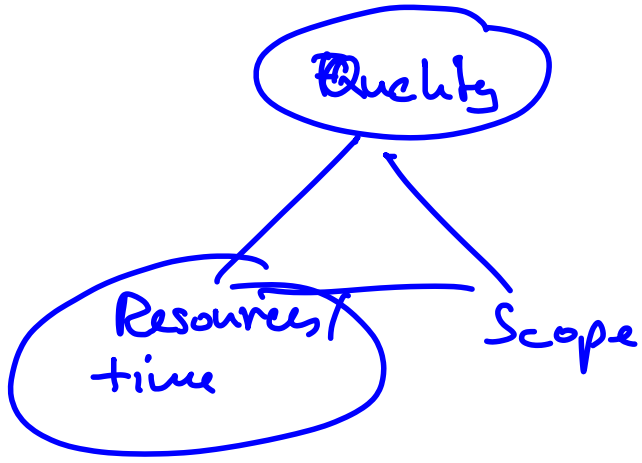
Online Kanban Tool: Trello

- `www.trello.com`: Electronic Kanban board useful for your project
- Example Kanban board `https://trello.com/b/4wddd1zf/kanban-workflow`

Contents

Project Planning

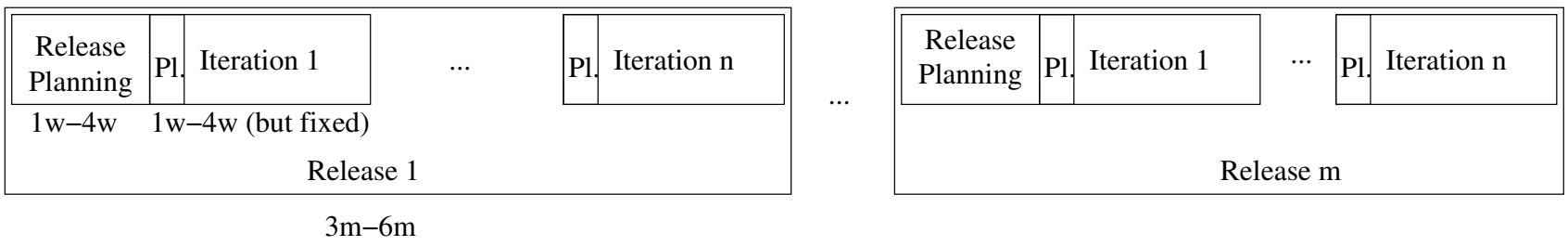
- Project plan
  - Defines:
    - How work is done
  - Estimate
    - Duration of work
    - Needed resources
    - → Price
- Project planning
  - Proposal stage
    - → Price
    - → Time to finish
  - Project start-up
    - → Staffing, . . .
  - During the project
    - Progress (tracking)
    - Adapt to changes



Planning Agile Projects

- *fixed* general structure

→ e.g. quarterly cycle / weekly cycle practices in XP / sprints in Scrum



- *time boxing*
  - fixed: release dates and iterations
  - adjustable: scope
- Planning: Which user story in which iteration / release

Planning game

- Goal of the game:
  - List of prioritized user stories
- Customer defines:
  - user stories
  - priorities
- Developer define:
  - costs, risks
  - suggest user stories
- Customer decides: is the user story worth its costs?
  - → split a user story
  - → change a user story

Scrump/XP: Project estimation and monitoring

- Estimation: two possibilities
  1) Estimate *ideal time* (e.g. person days / week) * load_factor
  2) Estimate *relative* to other user stories: *story points*
- Monitoring

ad 1) New *load factor*: total_iteration_time / user_story_time finished

ad 2) *velocity*: Number of points per iteration

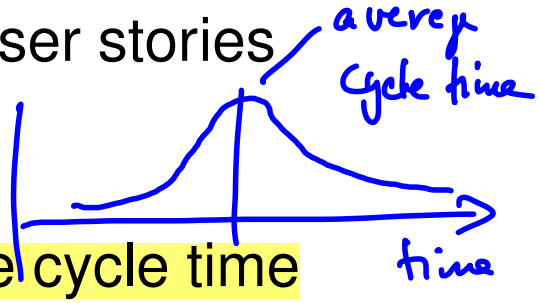
$\rightarrow$ What can be done in the next iteration

- Yesterdays weather: Calculate velocity/load_factor based on the *last* iteration only
- Important: If in trouble focus on *few* stories and finish them

Lean / Kanban: User story estimation

- No ”iterations”: user stories come in and flow through the system

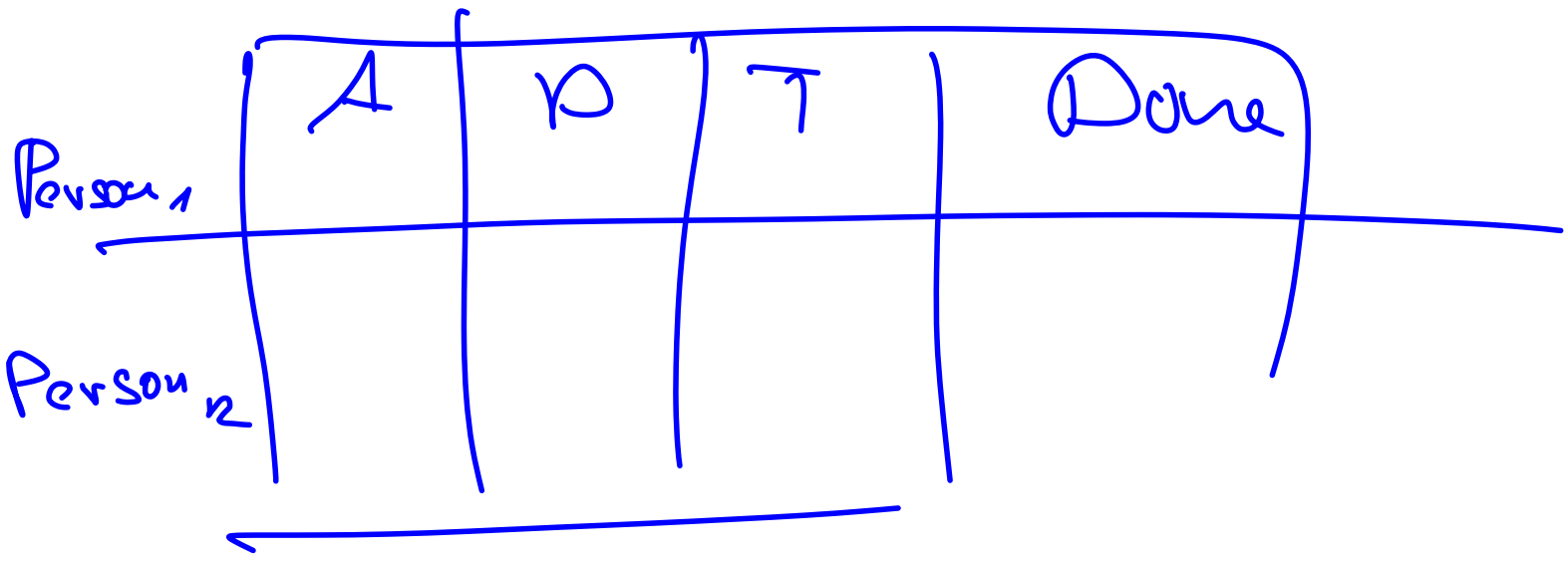
→ Only a rough estimation of the size of the user stories
  - try to level the size of the user stories
  - Divide larger into smaller ones



- Measure process parameters, e.g., average cycle time
  - E.g. ”After committing to a user story, it takes in average a week to have the user story finished”
- User average_cycle_time and WIP (Work In Progress) Limit to determine the capacity of the process and thus throughput

Example of a Kanban board for the exam project

- https://trello.com/b/iO29C07w/02161-example



Contents

What does this function do?

qsort sorting QuickSort

```
public List<Integer> f(List<Integer> list) {
   if (list.size() <= 1) return list;

   int p = list.elementAt(0);

   List<Integer> l1 = new ArrayList<Integer>();
   List<Integer> l2 = new ArrayList<Integer>();
   List<Integer> l3 = new ArrayList<Integer>();

   g(p,list,l1,l2,l3);

   List<Integer> r = f(l1);

   r.addAll(l2);
   r.addAll(f(l3));

   return r;
}

public void g(int p, List<Integer> list,
              List<Integer> l1, List<Integer> l2, List<Integer> l3) {
   for (int i : list) {
      if (i < p) l1.add(i);
      if (i == p) l3.add(i);
      if (i > p) l2.add(i);
   }
}
```

What does this function do?

```
public void testEmpy() {
  int[] a = {};
  List<Integer> r = f(Array.asList(a));
  assertTrue(r.isEmpty());
}

public void testOneElement() {
  int[] a = { 3 };
  List<Integer> r = f(Array.asList(a));
  assertEquals(Array.asList(3),r);
}

public void testTwoElements() {
  int[] a = {2, 1};
  List<Integer> r = f(Array.asList(a));
  assertEquals(Array.asList(1,2),r);
}

public void testThreeElements() {
  int[] a = {2, 3, 1};
  List<Integer> r = f(Array.asList(a));
  assertEquals(Array.asList(1,2,3),r);
}
...
```

What does this function do?

```
List<Integer> f(List<Integer> a)
```

Precondition: a is not **null**

Postcondition: For all $result, a \in$ List<Integer>:

$result == f(a)$

if and only if

isSorted(result) and sameElements(a,result)

where

isSorted(a) if and only if

for all $0 \leq i, j < a.size()$**:**

$i \leq j$ **implies** $a.get(i) \leq a.get(j)$

and

sameElements(a,b) if and only if

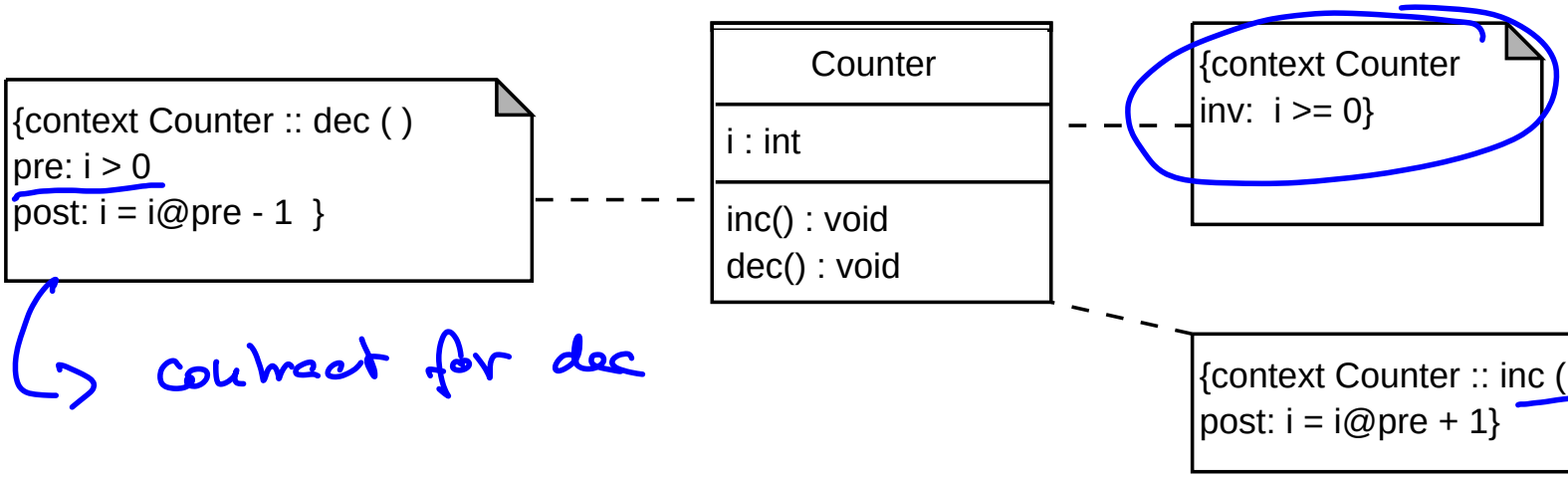
for all $i \in Integer$**:** $count(a, i) = count(b, i)$

$[\ 3 \leq 10\ ]$

$i \leq j$

Example Counter

formal specification



client

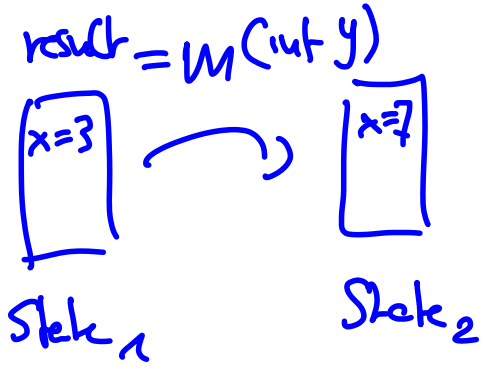
```
public T n(T1 a1, .., Tn an, Counter c)
...
// Here the precondition of c has to hold
// to fulfil the contract of Counter::dec
c.dec();
// Before returning from dec, c has to ensure the
// postcondition of dec
...
```

Design by contract

- Name invented by Bertrand Meyer (Eiffel programming language) for pre-/post-condition based formal methods applied to object-oriented designs/languages
- Pre-/post-conditions were invented by Tony Hoare and Robert W. Floyd

Contract for a method

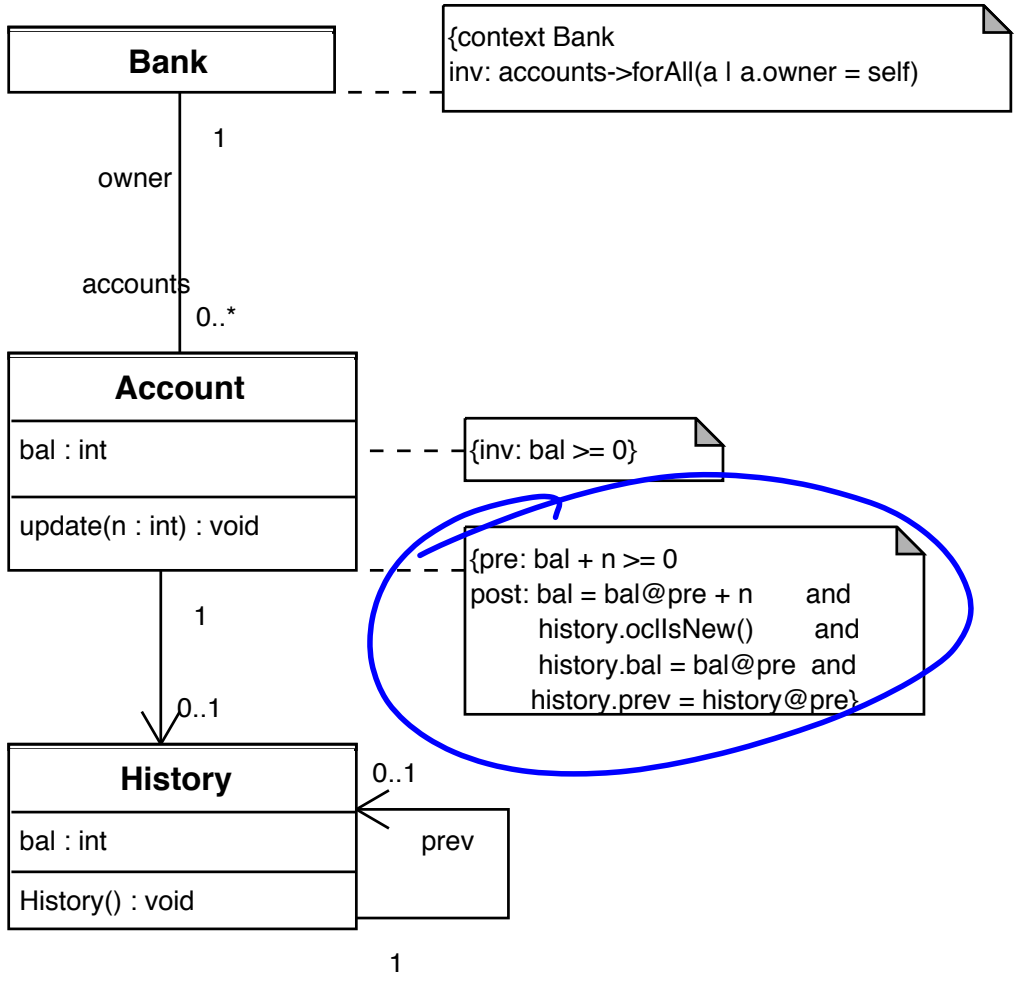
- precondition: a boolean expression over the state of the object and arguments *before* the execution of the method
- postcondition: a boolean expression over the state of the object and arguments *before* the execution of a method *and* the result of the method and the state of the object *after* the execution of the method



Contract between Caller and the Method

- Caller ensures precondition
- Method ensures postcondition

Bank example with constraints

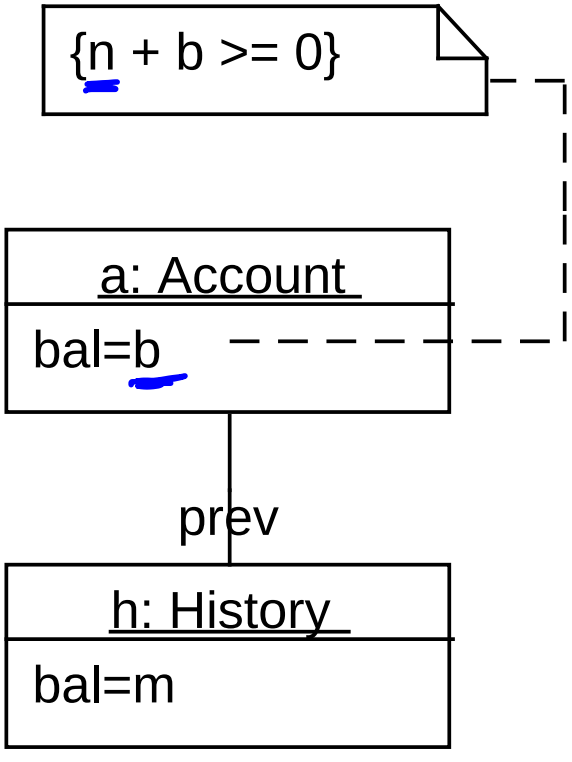


Update operation of Account

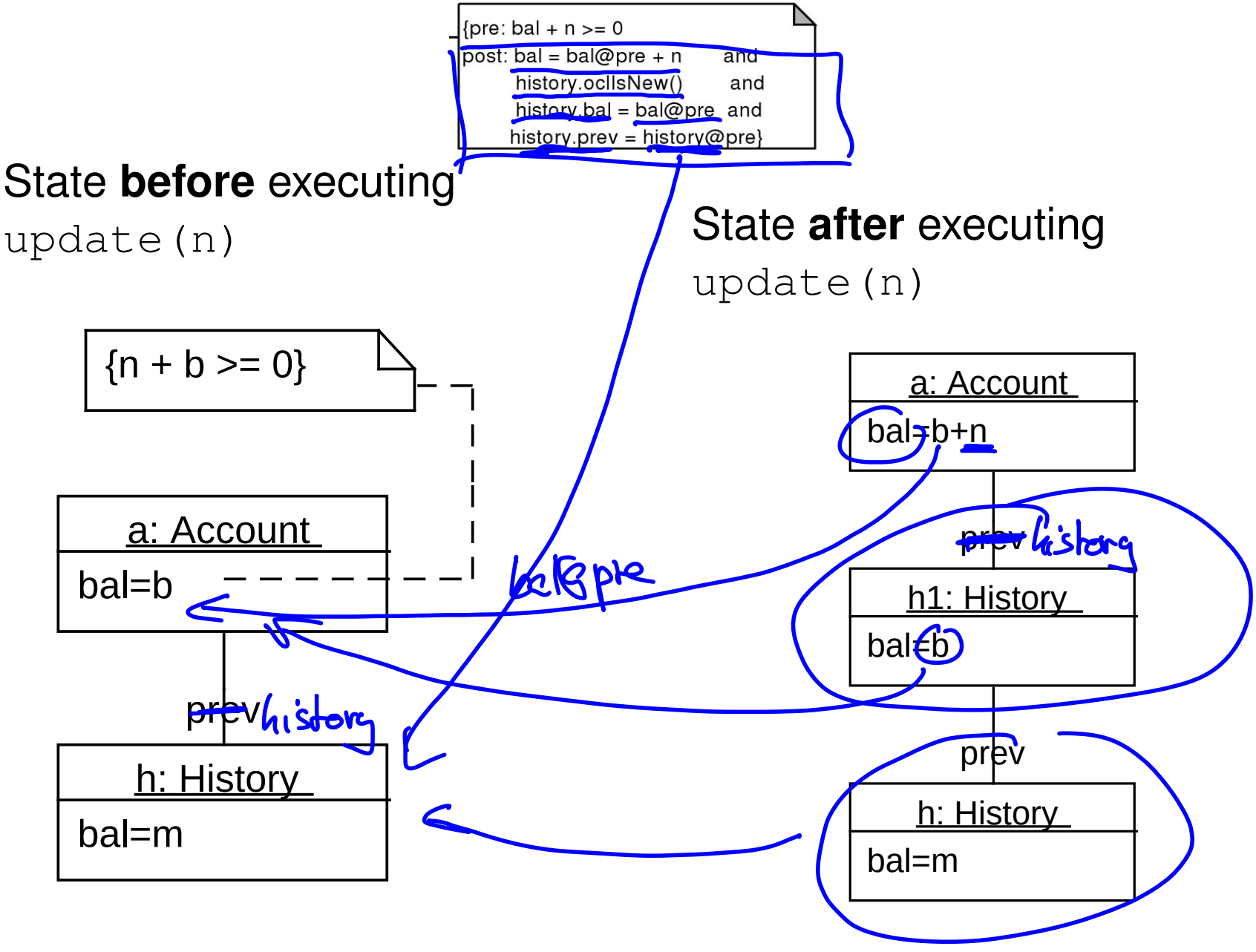
pre condition

{pre: bal + n >= 0
post: bal = bal@pre + n and
history.oclIsNew() and
history.bal = bal@pre and
history.prev = history@pre}

State **before** executing
`update(n)`



Update operation of Account



Example

Object Constraint Language

OCL expression for set union

formal language for UML

```
LibraryApp::addMedium(Medium m)
pre: adminLoggedIn
post: medium = medium@pre->including(m) and
        medium.library = this
```

```
LibraryApp::search(String string) : List<Medium>
post: result = medium->select(m |
                    m.title.contains(string) or
                    m.autor.contains(string) or
                    m.signature.contains(string))
   medium = medium@pre
```

```
User::borrowMedium(Medium m)
pre: borrowedMedium->size < 10
       and m != null
       and not(borrowedMedium->exists(m' | m'.isOverdue))
post: m.borrowDate = libApp.getDate() and
        borrowedMedium = borrowedMedium@pre->including(m)
```

Implementing DbC with assertions

- Many languages have an assert construct. In Java:
  `assert bexp;` or `assert bexp:string;`
- Contract for Counter::dec(i:int)
  - Pre: $i > 0$
  - Post: $i = i@pre - 1$

```
void dec() {
  assert i > 0 : "Precondition violated"; // Precondition
  int iatpre = i; // Remember the value of the counter
                  // to be used in the postcondition
  i--;
  assert i == iatpre-1 : "Postcondtion violated"; // Postcondition
}
```

- assert and assertTrue are not the same!
  - assertTrue: JUnit, used in a test
  - assert: assertion used inside a method.

Implementing DbC in Java

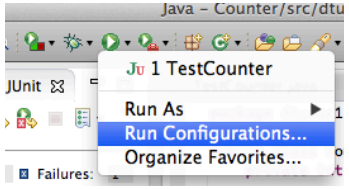
Pre: $args \neq null$ and $args.length > 0$

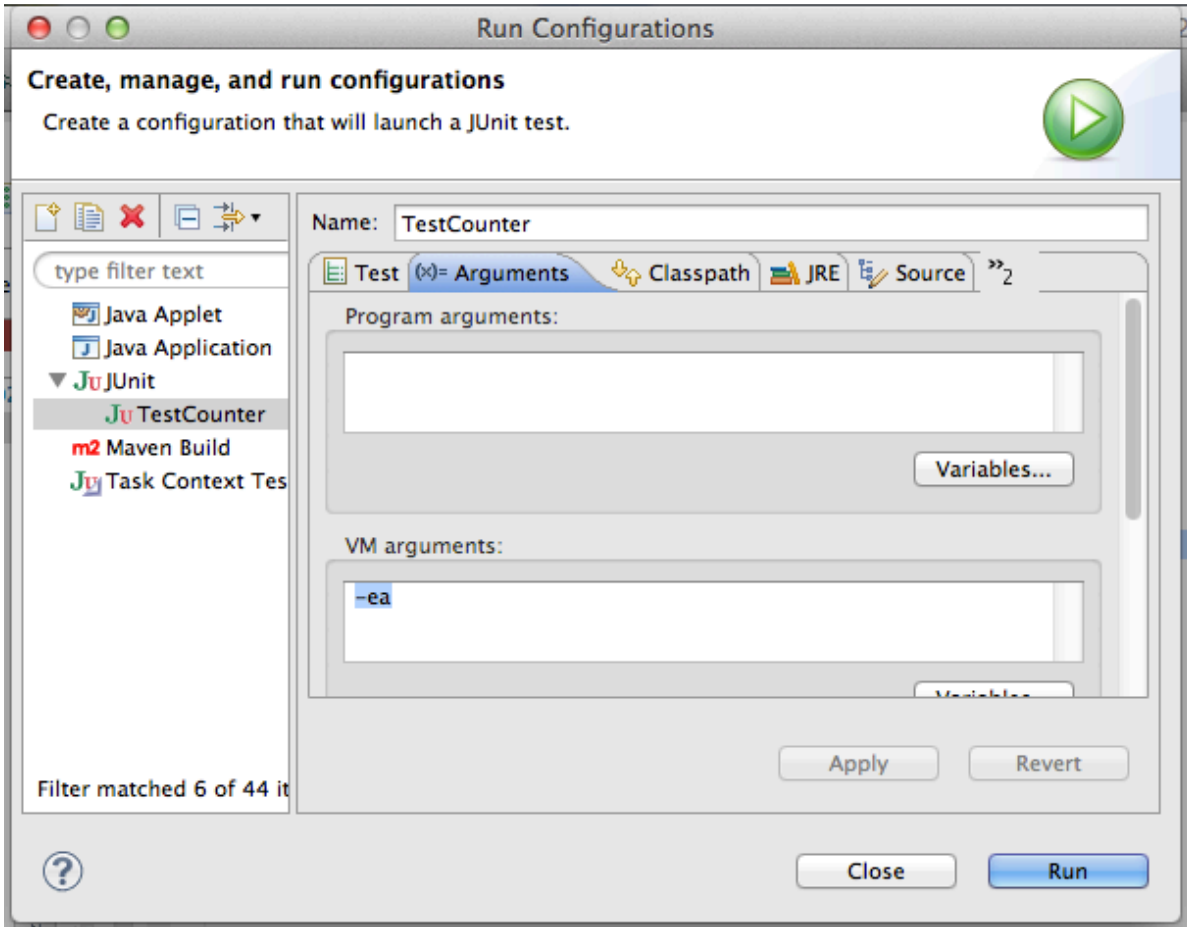
Post: $\forall n \in args : min \leq n \leq max$

```
public class MinMax {
  int min, max;

  public void minmax(int[] args) throws Error {
    assert args != null && args.length != 0;
    min = max = args[0];
    for (int i = 1; i < args.length; i++) {
      int obs = args[i];
      if (obs > max)
        max = obs;
      else if (min < obs)
        min = obs;
    }
    assert isBetweenMinMax(args);
  }

  private boolean isBetweenMinMax(int[] array) {
    boolean result = true;
    for (int n : array) {
      result = result && (min <= n && n <= max);
    }
    return result;
  }
```

Important

- Assertion checking is switched off by default in Java
  1) Use `java -ea Main` to enable assertion checking
  2) In Eclipse





Assertions

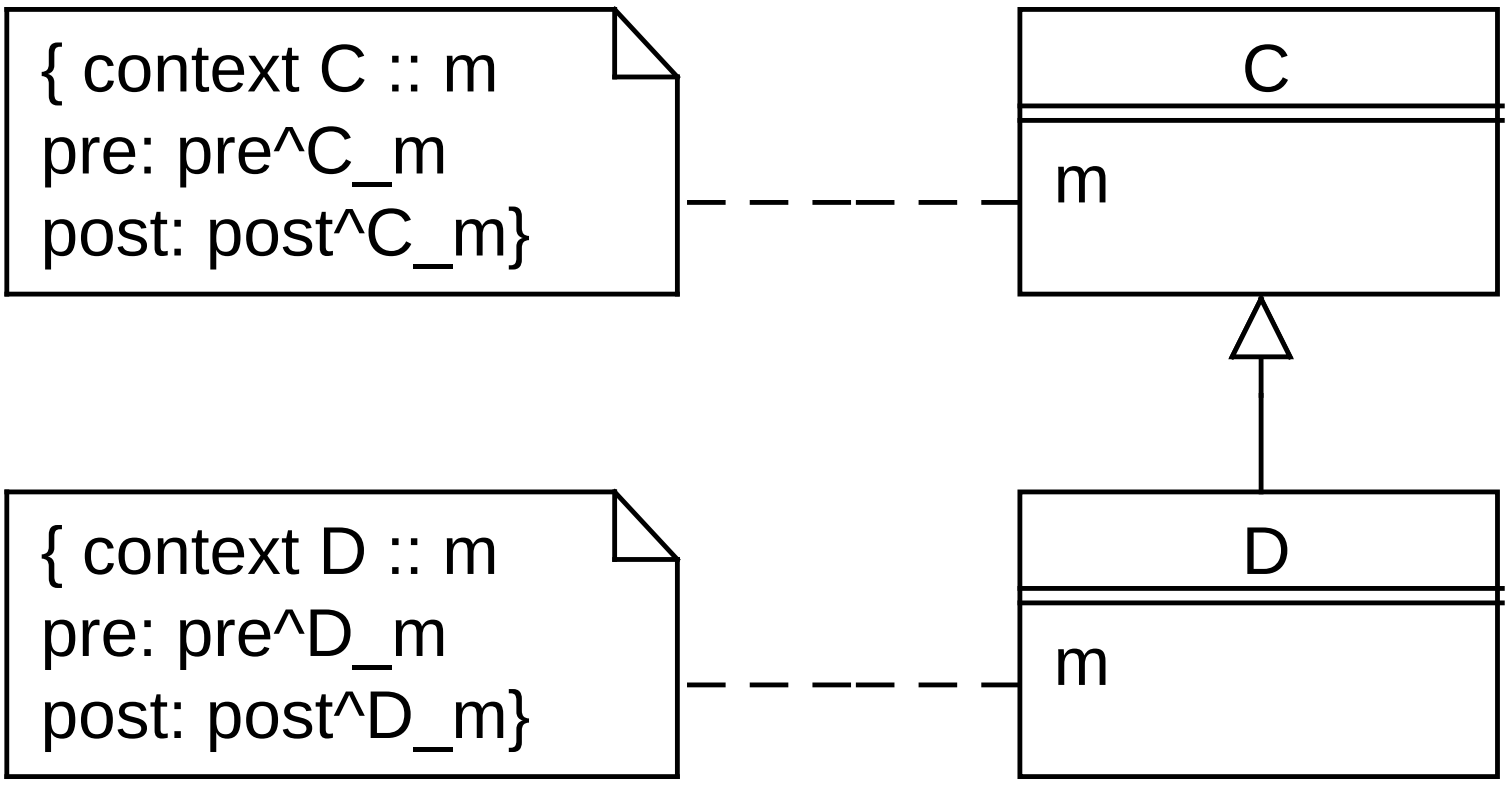
- Advantage
  - Postcondition is checked for each computation
  - Precondition is checked for each computation
- Disadvantage
  - Checking that a postcondition is satisfied can take as much time as computing the result

  $\rightarrow$ Performace problems
  - Solution:
    - Assertion checking is switched on during developing, debugging and testing and switched off in production systems

Assertion vs. Tests

- Assertion
  - Checks all computations (as long as assertion checking is switched on)
  - Checks also for contract violations from the client (i.e. precondition violations)
- Tests
  - Only checks test cases (concrete values)
  - Cannot check that the clients establish the precondition

Contracts and inheritance

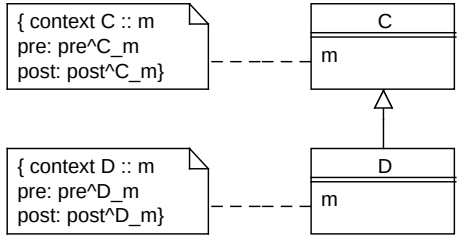


Contracts and Inheritance

Liskov / Wing Substitution principle:

At every place, where one can use objects of the superclass C, one can use objects of the subclass D

```
public T n(C c)
    ...
    // n has to ensure PreˆC_m
    c.m();
    // n can rely on PostˆC_m
    ...
```

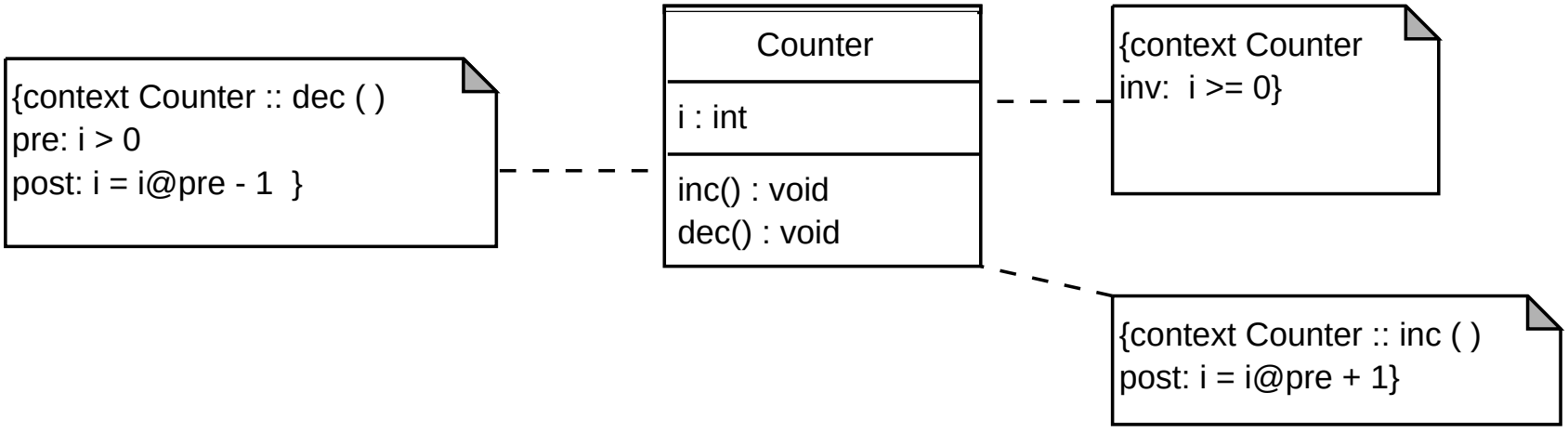


$t.n(new\ C())$ vs. $t.n(new\ D())$.

→ $Pre_m^C \implies Pre_m^D$ *weaken* precondition

→ $Post_m^D \implies Post_m^C$ *strengthen* postcondition

Invariants: Counter



- Methods
  - assume that invariant holds
  - ensure invariants
- When does an invariant hold?
  - After construction
  - After each *public* method

Invariants

- Contstructor has to ensure invariant

```
public Counter() {
   i = 0;
   assert i >= 0;  // Invariant
}
```

- Operations ensure and assume invariant

```
void dec() {
   assert i >= 0; // Invariant
   assert i > 0; // Precondition
   int iatpre = i; // Remember the value of the counter
                   // to be used in the postcondition
   i--;
   assert i == iatpre-1; // Postcondition
   assert i >= 0; // Invariant
}
```

Defensive Programming

- Can one trust the client to ensure the precondition?

```
void dec() { i--; }
```

- Depends if the programmer controls the client or not
  - e.g. if dec is private, only the programmer of the method can call dec
  - if dec is publick, potentially others can call the method

Defensive Programming

- If one does not trust the client
- Check explicitly that the precondition of a method is satisfied
  - Either

```
void dec() { if (i > 0) { i--; } }
```

  - Or

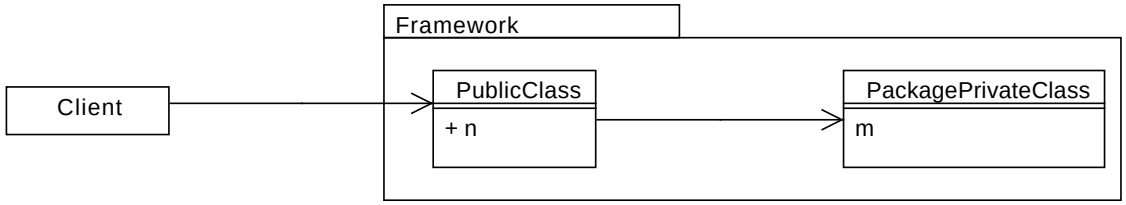
```
void dec() {
    if (i <= 0) {
        throw new Exception("Dec not allowed ...");
    }
    i--;
}
```

- Don't rely on the `assert` statement.
  - Why?

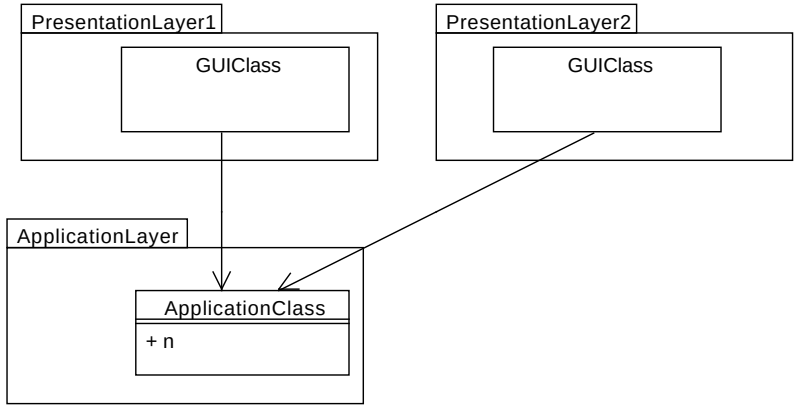
```
void dec() {
    assert i <= 0;
    i--;
}
```

Defensive Programming

- Use defensive programming with public methods
- Use asserts with private or package private methods
- For example public method of a library



- Public method of a class in the application/domain layer



Next week

- Principles of Good Design
- Design Patterns