Programming Language Design

2015

Week #9: Product lines, feature-orientation, context-orientation

Instructor: Hidehiko Masuhara

Q(1/2): Design classes for representing sets (10 min.)

- Common operations: insertion and deletion of an element
- Variations
  - ListSet: uses a linked-list
    - O(1) insertion, O(*n*) deletion
  - TreeSet: uses a binary tree
    - O(log*n*) insertion and deletion
    - additional op.: checking element containment
  - ConcurrentListSet: ListSet that can be concurrently accessed from multi-threads
  - ConcurrentTreeSet: TreeSet that can be...multi-threads)TreeSet
  - CountableListSet: ListSet with operation to determine # of elements
  - CountableTreeSet: TreeSet with ... # of elements
  - CountableConcurrentListSet: ibid.
  - CountableConcurrentTreeSet: ibid.

Goal: avoid code duplication

Q(2/2): Discuss how to optimize ListSet (5 min.)

- when (1) many elements are sequentially inserted, and then (2) elements are inserted and deleted concurrently
- by not using locks during (1)
- (if needed) by modifying definition of ListSet and definition of the code for (1) and (2)
- discuss or show a class design

```
//setup
ListSet students = new ListSet();
ListSet teachers = new ListSet();
readPeople(studentDB, students);
readPeople(teacherDB, teachers);
//main
new ThreadPool(10){
    void run() {
      ...students.insert(...)...
      ...teachers.insert(...)... }}
}
```

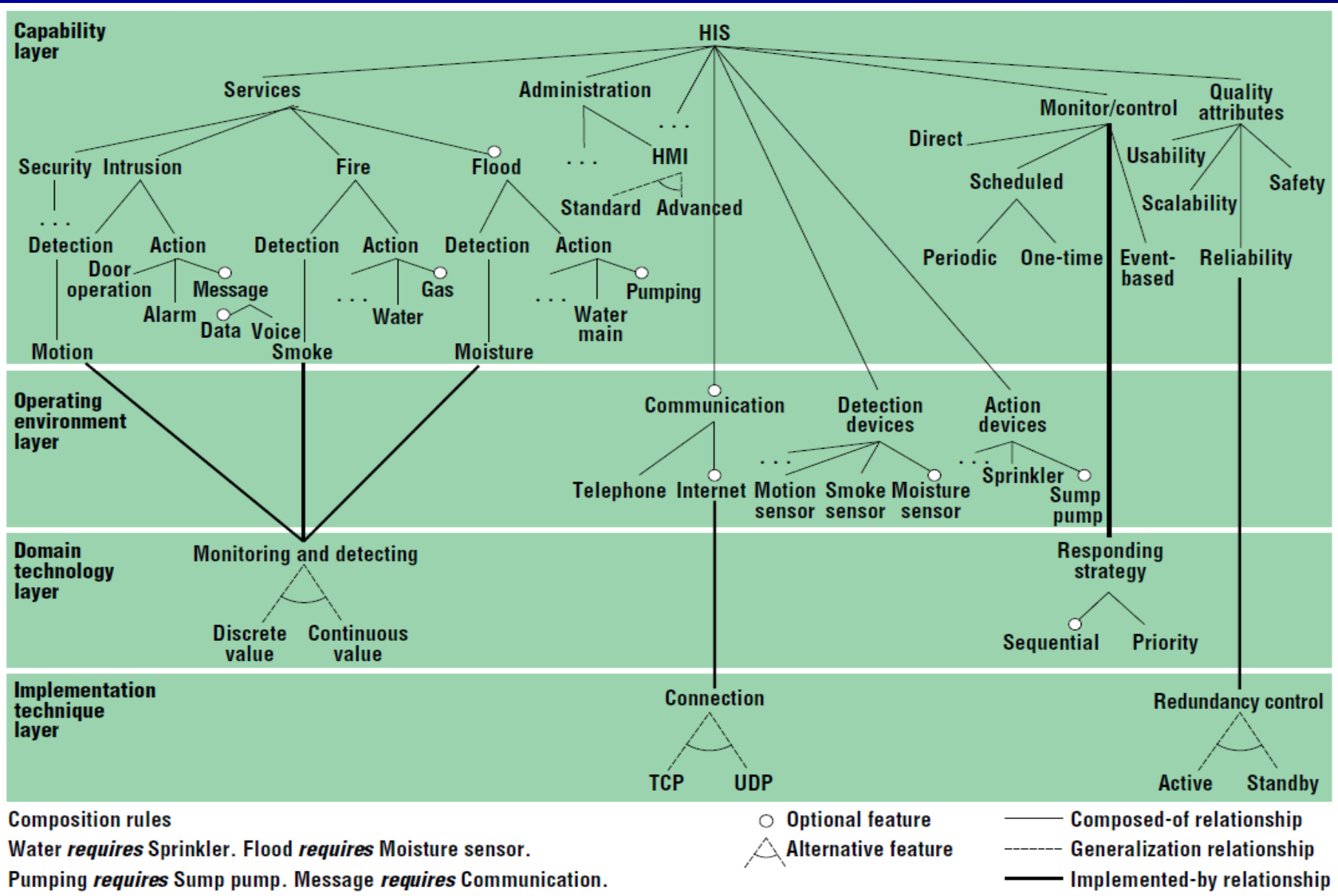
Product lines

- Definitions
  - *a range of similar products or services that are sold by the same company, with different features and different prices (Cambridge Dictionary Online)*
  - *a group of related products marketed by the same company (Collins)*
- Examples?
- How are they important in production?
  - design reuse

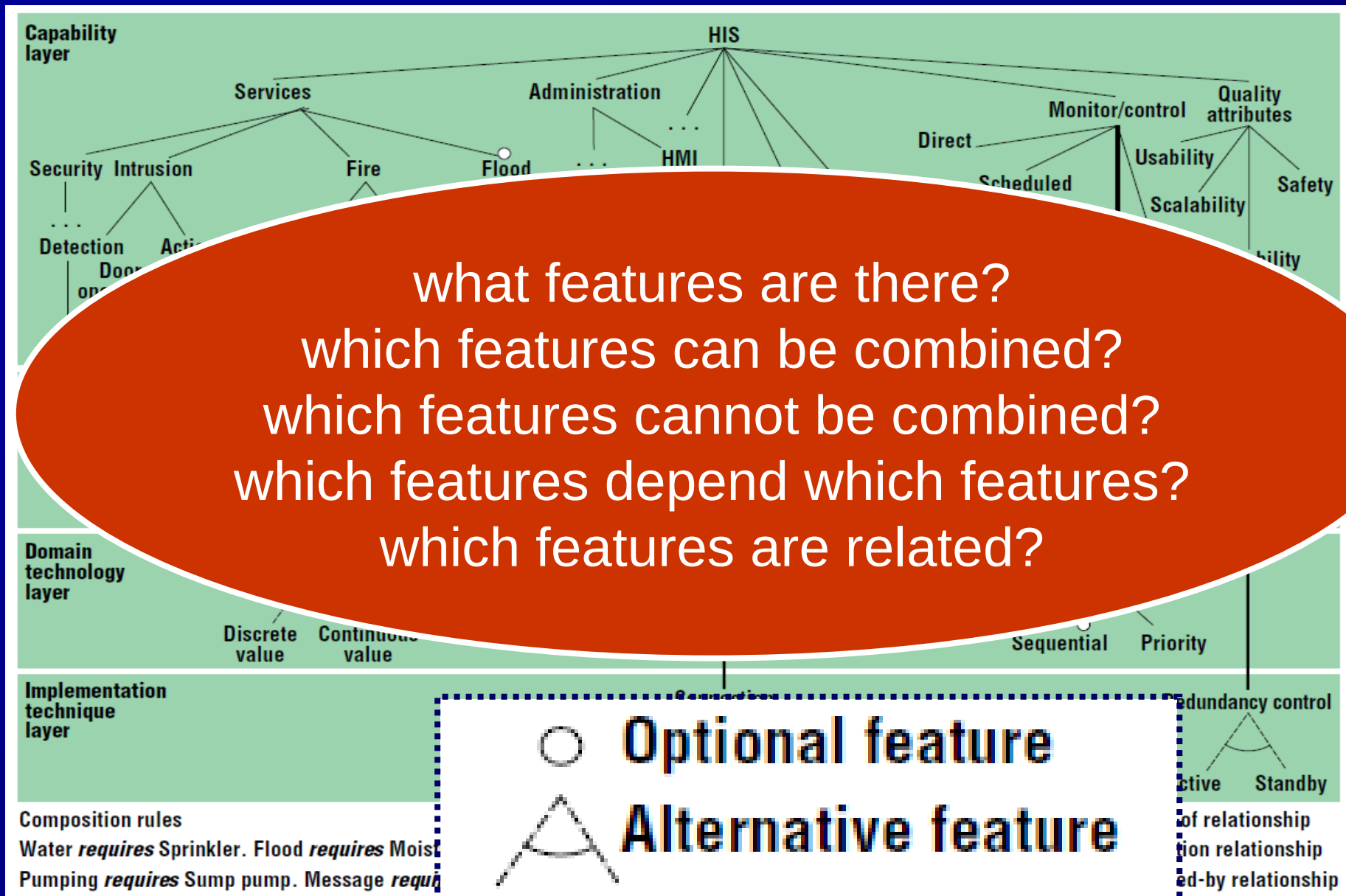
Feature-oriented product line engineering [KLP02]

- Domain: product lines development
- Goal: making outcomes more reusable
- Methods of: analysis, modeling and design
- Challenge: variability management
- Orientation: ***features*** in products
  - analyze dependency between features
  - not necessarily ***object***-oriented

Feature analysis example: Home Integration System [KLD02]



Feature analysis example: Home Integration System [KLD02]



How do we implement software product lines?

Realizing product lines in software: #ifdef

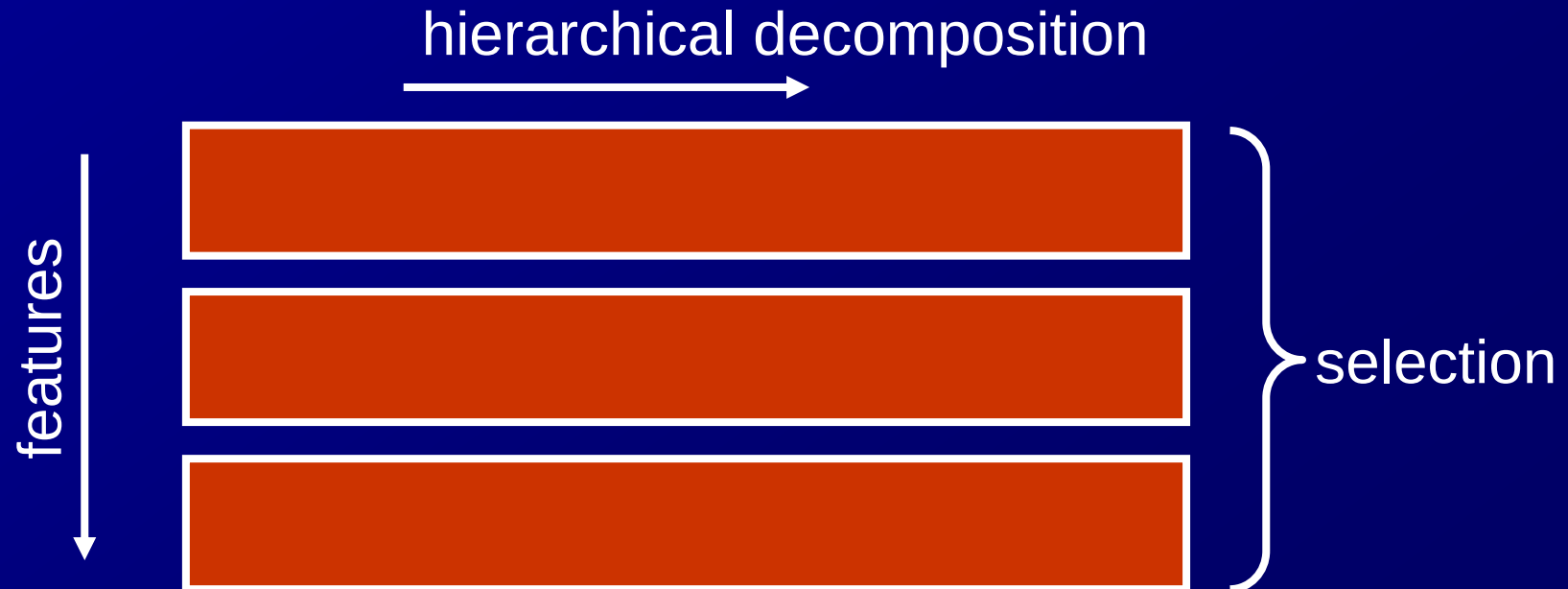
- Linux 3.2 has 12000 "features" [Reinhard14]
- Problems
  - hard to read
  - can only be verified after preprocessing
  - 2^n combinations

```
static int ___rep_queue_filedone(dbenv, rep, rfp)
  DB_ENV *dbenv;
  REP *rep;
  ___rep_fileinfo_args *rfp; {
#ifndef HAVE_QUEUE
  COMPQUIET(rep, NULL);
  COMPQUIET(rfp, NULL);
  return (___db_no_queue_am(dbenv));
#else
  db_pgno_t first, last;
  u_int32_t flags;
  int empty, ret, t_ret;
#ifdef DIAGNOSTIC
  DB_MSGBUF mb;
#endif
  // over 100 further lines of C code
#endif
}
```

uses of #ifdef in Oracle Berkeley DB ([KA13])

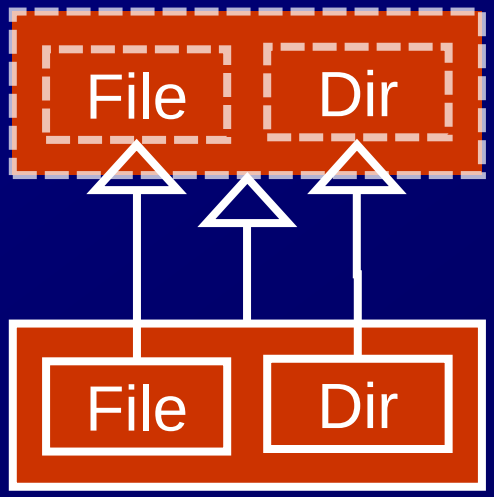
Feature-oriented programming

- Goal
  - Cohabitation of feature-orientation and hierarchical decomposition
  - Select and combine features
- Approach: Layer (=feature)based modularization

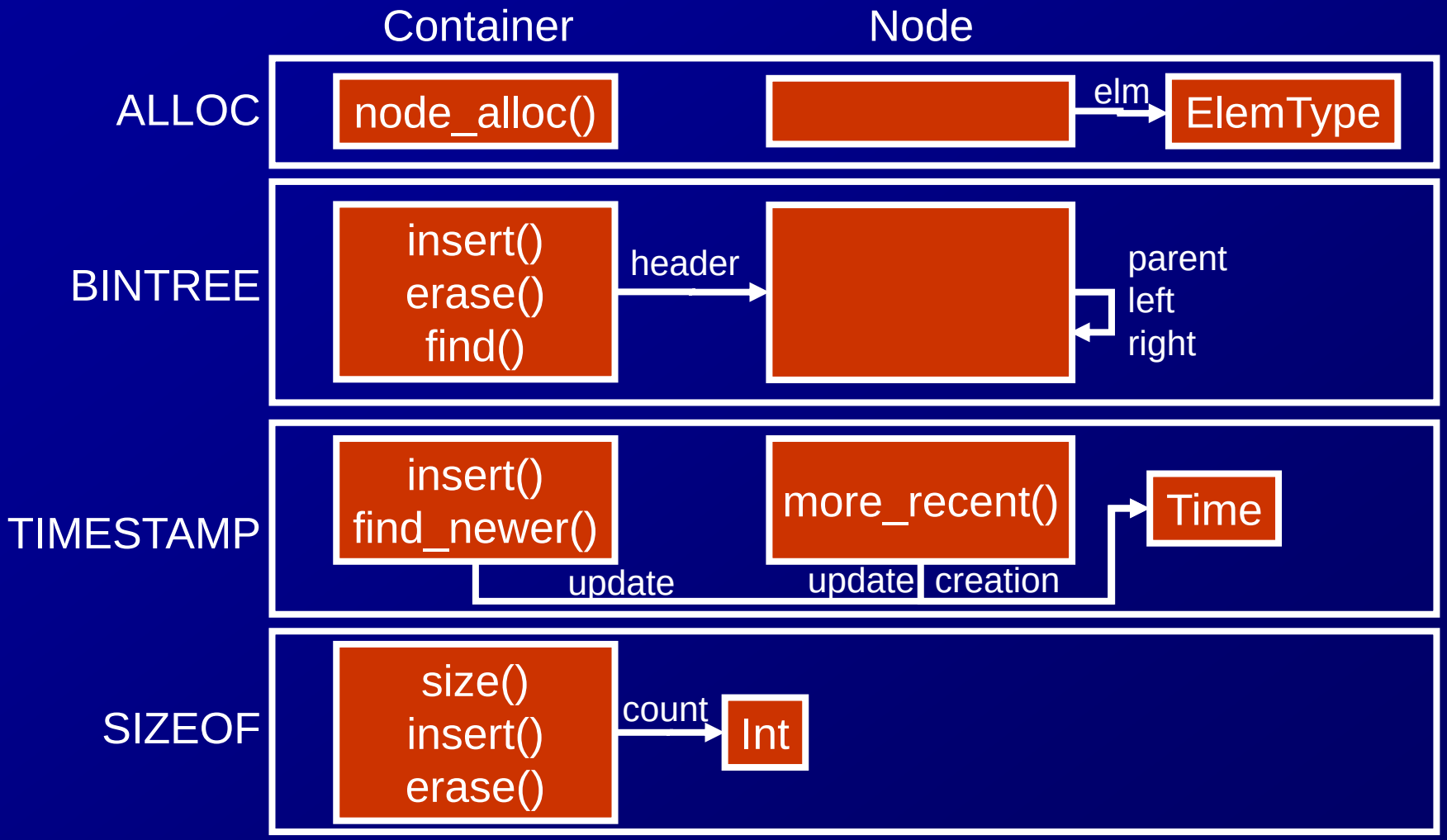


FOP with mixin layers [YB98]

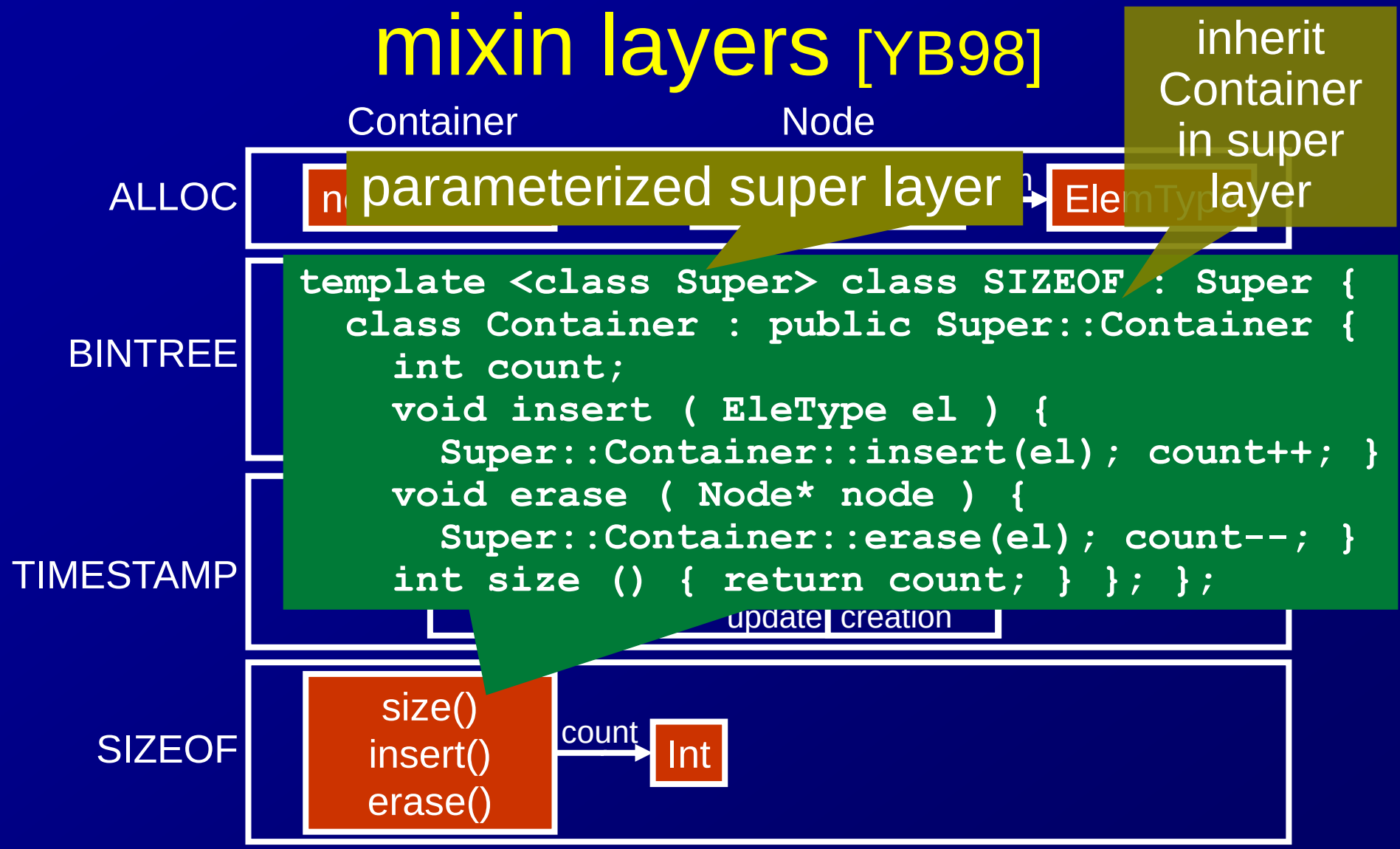
- (mixin: a class whose super is parameterized)
- mixin layer: nested mixins
  - outer mixin = featuers
  - inner class/mixin = hierachical decomposition
    - a subclass of the same name class in outer mixin's parent
- feature composition = outer mixin composition



FOP Collection library with mixin layers [YB98]



FOP Collection library with mixin layers [YB98]



Q(2/2): Discuss how to optimize ListSet (10 min.)

- when (1) many elements are sequentially inserted, and then (2) elements are inserted and deleted concurrently
- by not using locks during (1)
- (if needed) by modifying definition of ListSet and definition of the code for (1) and (2)
- discuss or show a class design

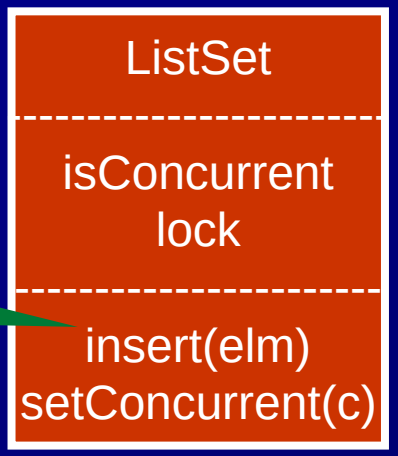
```
//setup
ListSet students = new ListSet();
ListSet teachers = new ListSet();
readPeople(studentDB, students);
readPeople(teacherDB, teachers);
//main
new ThreadPool(10){
    void run() {
      ...students.insert(...)...
      ...teachers.insert(...)... }}
}
```

Context-oriented programming [HCN08]

- Background
  - Things behave differently based on their contexts ---context-dependent behaviors---
  - to model in OOPLs
    - implement context-dependent behaviors in one class/method → many conditional braches
    - define a class for each context
      → one object no longer represents one "thing"

Modeling context-dependent behaviors in OOPL (1 class)

```
if (isConcurrent) lock.get();
head = new Cons(elm,head);
if (isConcurrent) lock.release();
```



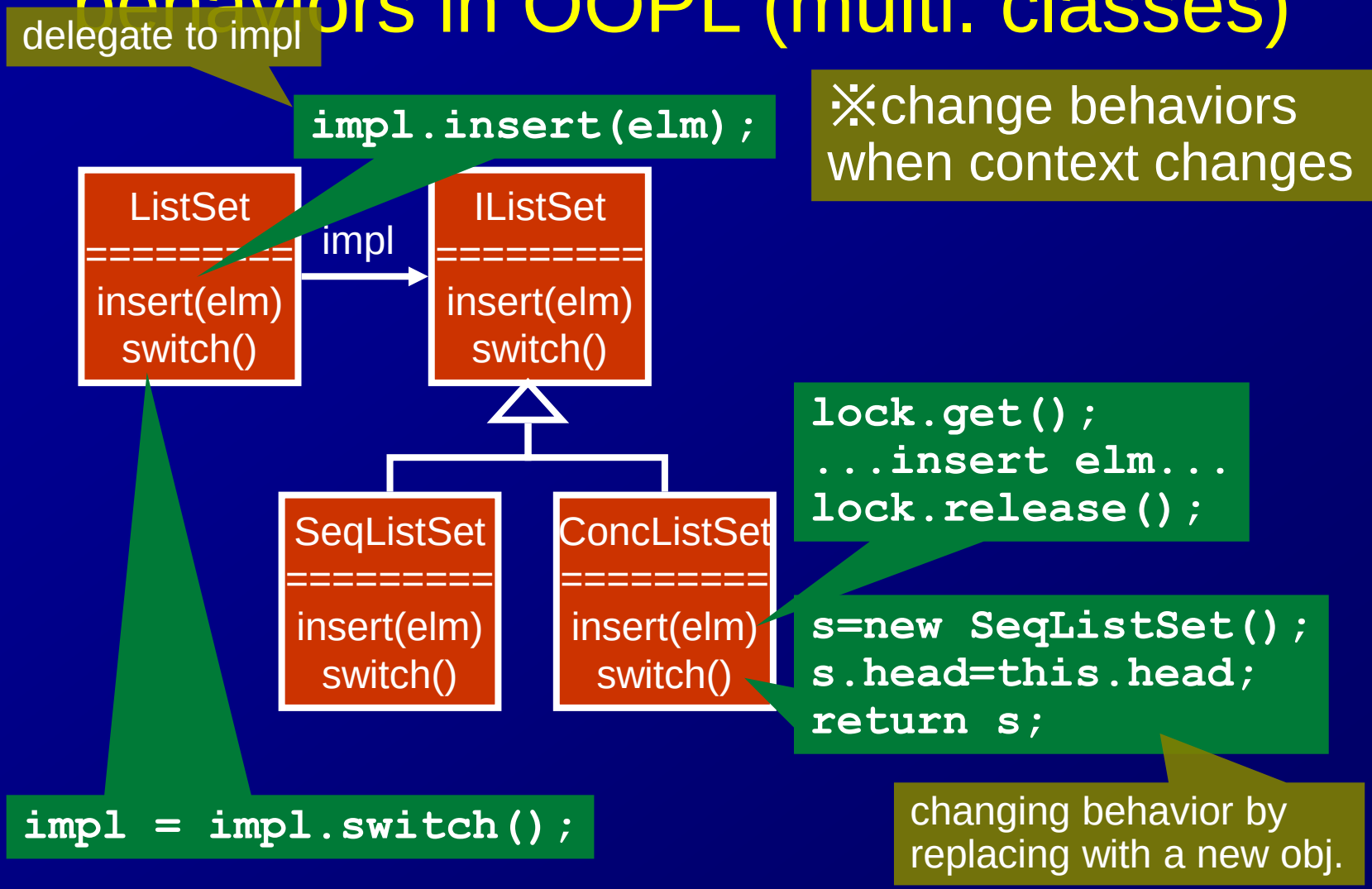

```
ListSet students = new ListSet();
ListSet teachers = new ListSet();
students.setConcurrent(false);
readPeople(studentDB, students);
teachers.setConcurrent(false);
readPeople(teacherDB, teachers);
students.setConcurrent(true);
teachers.setConcurrent(true);
new ThreadPool(10){
  void run() {
    ...students.insert(...)...
    ...teachers.insert(...)... }}
```

initialize sequentially

switch behaviors when context changes

update parallely

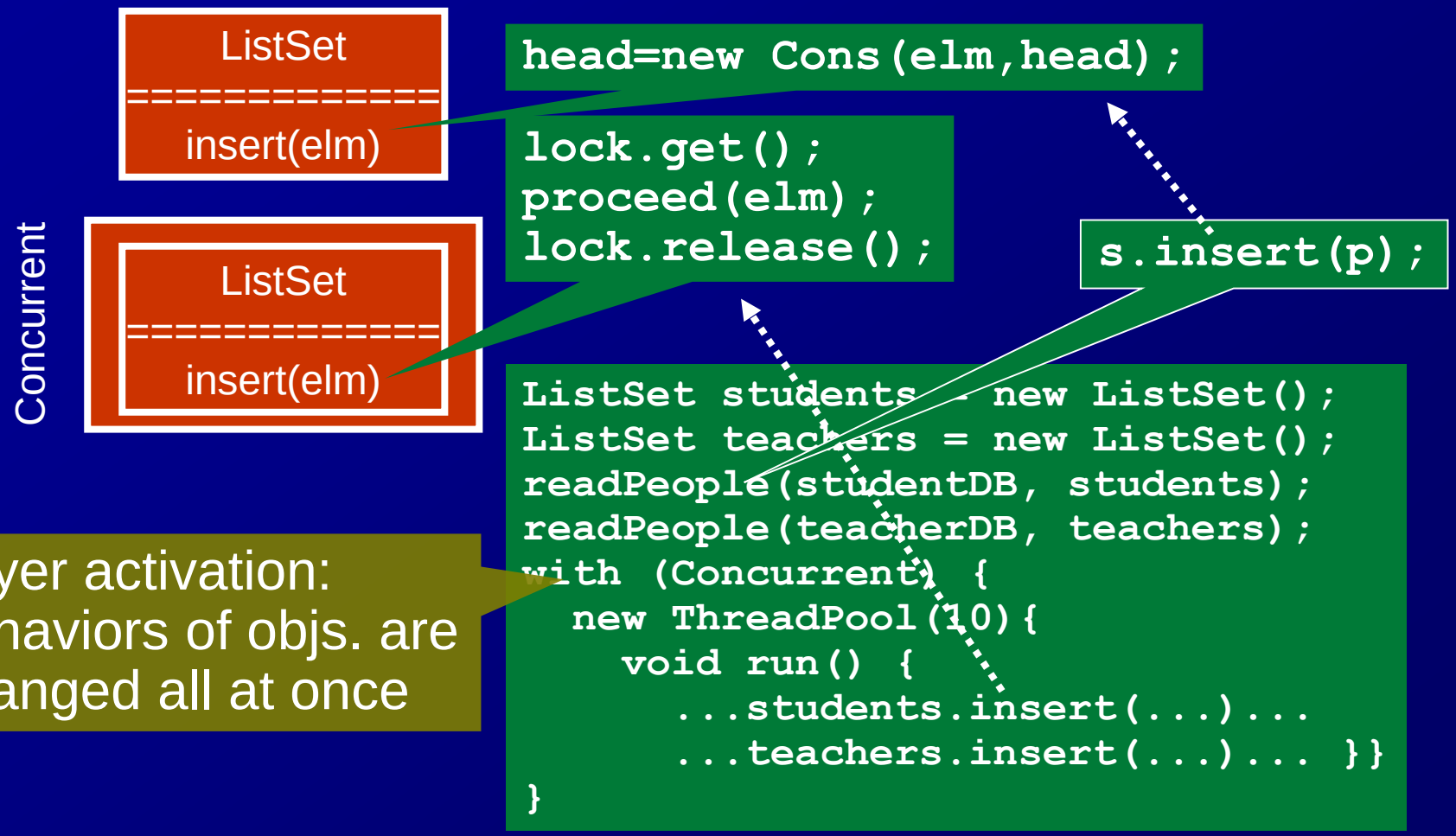
Modeling context-dependent behaviors in OOPL (multi. classes)



Context-oriented programming [HCN08]

- Difference in situations = context-dependency
- Define context-dependent behaviors in ***layers***
  - similar to mixin layers
  - overriding same name method in same name class
  - reusing overridden code by ***proceed*** (cf. super)
- Methods in a layer override only when the layer is active
  - controlled through activation commands
  - multiple active layers → multiple-overriding

Example of context-oriented programming



References

[KA13] Kästner, Christian, and Sven Apel. "Feature-Oriented Software Development." Generative and Transformational Techniques in Software Engineering IV. Springer Berlin Heidelberg, 2013. 346-382.

[Reinhard14] Tartler, Reinhard, et al. "Static analysis of variability in system software: The 90,000# ifdefs issue." Proc. USENIX Conf. 2014.

[KLP02] Kang, Kyo C., Jaejoon Lee, and Patrick Donohoe. "Feature-oriented product line engineering." IEEE software 19.4 (2002): 58-65.

[YB98] Smaragdakis, Yannis, and Don Batory. "Implementing layered designs with mixin layers." ECOOP'98—Object-Oriented Programming. Springer Berlin Heidelberg, 1998. 550-570.

[HCN08] Hirschfeld, Robert, Pascal Costanza, and Oscar Nierstrasz. "Context-oriented programming." Journal of Object Technology 7.3 (2008).