LECTURE 1

THURSDAY SEPTEMBER 5

Course Learning Outcomes

EECS1090

CLO1 Describe software specifications via Design by Contract, including the use of preconditions, postconditions, class invariants, as well as loop variants and invariants.

CLO2 Implement specifications with designs that are correct, efficient, and maintainable.

CLO3 Develop systematic approaches to organizing, writing, testing, and debugging software.

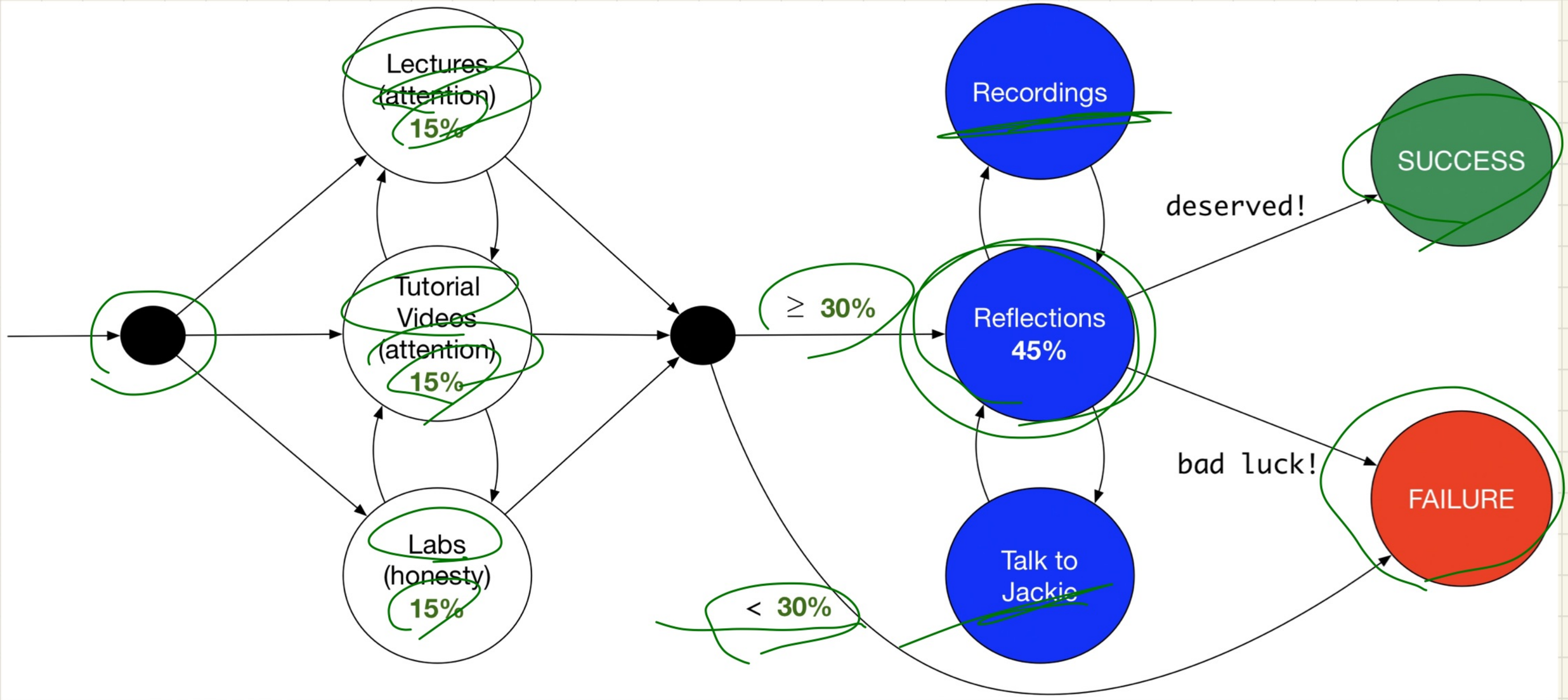
CLO4 Develop insight into the process of moving from an ambiguous problem statement to a well-designed solution.

CLO5 Design software using appropriate abstractions, modularity, information hiding, and design patterns.

CLO6 Develop facility in the use of an IDE for editing, organizing, writing, debugging, documenting designs, and the ability to deploy the software in an executable form.

CLO7 Write precise and concise software documentation that also describes the design decisions and why they were made.

Surviving through this course



Software Development Cycle

EECS 4312

331

2011
2020

Requirement

- Natural Language (incomplete, ambiguous, contradicting)
  - working payment system
  - easy to use and 4-phase authentication (face, touch, code, pass)
  - web interface and deployable on mobile devices
- elicitation

↓

Design

- blueprints
- not necessarily executable & testable

↓

Implementation

- API given
- efficiency (data structures & algorithms)
- unit tests

↓

Release

- Customer's acceptance?
- return?

Unit testing

- JUnit in Java
- Eiffel in Eiffel

API of methods

imp. details of methods.

Acceptance testing

- agreed interface with the customers.
- imp. design details hidden details

Client vs. Supplier in OOP

```
class Microwave {
  private boolean on;
  private boolean locked;
  void power() {on = true;}
  void lock() {locked = true;}
  void heat(Object stuff) {
    /* Assume: on && locked */
    /* stuff not explosive. */
  } }
```

```
class MicrowaveUser {
  public static void main(...) {
    Microwave m = new Microwave();
    Object obj = ???;
    [m.power(); m.lock();]
    m.heat(obj);
  } }
```

pre-state → client's ob- has to be fulfilled (on, locked exp'd)

not sure ??? may be explosive.

obj.heated()

Supplier m (Microwave)

Client this (MicrowaveUser)

post-state → supplier's ob. fulfilled

Pre-Condition:
a is sorted.
O(n)

binarySearch(int[] a) {

O(log n)

]

}

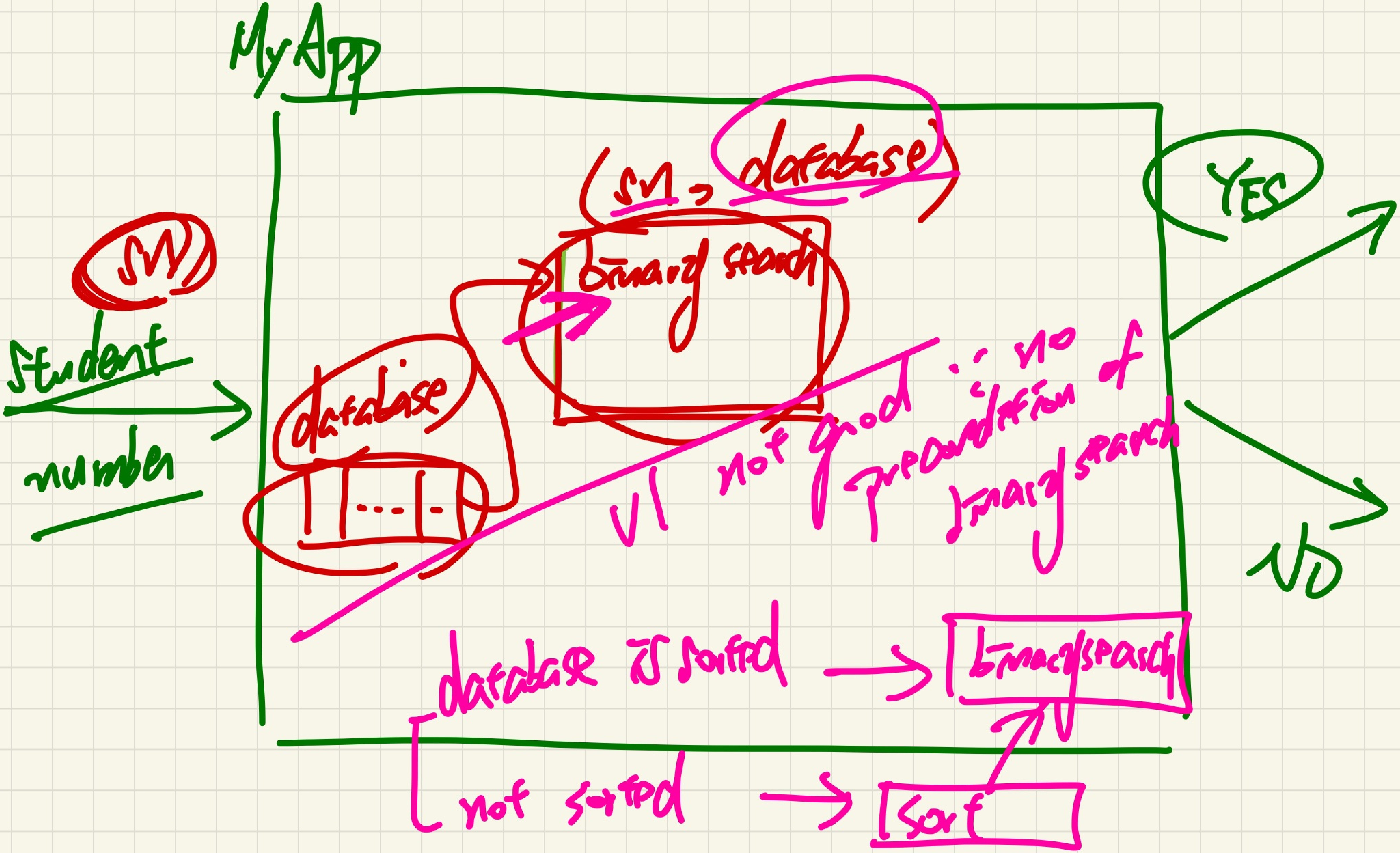
inefficient

only meant to be checked during development.

Turn all contracts off when finalizing.

application

```
{
  obj1.m1(...);
  obj2.m2(...);
  .
  .
  .
  objn.mn(...);
}
```


MyApp
SN
student number
database
binary search
(SN → database)
YES
NO
V1 not good ∵ no precondition of binary search
database is sorted → binary search
not sorted → sort