EECS3311 Software Design (Fall 2020)

Q&A - Lecture Series W11

Tuesday, December 1

observer (1-to-many)

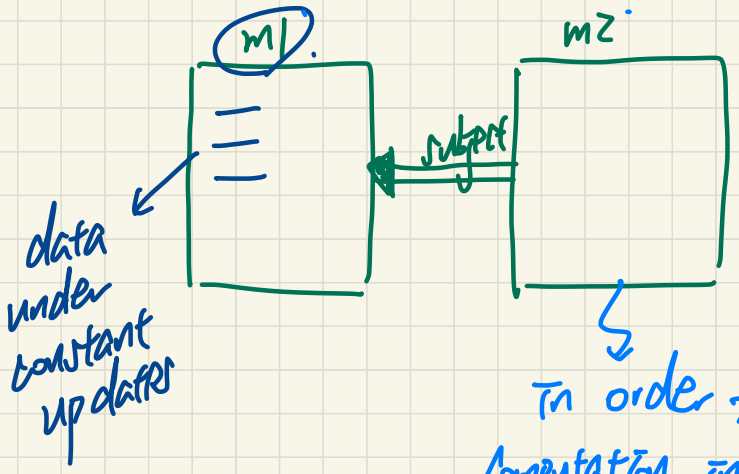
event-driven design (many-to-many)

distributed system

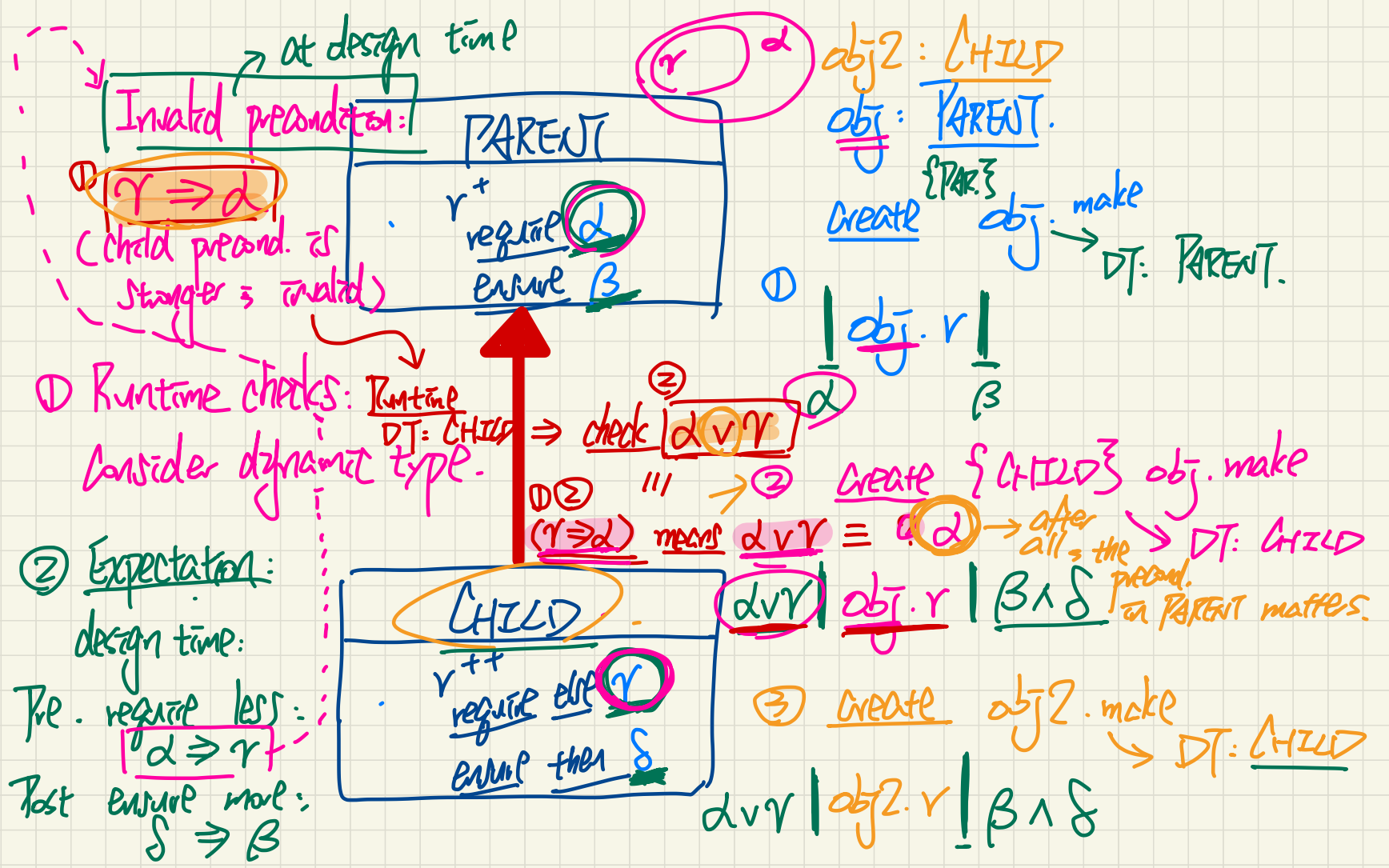
↳ 1. Geographically distributed clients and servers

(2.) As long as you can modularize your solution into design

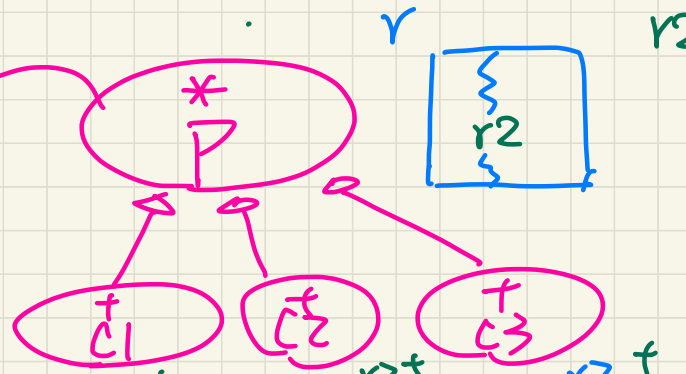
- (a) subject
- (b) observers



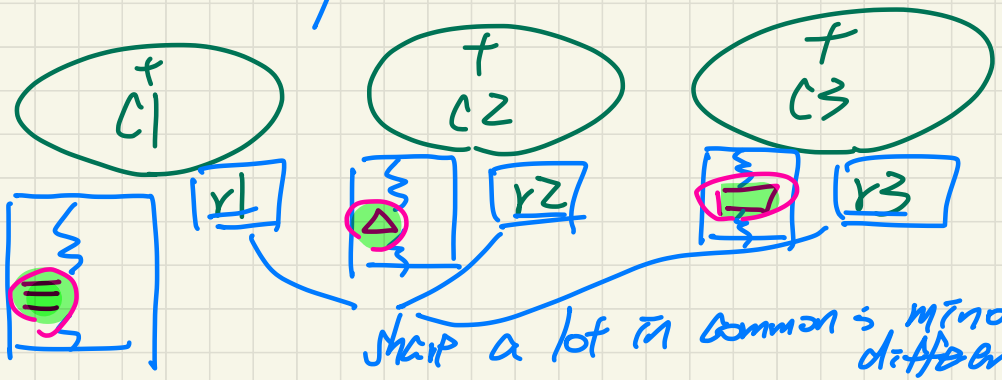
In order for computation in m2 to work, you need to get in sync with m1.


at design time
Invalid precondition:
① γ ⇒ α
(child precond. is stronger = invalid)
PARENT
r+
require α
ensure β
CHILD
r++
require else γ
ensure then δ
① Runtime checks: Runtime DT: CHILD ⇒ check α ∨ γ
Consider dynamic type.
② Expectation:
design time:
Pre. require less: α ⇒ γ
Post ensure more: δ ⇒ β
① ② (γ ⇒ α) means α ∨ γ ≡ α
α ∨ γ | obj.r | β ∧ δ
obj2 : CHILD
obj : PARENT.
{PAR.}
create obj. make → DT: PARENT.
① | obj.r | α | β
② create {CHILD} obj. make → DT: CHILD
after all, the precond. on PARENT matters.
③ create obj2. make → DT: CHILD
α ∨ γ | obj2.r | β ∧ δ

- Iterator
- singleton
- state
- template
- observer
- event-driven design
- composite
- visitor



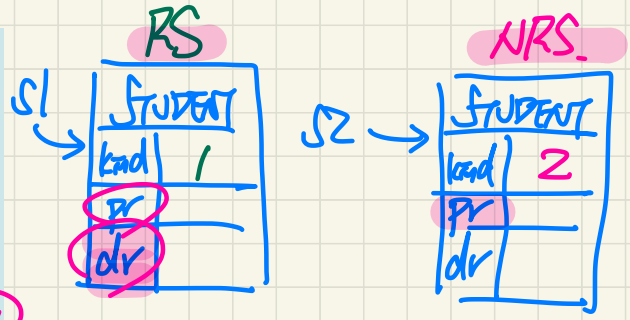
refactoring

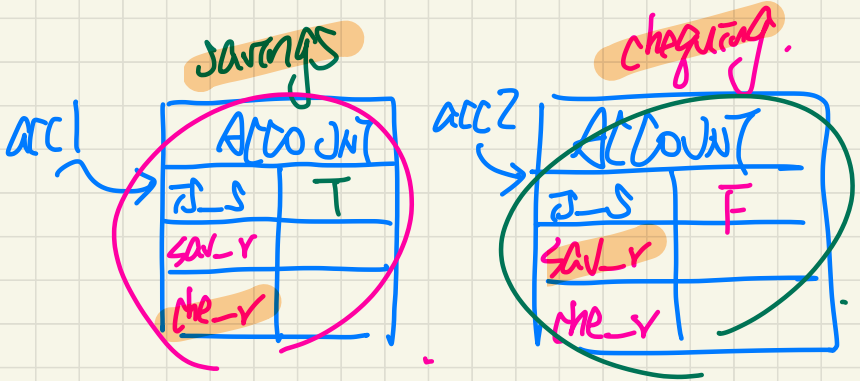


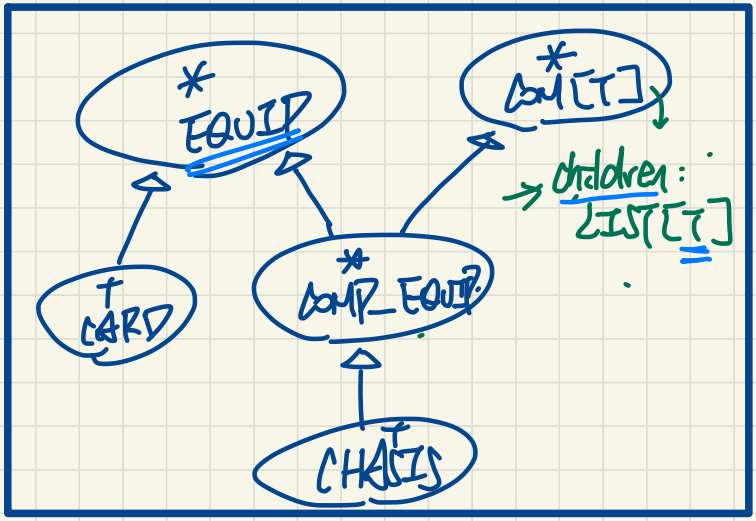
share a lot in common; minor differences.

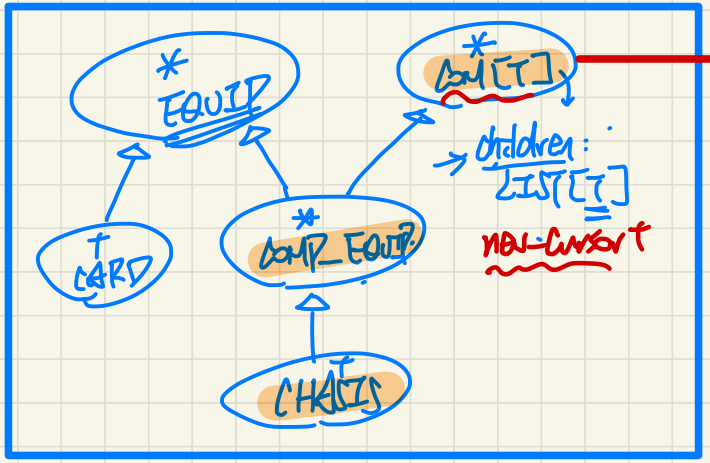
```
class ACCOUNT
create make
  feature -- Attribute
    is_savings: BOOLEAN -- false means it's a chequing account
  feature -- Commands
    make (savings_or_not: BOOLEAN)
      do
        is_savings := savings_or_not
      end
  -- other commands omitted
end
```

withdraw_from_savings.
withdraw_from_chequing









ITERABLE

Client

c : CHASIS

- initialize 'c'
- add various equip into it.

across c.children is equip

loop

-- do something with equip

end

EQUIPMENT

c : CHASIS

improvement on design
↳ simpler for clients.

across c is equip

loop

end → equip