Start

① Calculator.Kt
- composeable (UI)
- ① takes Calc.State & renders UI
- ② assigns viewmodel::onAction function to the **onClick** param of Buttons such that:
  when we click 'q' button, {onAction (CalcAction. Number(q))} is called
- param: state, vm::onAction

CalcState.Kt
- data class
- has a: ① n1,
- ② n2, Operation: Calc Operation? → nullable

② Calculator ViewModel
- viewmodel
- ① has a func 'onAction (Calc.Action c)' that takes a '**CalcAction**' as **param**
- ② onAction: take a Calc.Action, check its type, & then depending on the type (Add, Number, Clear, etc.), **Change the application State**

③ CalculatorAction.Kt
- sealed class
- what 'object' should be created when a user interacts (onClick occurs) keeping in mind that we pass this object to 'vm::onAction (obj)'
- Ask yourself: 'what all info needs to be passed when user does an (number/clear/add) action
- has a: number, clear, delete, Operation (val op : Calc.Operation)

representation of a user interaction;
helps decipher the code of Calc.State

④ Calc Operation. Kt
- deciphering point for how to code vm::onAction (a :Calc.Action) {

  when (a) {
  is Calc.Action. operation {
  // this means a = CalcAction.operation (Calc.Operation. Add)
  // ie. a contains an (a. Op == Add)

      if (a.op is Calc.Operation.Add) {
      Change state // ─→ state. copy (n2 = n1 + n2,
                                        op = null,
                                        n2 = " ")
  }
  }

→ **Specific Analysis**

① **Calc Action . Operation :**
- every Calc.Action.Operation 'has a' 'op' param such that op: Calc.Op

- i.e. "data class Operation (val op : Calc. Operation) : CalcAction() "
  (is a)

---

② every Calc Operation 'has a' symbol

Q) why add a symbol per Calc Operation?

ans: in   Calculator.Kt   when rendering Result,
we can say   text = n1 + op.symbol + n2

so ans: cuz we need to render a unique symbol per Calc.Operation