Composite: Need (Component) ← BasicNeed (Leaf), Bundle
(Composite with List<ChildPart>).

MVC split:

Model: Need/BasicNeed/Bundle, Ledger/LogEntry, Catalog, Repository, value objects

Controller: coordinates reads/writes, computes summaries for a date

View: renders totals + pie/stack for fixed/variable/fees

Dependency directions: View → Controller → Model (not the other way around).

app

ui

«interface» View
+ render(DaySummary)

SwingView

implements

CliView

implements

AppController
+ addDonation(d, amt)
+ addFulfillment(d, name, count)
+ computeDaySummary(d)

calls

depends

Ledger
- entries: List
+ append(e:LogEntry)
+ fundOn(d:LocalDate): Money
+ goalOn(d:LocalDate): Money
+ fulfillmentsOn(d:LocalDate): List

holds *

infrastructure

depends

domain

Catalog
- byName: Map
+ add(n:Need)
+ get(name:String): Need
+ contains(name:String): boolean

—Map<String,Need>—

«interface» Need
+ name(): String
+ fixed(): Money
+ variable(): Money
+ fees(): Money
+ total(): Money

LogEntry
- date: LocalDate
- type: EntryType
- name: String (NEED)
- count: int (NEED)
- amount: Money (FUND/GOAL)

«interface» Repository
+ loadNeeds(path)
+ loadLog(path)
+ appendLog(path, entry)

implements

CsvRepository

reads   *reads/writes*

realizes   *realizes*

BasicNeed
- name: String
- fixed: Money
- variable: Money
- fees: Money

Bundle
- name: String
- parts: List
- catalog: Catalog

uses

Money
- amount: BigDecimal
+ of(d:double): Money
+ plus(m:Money): Money
+ times(k:int): Money

needs.csv   log.csv

1..* parts

BundlePart
- needName: String
- count: int (>0)

DaySummary
- date: LocalDate
- funds, goal, fixed, variable, fees: Money
+ totalNeedCost(): Money
+ surplus(): Money
+ gapToGoal(): Money