“Объект хандалтат холбооноос функциональ загварчлал руу”

Объект хандалтат загвар (OOP) ба холбоо

OOP-д системийг **класс ба объектуудын цуглуулга** гэж үздэг.

Класс хоорондын **холбоо (relationship)** нь системийн бүтэц, хамаарлыг илэрхийлнэ.

Холбооны төрөл	**Тайлбар**	**Жишээ**
Association	Хоёр объект мэдээлэл солилцдог ч тус тусдаа оршиж чадна.	Employee ↔ Department
Aggregation	Нэг объект нөгөөг “агуулах” боловч хамаарал сул, тусдаа оршиж чадна.	Department ◇→ Employee
Composition	Нэг объект нөгөөг бүрэн эзэмшдэг, амьдралын мөчлөг хамт.	PayrollRecord ◆→ Contribution

Эдгээр нь OOP-д **объектын харилцаа, насжилт, эзэмшлийн** утгыг харуулдаг.

Функциональ програмчлал (FP)-ын философи

FP-д:

- Төрөл (type) ба утга (value) гол үүрэгтэй.
- Системийг **өгөгдлийн төрөл (data types)** ба **цэвэр функцүүдийн (pure functions)** хослолоор илэрхийлнэ.
- Объектын “зан үйл”-ийг **функц** гүйцэтгэнэ, “мэдээлэл”-ийг **immutable record/ADT** хадгална.

Иймд OOP-ийн “холбоо”-г FP-д **өгөгдлийн бүтэц ба функцийг холбох замаар** илэрхийлдэг.

Холбоог FP-д хэрхэн буулгах вэ

OOP холбоо	**FP ойлголт**	**Жишээ**
Association	Төрөл хоорондын “reference” буюу ID холбоо, эсвэл lookup функц	claim.kind: BenefitType
Aggregation	Тусдаа төрөл, **ID линкээр холбох**; тус тусдаа lifecycle	CivilServant.orgId: OrganizationId
Composition	Нэг record дотор **nested record / list** байдлаар агуулах	PayrollRecord.contributions: Contribution[]

FP-д “агуулах”, “эзэмших” гэдэг санаа нь **record composition** эсвэл **ADT nesting** хэлбэрээр илэрдэг.

FP-ийн үндсэн зарчим — цэвэр функц ба инвариант

- **Цэвэр функц (Pure Function):** Нэг ижил оролтод үргэлж ижил гаралт өгнө, гадны төлөв өөрчлөхгүй.
- **Инвариант:** Төрөл болон байгуулагч функцээр дамжуулан өгөгдлийн дүрмийг баталгаажуулна.
  → Жишээ: amount > 0 байх ёстой бол Positive төрлөөр хамгаална.

Функциональ системийн гурван давхарга

1. **Domain Types (ADT):** өгөгдлийн төрөл ба холбоо
2. **Pure Functions:** бизнесийн дүрмийг хэрэгжүүлнэ
3. **Effects Layer:** IO, DB, сүлжээ гэх мэт гадаад орчинтой харьцах (цэвэр функц биш хэсэг)

FP загварын давуу тал

Давуу тал	**Тайлбар**
Тодорхой	Холбоо, инвариант нь төрлийн түвшинд баталгаажна.
Туршигдах боломжтой	Цэвэр функц тул тест бичихэд хялбар.
Эффект тусгаарлагдсан	Гадны төлөв (DB, IO) системийн цөмд нөлөөлөхгүй.
Өгөгдөл хувьсахгүй	Immutability → найдвартай concurrency, бага алдаа.

Жишээ харьцуулалт

OOP хэлбэр	**FP хэлбэр**
class PayrollRecord { List<Contribution> contributions; }	type PayrollRecord = { contributions: ReadonlyArray<Contribution> }
record.addContribution(c)	const addContribution = (r, c) => ({ ...r, contributions: [...r.contributions, c] })
Composition → объект эзэмшинэ	Composition → record дотор data nested байна

Энэ онолыг практикт хэрэглэх хүрээ

- **TypeScript (fp-ts)**, **F#**, **Scala**, эсвэл **Haskell** дээр функциональ загварчлал хийхэд ашиглаж болно.
- Домэйн төвтэй загварчлал (DDD) болон functional architecture хоёрыг хослуулж болно.

Дүгнэлт

OOP-д “класс хоорондын холбоо” гэдэг бол FP-д “төрлүүдийн өгөгдлийн бүтэц ба функцуудын зохиомж” юм.

Composition/Aggregation/Association гэх ойлголтууд устахгүй — харин **тэмдэглэл бус төрөл ба цэвэр функцийн түвшинд** хэрэгждэг.