final

Chore
- String choreName
- String description
- FunctionalSpaceType functionalSpaceType
- Duration estimatedTime
- Integer points
- ChoreStatus status

+ Chore(String choreName, String description, FunctionalSpaceType functionalSpaceType, Duration estimatedTime, Integer points, ChoreStatus status)
+ getChoreName(): String
+ setChoreName(String choreName): void
+ getChoreDescription(): String
+ setChoreDescription(String): void
+ getFunctionalSpaceType(): FunctionalSpaceType
+ setFunctionalSpaceType(FunctionalSpaceType functionalSpaceType): void
+ getEstimatedTime(): Duration
+ setEstimatedTime(Duration estimatedTime): void
+ getPoints(): Integer
+ setPoints(Integer points): void
+ getStatus(): ChoreStatus
+ setStatus(ChoreStatus status): void
+ getSpaceId(): Integer
+ setSpaceId(Integer): void
+ getAssignedUser(): User
+ assignUser(User user): void
+ markAsCompleted(): void
+ toString(): String

ISpace
Interface

+ getName(): String
+ setName(String name): void
+ getUsers(): List<User>
+ addUser(User user): void
+ removeUser(User): void

BaseEntity
ABS
Integer id

+ getId(): Integer
+ setId(Integer): void

User
- username: String
- email: String
- password: String
- points: Integer
- spaceId: Integer

+ User(String username, String email, String password, Integer points)
+ getName(): String
+ setName(String): void
+ getEmail(): String
+ setEmail(String): void
+ getPassword(): String
+ setPassword(): void
+ getPoints(): Integer
+ setPoints(Integer): void
+ getSpaceId(): Integer
+ setSpaceId(Integer): void
+ toString(): String

Space
- name: String
- users: List<User>

+ Space(String name)
+ getName(): String
+ setName(String name): void
+ getUsers(): List<User>
+ addUser(User user): void
+ removeUser(User): void
+ toString(): String

Note: If we want to let users being able to create their own functionalSpace, then we need to make it a concrete class instead of enum. Not our current concern.

FunctionalSpaceType
<enum>
KITCHEN,
BATHROOM,
LIVING_ROOM,
BEDROOM,
GARAGE,
HOME_OFFICE,
DINING_ROOM,
BALCONY

IUser
Interface

+ getName(): String
+ setName(String): void
+ getEmail(): String
+ setEmail(String): void
+ getPassword(): String
+ setPassword(String): void
+ getPoints(): Integer
+ setPoints(Integer): void
+ getSpaceId(): Integer
+ setSpaceId(Integer): void

IChore
Interface

+ getChoreName(): String
+ setChoreName(String choreName): void
+ getChoreDescription(): String
+ setChoreDescription(String): void
+ getFunctionalSpaceType(): FunctionalSpaceType
+ setFunctionalSpaceType(FunctionalSpaceType functionalSpaceType): void
+ getEstimatedTime(): Duration
+ setEstimatedTime(Duration estimatedTime): void
+ getPoints(): Integer
+ setPoints(Integer points): void
+ getStatus(): ChoreStatus
+ setStatus(ChoreStatus status): void
+ getSpaceId(): Integer
+ setSpaceId(Integer): void
+ getAssignedUser(): User
+ assignUser(User user): void
+ markAsCompleted(): void

ChoreStatus
<enum>
PENDING
IN_PROGRESS
COMPLETED
OVERDUE

TemplateServiceImpl
- templateMap: Map<FunctionalSpaceType, DefaultChoreTemplate>

+ TemplateServiceImpl(templates: List<DefaultChoreTemplate>)
+ createDefaultChores(Integer spaceId, FunctionalSpaceType type): void

Note: maintain a Map where the key is FunctionalSpaceType and the value is the corresponding DefaultChoreTemplate. This allows dynamic lookup for multiple selected types.

*Note: Business Logic:
• Anything involving workflows, multiple models, or database operations should move to a service class.*

TemplateService
Interface

+ createDefaultChores(spaceId: Integer, type: FunctionalSpaceType): void

FunctionalSpaceType
<enum>
KITCHEN,
BATHROOM,
LIVING_ROOM,
BEDROOM,
GARAGE,
HOME_OFFICE,
DINING_ROOM,
BALCONY

DefaultChoreTemplate
ABS

+ DefaultChoreTemplate(ChoreMapper choreMapper)
+ createDefaultChores(Integer spaceId): void
# getDefaultChores(Integer spaceId): List<Chore> {abstract}
+ getFunctionalSpaceType(): FunctionalSpaceType {abstract}

KitchenChoreTemplate
+ getDefaultChores(Long spaceId): List<Chore>
+ getFunctionalSpaceType(): FunctionalSpaceType

BathroomChoreTemplate
+ getDefaultChores(Long spaceId): List<Chore>
+ getFunctionalSpaceType(): FunctionalSpaceType

LivingRoomChoreTemplate
+ getDefaultChores(Long spaceId): List<Chore>
+ getFunctionalSpaceType(): FunctionalSpaceType

BedroomChoreTemplate
+ getDefaultChores(Long spaceId): List<Chore>
+ getFunctionalSpaceType(): FunctionalSpaceType

DiningRoomChoreTemplate
+ getDefaultChores(Long spaceId): List<Chore>
+ getFunctionalSpaceType(): FunctionalSpaceType

HomeOfficeChoreTemplate
+ getDefaultChores(Long spaceId): List<Chore>
+ getFunctionalSpaceType(): FunctionalSpaceType

GarageChoreTemplate
+ getDefaultChores(Long spaceId): List<Chore>
+ getFunctionalSpaceType(): FunctionalSpaceType

SpaceServiceImpl
- SpaceMapper spaceMapper
- UserMapper userMapper

+ getSpaceById(Integer id): Space
+ createSpace(Space space): void
+ updateSpace(Space space): void
+ addUserToSpace(Integer spaceId, Integer userId): void
+ removeUserFromSpace(Integer spaceId, Integer userId): void

*Note: • Space manages its internal state (e.g., adding users).
• SpaceService orchestrates higher-level workflows like creating a space or joining an existing space.*

SpaceService
+ getSpaceById(Integer id): Space
+ createSpace(Space space): void
+ updateSpace(Space space): void
+ addUserToSpace(Integer spaceId, Integer userId): void
+ removeUserFromSpace(Integer spaceId, Integer userId): void

UserServiceImpl
- UserMapper userMapper

+ getUserById(Integer id): User
+ createUser(User user): void
+ updateUser(User user): void
+ deleteUser(Integer id): void
+ login(User user): User
+ addPoints(Integer userId, Integer points): void

UserService
+ getUserById(Integer id): User
+ createUser(User user): void
+ updateUser(User user): void
+ deleteUser(Integer id): void
+ login(User user): User
+ addPoints(Integer userId, Integer points): void

ChoreServiceImpl
- ChoreMapper choreMapper
- UserMapper userMapper

+ getAllChores(Integer spaceId): List<Chore>
+ createChore(Chore chore): Chore
+ updateChore(Chore chore): Chore
+ deleteChore(Integer id): void
+ assignUserToChore(Integer choreId, Integer userId): void
+ completeChore(Integer choreId): Chore
+ getChoresByUserId(Integer userId): List<Chore>

ChoreService
+ getAllChores(Integer spaceId): List<Chore>
+ createChore(Chore chore): Chore
+ updateChore(Chore chore): Chore
+ deleteChore(Integer id): void
+ assignUserToChore(Integer choreId, Integer userId): void
+ completeChore(Integer choreId): Chore
+ getChoresByUserId(Integer userId): List<Chore>