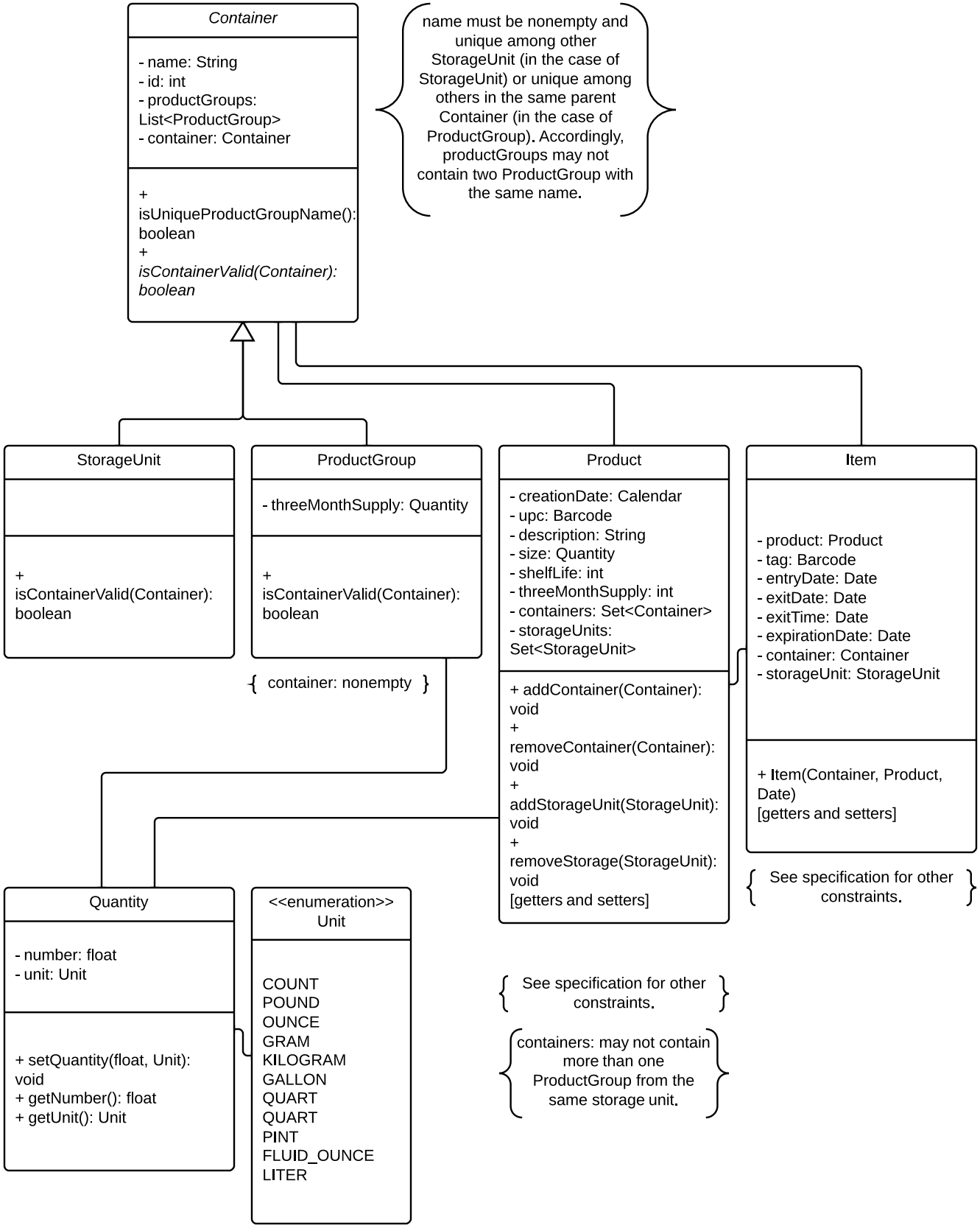
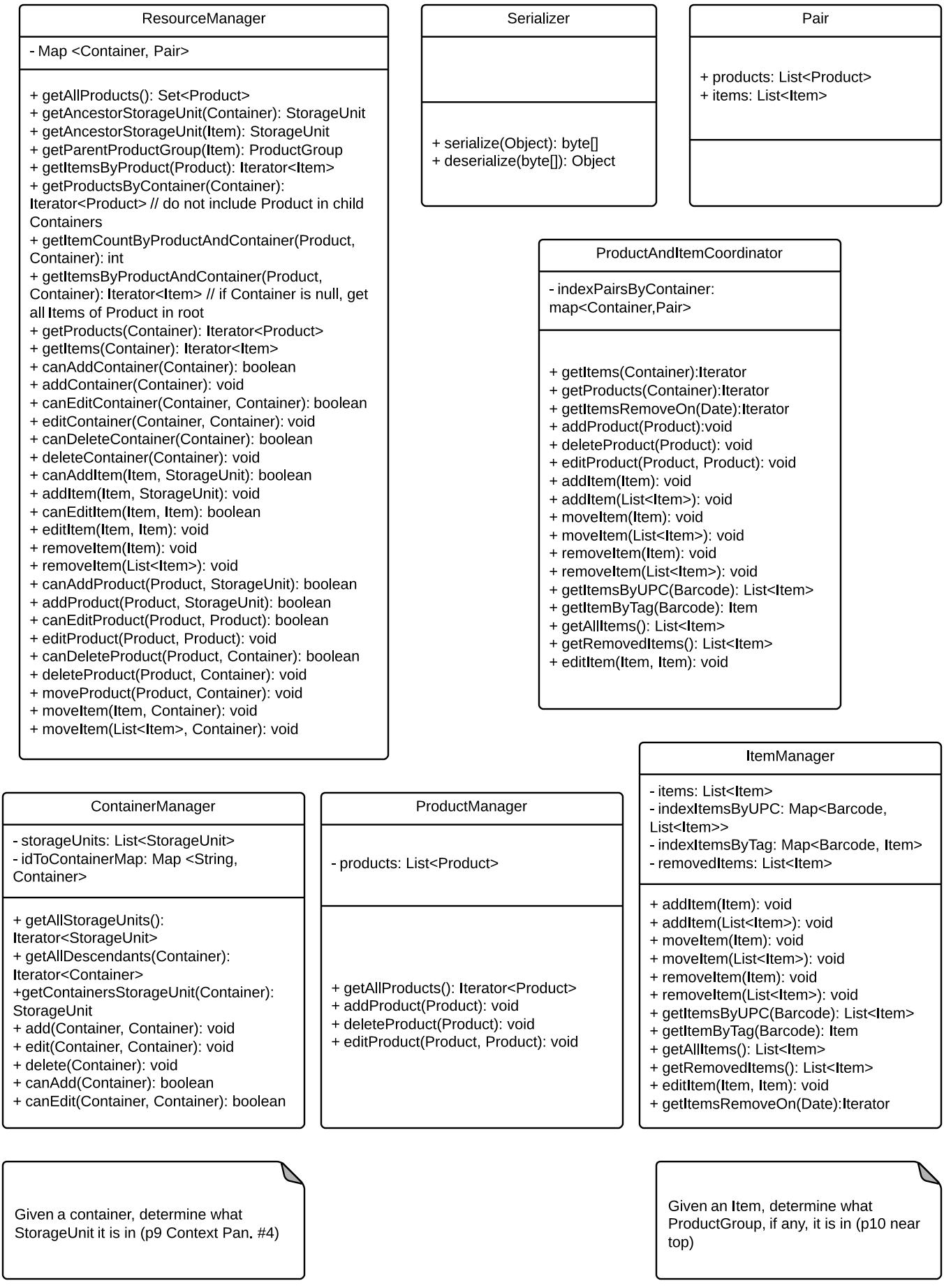

Container
- name: String
- id: int
- productGroups: List<ProductGroup>
- container: Container
+ isUniqueProductGroupName(): boolean
+ isContainerValid(Container): boolean
name must be nonempty and unique among other StorageUnit (in the case of StorageUnit) or unique among others in the same parent Container (in the case of ProductGroup). Accordingly, productGroups may not contain two ProductGroup with the same name.
StorageUnit
+ isContainerValid(Container): boolean
ProductGroup
- threeMonthSupply: Quantity
+ isContainerValid(Container): boolean
{ container: nonempty }
Product
- creationDate: Calendar
- upc: Barcode
- description: String
- size: Quantity
- shelfLife: int
- threeMonthSupply: int
- containers: Set<Container>
- storageUnits: Set<StorageUnit>
+ addContainer(Container): void
+ removeContainer(Container): void
+ addStorageUnit(StorageUnit): void
+ removeStorage(StorageUnit): void
[getters and setters]
{ See specification for other constraints. }
containers: may not contain more than one ProductGroup from the same storage unit.
Item
- product: Product
- tag: Barcode
- entryDate: Date
- exitDate: Date
- exitTime: Date
- expirationDate: Date
- container: Container
- storageUnit: StorageUnit
+ Item(Container, Product, Date)
[getters and setters]
{ See specification for other constraints. }
Quantity
- number: float
- unit: Unit
+ setQuantity(float, Unit): void
+ getNumber(): float
+ getUnit(): Unit
<<enumeration>> Unit
COUNT
POUND
OUNCE
GRAM
KILOGRAM
GALLON
QUART
QUART
PINT
FLUID_OUNCE
LITER


ResourceManager
- Map <Container, Pair>
+ getAllProducts(): Set<Product>
+ getAncestorStorageUnit(Container): StorageUnit
+ getAncestorStorageUnit(Item): StorageUnit
+ getParentProductGroup(Item): ProductGroup
+ getItemsByProduct(Product): Iterator<Item>
+ getProductsByContainer(Container): Iterator<Product> // do not include Product in child Containers
+ getItemCountByProductAndContainer(Product, Container): int
+ getItemsByProductAndContainer(Product, Container): Iterator<Item> // if Container is null, get all Items of Product in root
+ getProducts(Container): Iterator<Product>
+ getItems(Container): Iterator<Item>
+ canAddContainer(Container): boolean
+ addContainer(Container): void
+ canEditContainer(Container, Container): boolean
+ editContainer(Container, Container): void
+ canDeleteContainer(Container): boolean
+ deleteContainer(Container): void
+ canAddItem(Item, StorageUnit): boolean
+ addItem(Item, StorageUnit): void
+ canEditItem(Item, Item): boolean
+ editItem(Item, Item): void
+ removeItem(Item): void
+ removeItem(List<Item>): void
+ canAddProduct(Product, StorageUnit): boolean
+ addProduct(Product, StorageUnit): boolean
+ canEditProduct(Product, Product): boolean
+ editProduct(Product, Product): void
+ canDeleteProduct(Product, Container): boolean
+ deleteProduct(Product, Container): void
+ moveProduct(Product, Container): void
+ moveItem(Item, Container): void
+ moveItem(List<Item>, Container): void
Serializer
+ serialize(Object): byte[]
+ deserialize(byte[]): Object
Pair
+ products: List<Product>
+ items: List<Item>
ProductAndItemCoordinator
- indexPairsByContainer: map<Container,Pair>
+ getItems(Container):Iterator
+ getProducts(Container):Iterator
+ getItemsRemoveOn(Date):Iterator
+ addProduct(Product):void
+ deleteProduct(Product): void
+ editProduct(Product, Product): void
+ addItem(Item): void
+ addItem(List<Item>): void
+ moveItem(Item): void
+ moveItem(List<Item>): void
+ removeItem(Item): void
+ removeItem(List<Item>): void
+ getItemsByUPC(Barcode): List<Item>
+ getItemByTag(Barcode): Item
+ getAllItems(): List<Item>
+ getRemovedItems(): List<Item>
+ editItem(Item, Item): void
ItemManager
- items: List<Item>
- indexItemsByUPC: Map<Barcode, List<Item>>
- indexItemsByTag: Map<Barcode, Item>
- removedItems: List<Item>
+ addItem(Item): void
+ addItem(List<Item>): void
+ moveItem(Item): void
+ moveItem(List<Item>): void
+ removeItem(Item): void
+ removeItem(List<Item>): void
+ getItemsByUPC(Barcode): List<Item>
+ getItemByTag(Barcode): Item
+ getAllItems(): List<Item>
+ getRemovedItems(): List<Item>
+ editItem(Item, Item): void
+ getItemsRemoveOn(Date):Iterator
ContainerManager
- storageUnits: List<StorageUnit>
- idToContainerMap: Map <String, Container>
+ getAllStorageUnits(): Iterator<StorageUnit>
+ getAllDescendants(Container): Iterator<Container>
+getContainersStorageUnit(Container): StorageUnit
+ add(Container, Container): void
+ edit(Container, Container): void
+ delete(Container): void
+ canAdd(Container): boolean
+ canEdit(Container, Container): boolean
ProductManager
- products: List<Product>
+ getAllProducts(): Iterator<Product>
+ addProduct(Product): void
+ deleteProduct(Product): void
+ editProduct(Product, Product): void
Given a container, determine what StorageUnit it is in (p9 Context Pan. #4)
Given an Item, determine what ProductGroup, if any, it is in (p10 near top)