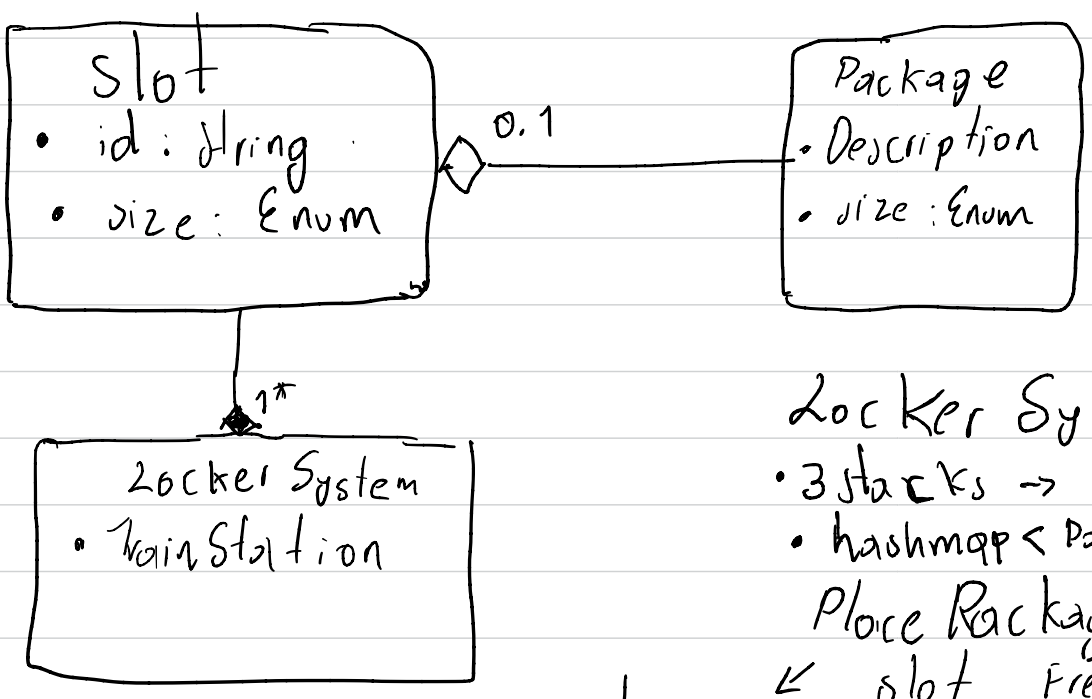
Design a Locker system

1. Who is going to use it? The peroon in a train station
2. how is going to use it? the peson receives a package, clasifies into (size) and put into a slot fitting the package.
3. You mentioned slot size, could you give me the sizes? - Sizes are S, m, 2.
4. How does apackage is placed? After measureed, it is assigned a size, the person looks for a spot of the size and places it.
5. What if the package does not fit? the person tries in a bigger spot. if there is no free spot, the package could not be taken.
6. How do you remove a package? the personal uses the description of the package.
7. - Does it have Pricing? - For this problem no.
8. Do we need to fill things in certain order? No, just try to use the slot with the correct size.



Locker System
- 3 stacks → For the free spots (S, M, L)
- hashmap<Pacake, slot> occupied

Place Package(Package p)
↙ slot FreeSlot = Find FreeSlot(p.size) O(1)
Consider case if (slot == null) throw Exception
same package occupied.put(p, FreeSlot);

```
FindFreeSlot(size, enum){
  FreeSlot = null
  (size -> stack)
  For (size in sizes)
    if (Psize > size) contiue
    if (!sizeMp.get(size).isEmpty) return
                ↳ .pop();
  return null
```

```
Class slot {
    enum size;
    String id;
}
```

```
class Package {
    String description
    Enum size;
}
```

```
class LockerSystem {

    List<Stack<slot>> FreeSlots;
    Hashmap<Package, slot> occupiedSlot;

    LockerSystem (List<Stack<slot>> FreeSlots) {
        this.FreeSlots = FreeSlots;        <- sort
        occupiedSlot = new Hashmap<>();
    }

    LockerSystem (List<size> sizes, List<integere> slotpersize) {
    }...

    slot PlacePackage (Package p) {
        if (occupiedSlots.containsKey(p)) throw new Error();
        slot Freespot: this.FindFreeslot(P.size); // later Defined
        if (Freespot == null ) throw new Error();
        occupiedSlot.put(P, slot);
    }

    Slot Find Free Slot (Size S) {
        For (Stac<slot> slots: FreeslotPool) {
            if (slots.isEmpty()) continue
            if (Slots.peek().size < S) continue
            return stots.pop();
        }
    return null
    }

    Slot Remove Package (Package P){
        if(!occupied.containskey(P)) throw new Error();
        slot S = occupied.get(P);
        Free slot (S);
        occupied.remove(P);

    }
}
```