Flip-Pod

Flip-Pod is a new delivery initiative by Flipkart where orders will be delivered to pods from where customers can pick according to their convenience.
The **pod** is a collection of different size **lockers** that store orders. Customers can come to the pods and collect all the orders present for them.

Implementation (Basic)

1. Design a Flip-Pod system where you can specify the pods and the number of locks with the corresponding size in that pod
2. A customer can come on Flipkart's website and place an order specifying the size
3. The customer can go to the pods and collect all his available orders
4. A pod should be uniquely identified and can have multiple lockers
5. A locker can have orders from only 1 customer
6. A locker can be of size Small or Large
7. An order can also be of size Small or Large
8. Relation in sizes: 1 Large = 2 Small. Hence 1 Large locker can contain 1 Large Order or 2 Small orders. For allocation details refer to 9 e)
9. Criteria for allocating a locker to an order:
   a. Allocation can be done once an order is placed if an empty locker is available
      i. A locker can become available if a new locker gets added
      ii. A locker can become available if an existing locker gets freed when a customer collects his order
      iii. If no locker is available, order must be placed in a queue
   b. Priority is given to an order having the same size as the locker.
   For eg: Consider an order of small size has been placed and 2 lockers, one of small size and one of large size are available. In such a scenario, choose the locker of small size to allocate the order.
   c. In case multiple orders are available, priority will be given to the oldest order in the queue pending for allocation.
   d. An order once allocated cannot be reallocated to another locker
   e. If you have 1 locker of size Large and 2 orders from the same customer of size small, then both can be allocated to the same locker. However 1 large locker cannot contain 1 small order of customer1 and 1 small order of customer2.

User Triggered Actions

1. A Pod can be added to the system
2. A locker can be created by providing a size and it can be added to an existing pod
3. A customer can be created
4. A customer can place an order with a specific order size
5. A customer can collect all his available orders from the pods

6. Print the status of the pod/queue after each allocation/removal of an order

Assumptions

1. There is no time lag between the customer placing an order and the order reaching the pod
2. 1 order contains exactly 1 Item

Bonus

1. Prioritise order allocation for Flipkart Plus Customers

Guidelines

1. Do not use any database, no-sql store. **Only use an in-memory data structure**
2. Do not create any UI for the application
3. **Write a driver class for demo purposes**. This will execute all the commands at one place in the code and have test cases.
4. Prioritise code compilation, execution and completion
5. Work on the expected output first and then add good-to-have features of your own

Sample Tests

Sl No.	Action	Pod	Locker State	Queue State	Expected
1	add_customer() -> c1	null	null	null	SUCCESS
2	add_customer() -> c2	null	null	null	SUCCESS
3	add_customer() -> c3	null	null	null	SUCCESS
4	add_pod() -> p1	p1	null	null	SUCCESS
5	recieve_order(c1)	p1	null	null	EMPTY
6	add_locker(p1, small) -> ls1	p1	[ls1{}]	null	SUCCESS
7	add_locker(p1, large) -> ll2	p1	[ls1{},ll2{}]	null	SUCCESS
8	place_order(c1,small) -> c1-s1	p1	[ls1{c1-s1},ll2{}]	null	Locker ls1 has been assigned to order c1-s1
9	place_order(c1,small) -> c1-s2	p1	[ls1{c1-s1}, ll2{c1-s2}]	null	Locker ll2 has been assigned to

					order c1-s2
10	place_order(c1,small) -> c1-s3	p1	[ls1{c1-s1}, ll2{c1-s2,c1-s3}]	null	Locker ll2 has been assigned to order c1-s3
11	add_locker(p1, large) -> ll3	p1	[ls1{c1-s1}, ll2{c1-s2,c1-s3}, ll3]	null	SUCCESS
12	place_order(c2,small) -> c2-s1	p1	[ls1{c1-s1}, ll2{c1-s2,c1-s3}, ll3{c2-s1}]	null	locker ll3 has been assigned to order c2-s1
13	place_order(c3,small) -> c3-s1	p1	[ls1{c1-s1}, ll2{c1-s2,c1-s3}, ll3{c2-s1}]	[c3-s1]	Order c3-s1 cannot be grouped and has been added to the queue
14	place_order(c2,small) -> c2-s2	p1	[ls1{c1-s1}, ll2{c1-s2,c1-s3}, ll3{c2-s1,c2-s2}]	[c3-s1]	locker ll3 has been assigned to order c2-s2
15	place_order(c3,small) -> c3-s2	p1	[ls1{c1-s1}, ll2{c1-s2,c1-s3}, ll3{c2-s1,c2-s2}]	[c3-s1, c3-s2]	Order c3-s2 cannot be grouped and has been added to the queue
16	receive_order(c2)	p1	[ls1{c1-s1}, ll2{c1-s2,c1-s3}, ll3{c3-s1,c3-s2}]	null	Orders collected are: c2-s1, c2-s2 Locker ll3 has been assigned to orders c3-s1,c3-s2