برای پیاده سازی FIFO ابتدا یک کلاس برای آن تعریف کردیم. این کلاس آرایه ای از صندلی ها دارد (int[] Seats) و emptyIDX نشان دهنده صندلی خالی بعدی است که باید پر شود.

تابع isIn، شماره صندلی مشتری با شماره x را بر میگرداند و اگر x در رستوران نبود 1- را برمیگرداند.

تابع add، مشتری با شماره x را به رستوران اضافه میکند. چگونه ؟ ابتدا تابع isIn(x) را صدا میزند، و اگر مشتری در رستوران بود، از تابع خارج میشود. (این همان حالتی است که مشتری از قبل در رستوران بوده و دوباره سفارش داده است.) در غیر این صورت مشتری x باید به رستوران اضافه شود. پس pageFault داشته ایم. حال یا صندلی خالی در رستوران داریم، که در این صورت آن صندلی را به x اختصاص میدهیم و یا تمام صندلی ها پر هستند و باید اولین نفری که وارد رستوران شده را بیرون کرده و صندلی اش را به x بدهیم.

دقت کنید که صندلی ها را به ترتیب از 0 تا n به مشتری ها اختصاص میدادیم. پس صندلی 0 قدیمی ترین مشتری را دارد. پس کافیست آرایه را یک واحد به چپ شیفت بدهیم و مشتری x را در صندلی n بنشانیم.

برای پیاده سازی LRU نیز کلاسی تعریف میکنیم که از FIFO ارث بری میکند. آرایه frequency را به آن اضافه میکنیم که نشان دهنده تعداد سفارشات مشتری متناظرش است.

تابع add را کمی تغییر میدهیم بدین صورت که اگر مشتری از قبل در رستوران بود تعداد سفارشات آنرا بروز رسانی میکنیم. و اگر نبود، مشتری ای که کمترین تعداد سفارش را داشته را بیرون میکنیم و مانند FIFO شیفت به چپ میدهیم تا بین مشتری هایی که تعداد سفارش برابر داشته اند، قدیمی ترینشان اول قرار بگیرد.

Second Chance نیز کاملا مشابه LRU است با این فرق که برای پیدا کردن Victim از pointer شروع میکند نه از اول. همچنین در مسیرش اگر تعداد سفارش برابر یک دید آنرا صفر میکند. همچنین اگر مشتری از قبل داخل رستوران بود تعداد سفارش آن برابر یک میشود (تعداد سفارشات در واقع نشاند دهنده این است که مشتری x دارای شانس دوم هست یا نه، اگر تعداد سفارشات برابر یک بود یعنی بله و اگر صفر بود یعنی خیر.)

در Second Chance به چپ شیفت نمیدهیم چون مشتری جدید صندلی victim را اختیار میکند و pointer بروز رسانی میشود.

برای دریافت پیغام از server ترد t1 را میسازیم که نقش producer را بازی میکند و برای اجرای الگوریتم ها از ترد t2 استفاده میکنیم که در واقع مثل consumer در مسئله Producer & Consumer عمل میکند.

ترد t1 در بدنه اصلی خود از سرور شماره مشتری را دریافت میکند و از طریق شیئی از کلاس PC تابع produce را صدا میزند. در کلاس PC یک لیست با ماکزیمم طول 5 داریم که هر گاه produce صدا زده شود یک عدد به این لیست اضافه میشود مگر اینکه لیست پر بوده باشد، در این صورت produce یا همان ترد t1 صبر میکند تا ترد t2 حداقل یکی از اعضای این لیست را مصرف کند. در صورتیکه از سرور عدد 0 دریافت شود، ترد t2 توسط t1، interrupt میشود و t1 نیز break میکند.

ترد t2 ابتدا یکبار منتظر میماند تا اولین پیغام (که نشان دهنده تعداد صندلی های رستوران است)از طریق سرور دریافت شود و از طریق t1 تولید شود. سپس آنرا مصرف میکند و وارد حلقه بینهایتی میشود که در آن به مصرف کردن ادامه میدهد و هر عددی که مصرف میکند را به الگوریتم ها از طریق تابع add، اضافه میکند و تابع add مابقی کارها را انجام میدهد. برای برقرار کانکشن نیز از یک سوکت با پورت 8080 استفاده کردیم و آنرا به ترد t1 پاس دادیم و t1 از طریق DataInputStream از این سوکت پیغام دریافت میکند.

FIFO

```
Customer ID: 5
Seats: 5 # #
PAGE FAULT: 1

Customer ID: 3
Seats: 5 3 #
PAGE FAULT: 2

Customer ID: 12
Seats: 5 3 12
PAGE FAULT: 3

Customer ID: 9
Seats: 3 12 9
PAGE FAULT: 4

Customer ID: 3
Seats: 3 12 9
PAGE FAULT: 4

Customer ID: 3
Seats: 3 12 9
PAGE FAULT: 4

Customer ID: 9
Seats: 3 12 9
PAGE FAULT: 4

Customer ID: 7
Seats: 12 9 7
PAGE FAULT: 5

Customer ID: 5
Seats: 9 7 5
PAGE FAULT: 6
```

LRU

```
Customer ID: 5
Seats: 5 # #
Freqs: 0  0  0
PAGE FAULT: 1

Customer ID: 3
Seats: 5 3 #
Freqs: 0  0  0
PAGE FAULT: 2

Customer ID: 12
Seats: 5 3 12
Freqs: 0  0  0
PAGE FAULT: 3

Customer ID: 9
Seats: 3 12 9
Freqs: 0  0  0
PAGE FAULT: 4

Customer ID: 3
Seats: 3 12 9
Freqs: 1  0  0
PAGE FAULT: 4

Customer ID: 3
Seats: 3 12 9
Freqs: 2  0  0
PAGE FAULT: 4

Customer ID: 9
Seats: 3 12 9
Freqs: 2  0  1
PAGE FAULT: 4
```

Second-Chance

```
Customer ID: 5
Seats:  5 # #
Chance: N  N  N
PAGE FAULT: 1

Customer ID: 3
Seats:  5 3 #
Chance: N  N  N
PAGE FAULT: 2

Customer ID: 12
Seats:  5 3 12
Chance: N  N  N
PAGE FAULT: 3

Customer ID: 9
Seats:  9 3 12
Chance: N  N  N
PAGE FAULT: 4

Customer ID: 3
Seats:  9 3 12
Chance: N  Y  N
PAGE FAULT: 4

Customer ID: 3
Seats:  9 3 12
Chance: N  Y  N
PAGE FAULT: 4

Customer ID: 9
Seats:  9 3 12
Chance: Y  Y  N
PAGE FAULT: 4
```