به نام خدا

محمد سعید زارع – ۹۹۲۴۳۰۳۷

گزارش پروژه درس سیستم عامل

---

۷ کلاس با نام های زیر نوشته شده است که به شرح هر کدام میپردازیم.

۱. کلاس client: این کلاس ابتدا بر روی localhost و پورت ۸۰۸۰ به server نوشته شده با کانکشن سوکت وصل میشود. سپس استریم ورودی برای دریافت اطلاعات از سمت سرور تعریف میکنیم و نام آن را din میگذاریم. همانطور که میدانیم سرور ابتدا مقدار سایز مموری را به طرف client ارسال میکند و سپس در این قسمت کد آن را میگیریم و پرینت میکنیم.
در این قسمت یک نمونه بافر ساختیم که این کلاس از دیزاین پترن سینگلتون استفاده میکند که یک ساختمان داده برای هماهنگی بین دو ترد دریافت کننده اطلاعات و ترد محاسبه کننده است که در قسمت های بعد با ساختار آن آشنا میشویم.
یک ترد جدید ایجاد میکنیم که کار آن محاسبه تعداد page fault هاست و عملا کلاس PageFault را در ورودی میگیرد. این بخش نیز بعدا توضیح داده خواهد شد. سپس این ترد را شروع میکنیم.
حالا یک حلقه بی نهایت میزنیم که در هر iteration یک عدد از سرور بگیرد و اگر این عدد مخالف ۰ بود آن را در بافر بنویسد ولی اگر این مقدار دقیقا ۰ بود -۱ را در بافر مینویسیم که نشاندهنده تمام شدن کار writer است.
و در آخر پس این حلقه thread.join() را صدا میزنیم که پراسس main زود تر از پراسس دوم تمام نشود و عملا منتظر بماند و هر دو با هم پایان یابند.

۲. کلاس PageFault : این کلاس یک ترد جدا است که بافر را در این کلاس نیز صدا میزنیم و به این صورت عمل میکند که هر سه نمونه از calculator ها busy wait انجام میدهند تا وقتی که داده جدیدی وارد بافر شود که این نمونه ها آن را نخوانده باشند سپس آن مورد را از بافر میخوانند و برایش متد requestToMemory را صدا میزنند و این متد ها عملا بررسی میکنند که page fault رخ میدهد یا خیر و تعداد page fault ها و صف های داخل خودشان را آپدیت میکنند. و در اخر هنگامی که ما از هر سه head مقدار ۱- خواندیم به این معنا است که هر سه روش به انتهای خودشان رسیده اند و در نتیجه متد isFinished فعال میشود و از حالت busy wait خارج میشویم. (در این قسمت میتوانستیم از wait و notify استفاده کنیم ولی ترجیح داده شد از busy wait استفاده شود.) و در اخر تعداد page fault های هر روش را خروجی میدهیم.

۳. کلاس replacement: این کلاس یک ساختمان داده برای نگه داری جای نشستن مشتری هاست و عملا یک حافظه جداگانه از محاسبات در نظر گرفته شده تا هنگامی که فرد جدیدی میاید

همه مشتری ها شیفت نخورند در واقع ما با روش های گفته شده فرد هدف را انتخاب میکنیم و افراد را در صف شیفت میدهیم ولی در واقعیت این شیفت نیاز نیست.
این کلاس دو متد اصلی دارد که یکی sitDown است و زمانی استفاده میشود که هنوز در رستوران جای خالی وجود دارد و دومی replace است که یکی را که با توجه به الگوریتم ها انتخاب کردیم را بیرون میندازد و فرد جدیدی را جایگزین میکند.
متد print برای چاپ کردن میزهای رستوران و متد findIndex برای پیدا کردن محل قرار گیری فرد address است.

٤. کلاس SharedMemoryBuffer : این کلاس از نوع سینگلتون است و عملا دارای یک آرایه است که در آن هر یک از الگوریتم ها هد های جداگانه ای دارند که روی آرایه حرکت میکنند (مثل مساله readers – writers که ۳ تا reader و یکی writer داریم.) این کلاس یک کلاس مشترک بین دو ترد است پس باید فانکشن هایش ترد سیف باشند پس متد ها را synchronized تعریف کردیم. این کلاس دارای ۵ متد اصلی است. متد write توسط ترد writer انجام میشود و به این صورت عمل میکند که صرفا در داخل بافر مینویسد. سه متد برای هر کدام از الگوریتم ها داریم که هر کدام یکی از هد خودشان را میخوانند و آن هد را یکی زیاد میکنند. و متد آخر isFinished است که زمانی فعال میشود که هر سه هد مقدار ۱- بخوانند.

حالا سراغ سه کلاس اصلی برای این سوال میرویم.

هر سه الگوریتم متد RequestToMemory را دارند که پیاده سازی هر کدام را شرح میدهیم.

- **کلاس Fifo**: همانطور که میدانیم این الگوریتم هر فردی که زود تر وارد شده باشد را وقتی رستوران پر شد زود تر خارج میکند پس برای همین یک صف در نظر گرفتیم که تا وقتی که میزها پر نشده فرد به ته صف اضافه شود و وقتی که میزها پر شد فردی که زود تر از همه امده فارغ از تعداد تکرار در مراحل بعد باید از رستوران خارج شود. در این مرحله فردی که باید از سیستم خارج شود را با فردی که وارد شود جاهایشان را عوض میکنند.
- **کلاس Lru**: این الگوریتم به این صورت عمل میکند که فردی که تازه دیده شده اولویت کمتری برای خروج دارد پس یک صف در نظر میگیریم که افراد را وارد این صف میکنیم و وقتی یک عددی خواندیم که سر میزی نشسته یا به عبارتی hit میخورد پس چون آن را باز تازه مشاهده کردیم به ته صف میبریم و اگر جدید آمده بود miss میخورد ولی باز به ته صف میرود و مثل قبل هنگامی که جا باید عوض شود متد replace را صدا میزنیم.
- **کلاس SecondChance**: الگوریتم آخر الگوریتم secondChance است که این یک بیت reference دارد که به این صورت است که هنگامی که یک عدد که در رستوران هست باز دیده شود این بیتش ۱ میشود و بقیه الگوریتم مثل صف است و هنگامی که رستوران پر میشود

و یک فرد جدید باید بیاید عنصر سر صف اگر این بیتش ۰ بود خارج میشود ولی اگر ۱ بود شانس مجدد به آن داده میشود و این بیت ۰ میشود و در انتهای صف قرار میگیرد. دقیقا کد این قسمت مشابه با این توضیحات آمده است که یک آرایه seen تعریف کردیم که اندیس هایش آدرس ها و داخلش یا صفر و یا یک مینویسیم. و متد replace و sitdown را دقیقا مشابه قبل صدا میزنیم. (برای قسمت حذف از سر صف یک حلقه زدیم تا وقتی که بیت refrence عنصر head یک هست اون عنصر را از سر صف حذف کند و به ته صف اضافه کند و بیتش را صفر کند.)

تست صحت راه حل:

تست ۱:

اعداد زیر وارد شده: (size = ۲)

۶ ۶ ۸ ۱ ۷ ۵ ۵ ۱۰ ۸ ۸ ۷ ۸ ۲ ۷ ۲ ۸ ۸ ۸ ۹ ۱۰ ۲ ۵ ۴ ۷ ۷ ۶ ۷ ۶ ۳ ۵ ۴ ۹ ۱۰ ۳ ۱ ۶ ۲ ۱ ۱ ۰ ۱ ۵ ۳ ۹

۳ ۲ ۱ ۲ ۷ ۲ ۵ ۶ ۱ ۷ ۱۰ ۵ ۷ ۳ ۱۰ ۷ ۶ ۲ ۴ ۸ ۸ ۴ ۵ ۱۰ ۵ ۱۰ ۱۰ ۶ ۱ ۶ ۱۰ ۷ ۴ ۱

به ازای این اعداد خروجی برنامه ما به این صورت است.

```
LRU: ..(1) 3..(2) 2
Second Chance: ..(1) 3..(2) 2
person -1 enter!
LRU: 58 FIFO: 58 Second-chance: 59
```

که حالا با استفاده از سایت ها آنلاین این مقدار را بررسی میکنیم.

نتیجه FIFO:

- Total frames: 2
- Algorithm: FIFO
- Reference string length: 77 references
- String: 9 3 5 1 10 1 2 6 1 3 10 9 4 5 3 6 7 6 7 7 4 5 2 10 9 8 8 8 2 7 2 8 7 8 8 10 5 5 7 1 8 6 6 1 4 7 10 6 1 6 10 10 5 10 5 4 8 8 4 2 6 7 10 3 7 5 10 7 1 6 5 2 7 2 1 2 3

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
ref		9	3	5	1	10	1	2	6	1	3	10	9	4	5	3	6	7	6	7	7	4	5	2	10	9	8	8	8	2
f		9	3	5	1	10	10	2	6	1	3	10	9	4	5	3	6	7	7	7	7	4	5	2	10	9	8	8	8	2
f			9	3	5	1	1	10	2	6	1	3	10	9	4	5	3	6	6	6	6	7	4	5	2	10	9	9	9	8
hit		X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	X	X	✓	✓	✓	X	X	X	X	X	X	✓	✓	X
v				9	3	5		1	10	2	6	1	3	10	9	4	5	3				6	7	4	5	2	10			9

t	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
ref	7	2	8	7	8	8	10	5	5	7	1	8	6	6	1	4	7	10	6	1	6	10	10	5	10	5	4	8	8	4
f	7	7	8	8	8	8	10	5	5	7	1	8	6	6	1	4	7	10	6	1	1	10	10	5	5	5	4	8	8	8
f	2	2	7	7	7	7	8	10	10	5	7	1	8	8	6	1	4	7	10	6	6	1	1	10	10	10	5	4	4	4
hit	X	✓	X	✓	✓	✓	X	X	✓	X	X	X	X	✓	X	X	X	X	X	X	✓	X	✓	X	✓	✓	X	X	✓	✓
v	8		2				7	8		10	5	7	1		8	6	1	4	7	10		6		1			10	5		

t	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77												
ref	2	6	7	10	3	7	5	10	7	1	6	5	2	7	2	1	2	3												
f	2	6	7	10	3	7	5	10	7	1	6	5	2	7	7	1	2	3	-	-	-	-	-	-	-	-	-	-	-	-
f	8	2	6	7	10	3	7	5	10	7	1	6	5	2	2	7	1	2	-	-	-	-	-	-	-	-	-	-	-	-
hit	X	X	X	X	X	X	X	X	X	X	X	X	X	X	✓	X	X	X												
v	4	8	2	6	7	10	3	7	5	10	7	1	6	5		2	7	1												

- Total references: 77
- Total distinct references: 10
- Hits: 19
- Faults: 58
- **Hit rate:** 19/77 = **24.675324675325%**
- **Fault rate:** 58/77 = **75.324675324675%**

نتیجه LRU:

- Total frames: 2
- Algorithm: LRU
- Reference string length: 77 references
- String: 9 3 5 1 10 1 2 6 1 3 10 9 4 5 3 6 7 6 7 7 4 5 2 10 9 8 8 8 2 7 2 8 7 8 8 10 5 5 7 1 8 6 6 1 4 7 10 6 1 6 10 10 5 10 5 4 8 8 4 2 6 7 10 3 7 5 10 7 1 6 5 2 7 2 1 2 3

Solution visualization

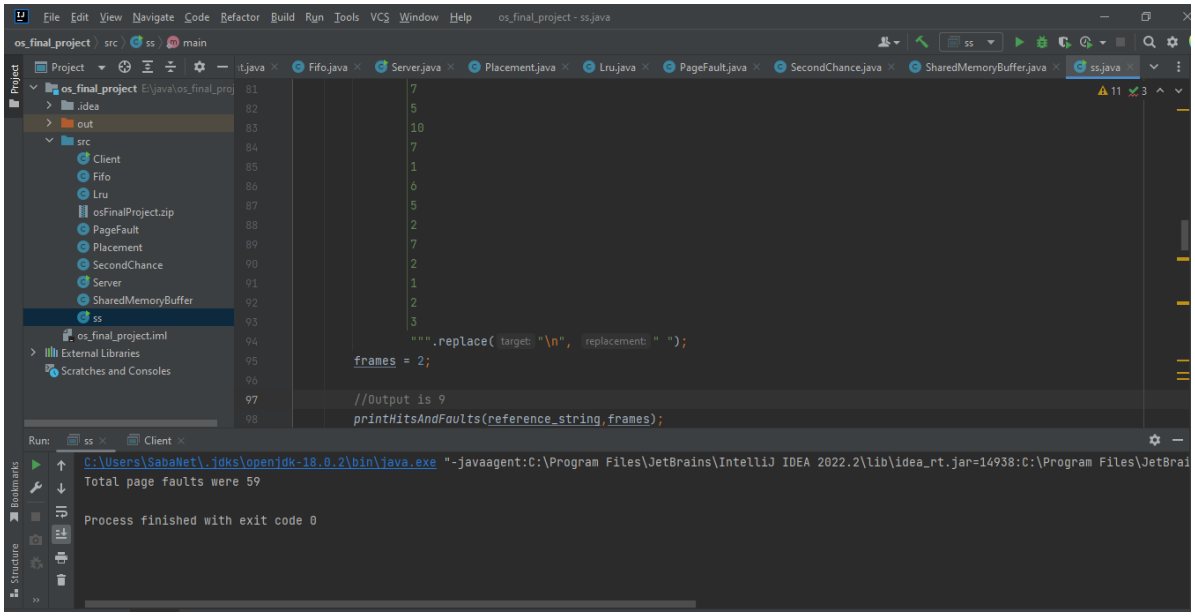
t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
ref		9	3	5	1	10	1	2	6	1	3	10	9	4	5	3	6	7	6	7	7	4	5	2	10	9	8	8	8	2
f		9	3	5	1	10	1	2	6	1	3	10	9	4	5	3	6	7	6	7	7	4	5	2	10	9	8	8	8	2
f			9	3	5	1	10	1	2	6	1	3	10	9	4	5	3	6	7	6	6	7	4	5	2	10	9	9	9	8
hit		X	X	X	X	X	✓	X	X	X	X	X	X	X	X	X	X	X	✓	✓	✓	X	X	X	X	X	X	✓	✓	X
v				9	3	5		10	1	2	6	1	3	10	9	4	5	3				6	7	4	5	2	10			9

t	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57	58	59
ref	7	2	8	7	8	8	10	5	5	7	1	8	6	6	1	4	7	10	6	1	6	10	10	5	10	5	4	8	8	4
f	7	2	8	7	8	8	10	5	5	7	1	8	6	6	1	4	7	10	6	1	6	10	10	5	10	5	4	8	8	4
f	2	7	2	8	7	7	8	10	10	5	7	1	8	8	6	1	4	7	10	6	1	6	6	10	5	10	5	4	4	8
hit	X	✓	X	X	✓	✓	X	X	✓	X	X	X	X	✓	X	X	X	X	X	X	✓	X	✓	X	✓	✓	X	X	✓	✓
v	8		7	2			7	8		10	5	7	1		8	6	1	4	7	10		1		6			10	5		

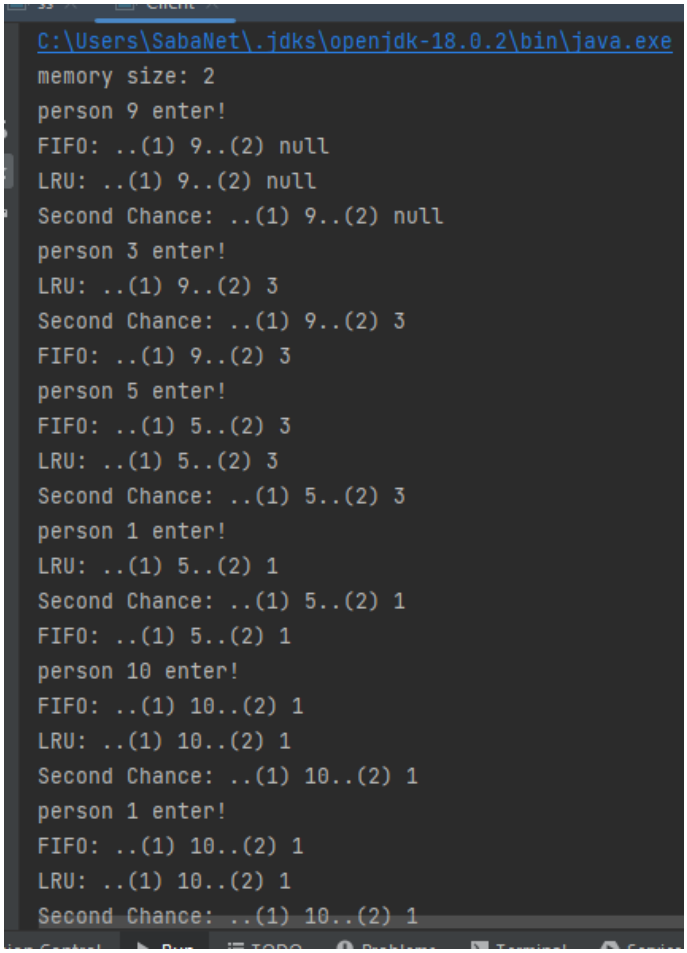
t	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77												
ref	2	6	7	10	3	7	5	10	7	1	6	5	2	7	2	1	2	3												
f	2	6	7	10	3	7	5	10	7	1	6	5	2	7	2	1	2	3	-	-	-	-	-	-	-	-	-	-	-	-
f	4	2	6	7	10	3	7	5	10	7	1	6	5	2	7	2	1	2	-	-	-	-	-	-	-	-	-	-	-	-
hit	X	X	X	X	X	X	X	X	X	X	X	X	X	X	✓	X	✓	X												
v	8	4	2	6	7	10	3	7	5	10	7	1	6	5		7		1												

- Total references: 77
- Total distinct references: 10
- Hits: 19
- Faults: 58
- **Hit rate:** 19/77 = **24.675324675325%**
- **Fault rate:** 58/77 = **75.324675324675%**

و چون برای الگوریتم SecondChance سایتی وجود نداشت و تعداد ورودی ها زیاد شد پس از کد داخل سایت gfg استفاده میکنیم.



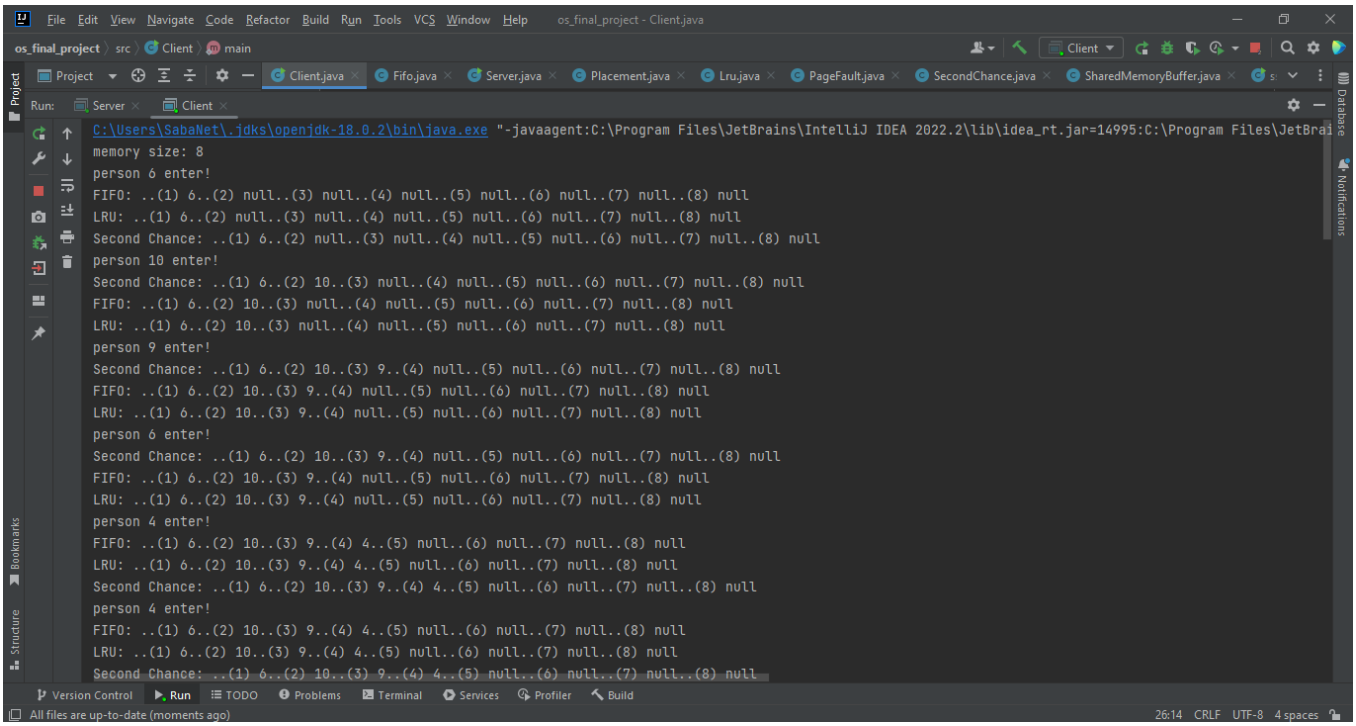
که دقیقا هر سه خروجی مانند خروجی ها کد ما است.



خروجی برنامه ما به این صورت است در خط اول تعداد میزها چاپ میشود که اینجا دو است و سپس شماره میز ها داخل پرانتز نمایش داده میشود و سپس بعد از آن شماره فردی که روی آن صندلی نشسته داده میشود مثلا اینجا اول رستوران خالی است سپس فرد شماره ۹ وارد میشود و روی اولین میز مینشیند (در هر سه الگوریتم یکسان است چون جای خالی داریم) سپس ۳ میاید و در جای خالی بعد مینشیند سپس ۵ میاید و طبق هر سه الگوریتم باید فرد ۹ که هم زود تر آمده و هم least recently used و هم بیت refrence آن صفر است باید از صف خارج شود پس شکل نوشتاری ما به این صورت است و به همین ترتیب تا آخر ادامه پیدا میکند.

تست دوم:

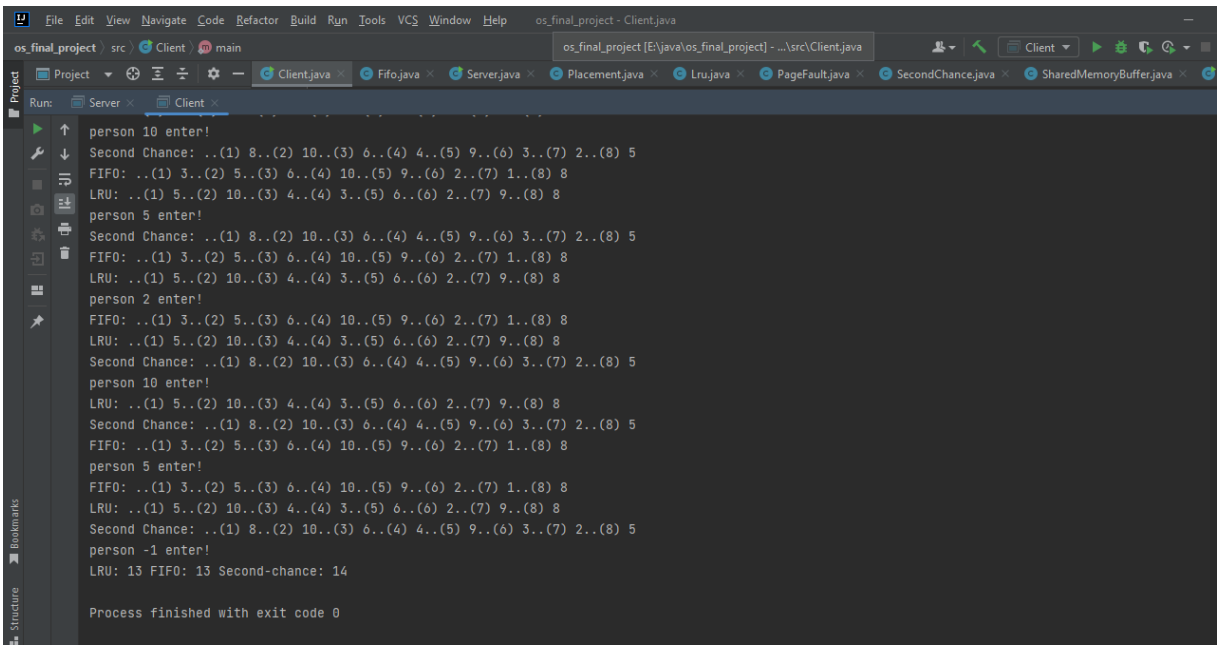
خروجی کلی:



اعداد وارد شده:

۵ ۱۰ ۲ ۵ ۱۰ ۸ ۸ ۹ ۱۰ ۴ ۳ ۶ ۶ ۸ ۵ ۲ ۱۰ ۲ ۴ ۲ ۴ ۲ ۱۰ ۳ ۸ ۱ ۷ ۱۰ ۶ ۱۰ ۲ ۹ ۹ ۹ ۷ ۴ ۴ ۶ ۹ ۱ ۰ ۶

نتیجه:



خروجی LRU سایت:

Summary - LRU algorithm

- Total frames: 8
- Algorithm: LRU
- Reference string length: 41 references
- String: 6 10 9 6 4 4 7 9 9 9 2 10 6 10 7 1 8 3 10 2 4 2 4 2 10 2 5 8 6 6 3 4 10 9 8 8 10 5 2 10 5

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
ref		**6**	**10**	**9**	**6**	**4**	**4**	**7**	**9**	**9**	**9**	**2**	**10**	**6**	**10**	**7**	**1**	**8**	**3**	**10**	**2**	**4**	**2**	**4**	**2**	**10**	**2**	**5**	**8**	**6**
f		6	10	9	6	4	4	7	9	9	9	2	10	6	10	7	1	8	3	10	2	4	2	4	2	10	2	5	8	6
f			6	10	9	6	6	4	7	7	7	9	2	10	6	10	7	1	8	3	10	2	4	2	4	2	10	2	5	8
f				6	10	9	9	6	4	4	4	7	9	2	2	6	10	7	1	8	3	10	10	10	10	4	4	10	2	5
f						10	10	9	6	6	6	4	7	9	9	2	6	10	7	1	8	3	3	3	3	3	3	4	10	2
f								10	10	10	10	6	4	7	7	9	2	6	10	7	1	8	8	8	8	8	8	3	4	10
f												10	6	4	4	4	9	2	6	6	7	1	1	1	1	1	1	8	3	4
f																	4	9	2	2	6	7	7	7	7	7	7	1	1	3
f																		4	9	9	9	6	6	6	6	6	6	7	7	1
hit		✗	✗	✗	✓	✗	✓	✗	✓	✓	✓	✗	✓	✓	✓	✓	✗	✗	✗	✓	✓	✗	✓	✓	✓	✓	✓	✗	✓	✗
v																			4			9						6		7

t	30	31	32	33	34	35	36	37	38	39	40	41																		
ref	**6**	**3**	**4**	**10**	**9**	**8**	**8**	**10**	**5**	**2**	**10**	**5**																		
f	6	3	4	10	9	8	8	10	5	2	10	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	8	6	3	4	10	9	9	8	10	5	2	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	5	8	6	3	4	10	10	9	8	10	5	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	2	5	8	6	3	4	4	4	9	8	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	10	2	5	8	6	3	3	3	4	9	9	9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	4	10	2	5	8	6	6	6	3	4	4	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	3	4	10	2	5	5	5	5	6	3	3	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	1	1	1	1	2	2	2	2	2	6	6	6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hit	✓	✓	✓	✓	✗	✓	✓	✓	✓	✓	✓	✓																		
v					1																									

- Total references: 41
- Total distinct references: 10
- Hits: 28
- Faults: 13
- **Hit rate:** 28/41 = **68.292682926829%**
- **Fault rate:** 13/41 = **31.707317073171%**

خروجی FIFO سایت:

Summary - FIFO algorithm

- Total frames: 8
- Algorithm: FIFO
- Reference string length: 41 references
- String: 6 10 9 6 4 4 7 9 9 9 2 10 6 10 7 1 8 3 10 2 4 2 4 2 10 2 5 8 6 6 3 4 10 9 8 8 10 5 2 10 5

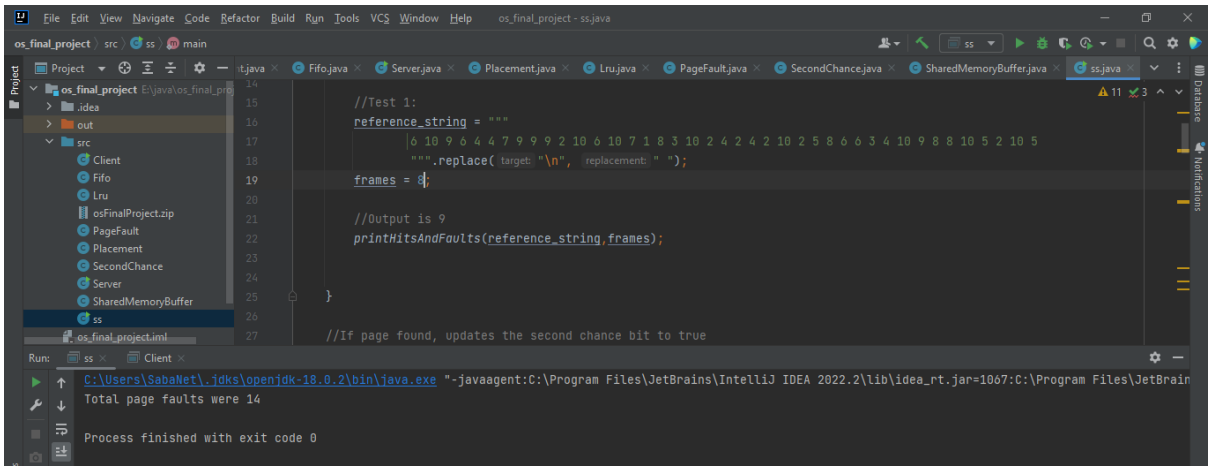
Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
ref		6	10	9	6	4	4	7	9	9	9	2	10	6	10	7	1	8	3	10	2	4	2	4	2	10	2	5	8	6
f		6	10	9	9	4	4	7	7	7	7	2	2	2	2	2	1	8	3	3	3	3	3	3	3	3	3	5	5	6
f			6	10	10	9	9	4	4	4	4	7	7	7	7	7	2	1	8	8	8	8	8	8	8	8	8	3	3	5
f				6	6	10	10	9	9	9	9	4	4	4	4	4	7	2	1	1	1	1	1	1	1	1	8	8	3
f						6	6	10	10	10	10	9	9	9	9	9	4	7	2	2	2	2	2	2	2	2	1	1	8
f								6	6	6	6	10	10	10	10	10	9	4	7	7	7	7	7	7	7	7	2	2	1
f												6	6	6	6	6	10	9	4	4	4	4	4	4	4	4	7	7	2
f																	6	10	9	9	9	9	9	9	9	9	4	4	7
f																		6	10	10	10	10	10	10	10	10	9	9	4
hit		X	X	X	✓	X	✓	X	✓	✓	✓	X	✓	✓	✓	✓	X	X	X	✓	✓	✓	✓	✓	✓	✓	✓	X	✓	X
v																			6								10		9

t	30	31	32	33	34	35	36	37	38	39	40	41																	
ref	6	3	4	10	9	8	8	10	5	2	10	5																	
f	6	6	6	10	9	9	9	9	9	9	9	9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	5	5	5	6	10	10	10	10	10	10	10	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	3	3	3	5	6	6	6	6	6	6	6	6	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	8	8	8	3	5	5	5	5	5	5	5	5	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	1	1	1	8	3	3	3	3	3	3	3	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	2	2	2	1	8	8	8	8	8	8	8	8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	7	7	7	2	1	1	1	1	1	1	1	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
f	4	4	4	7	2	2	2	2	2	2	2	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
hit	✓	✓	✓	X	X	✓	✓	✓	✓	✓	✓	✓																	
v				4	7																								

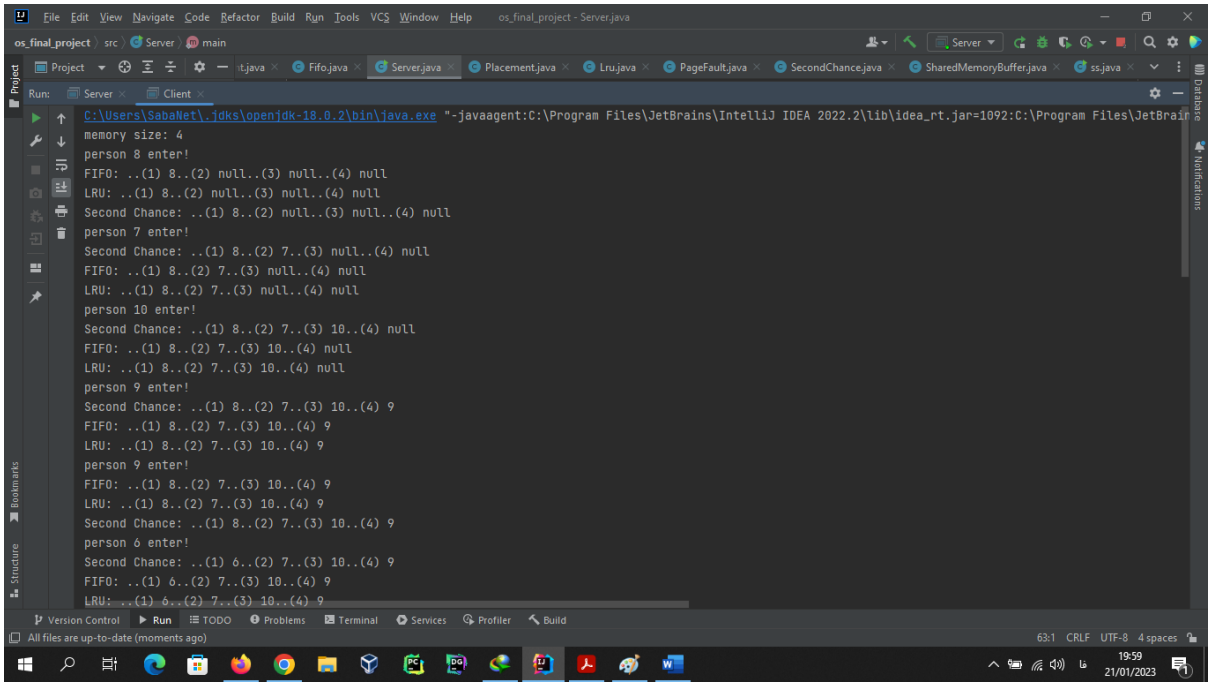
- Total references: 41
- Total distinct references: 10
- Hits: 28
- Faults: 13
- **Hit rate:** 28/41 = **68.292682926829%**
- **Fault rate:** 13/41 = **31.707317073171%**

خروجی Second Chance کد gfg:



تست سوم:

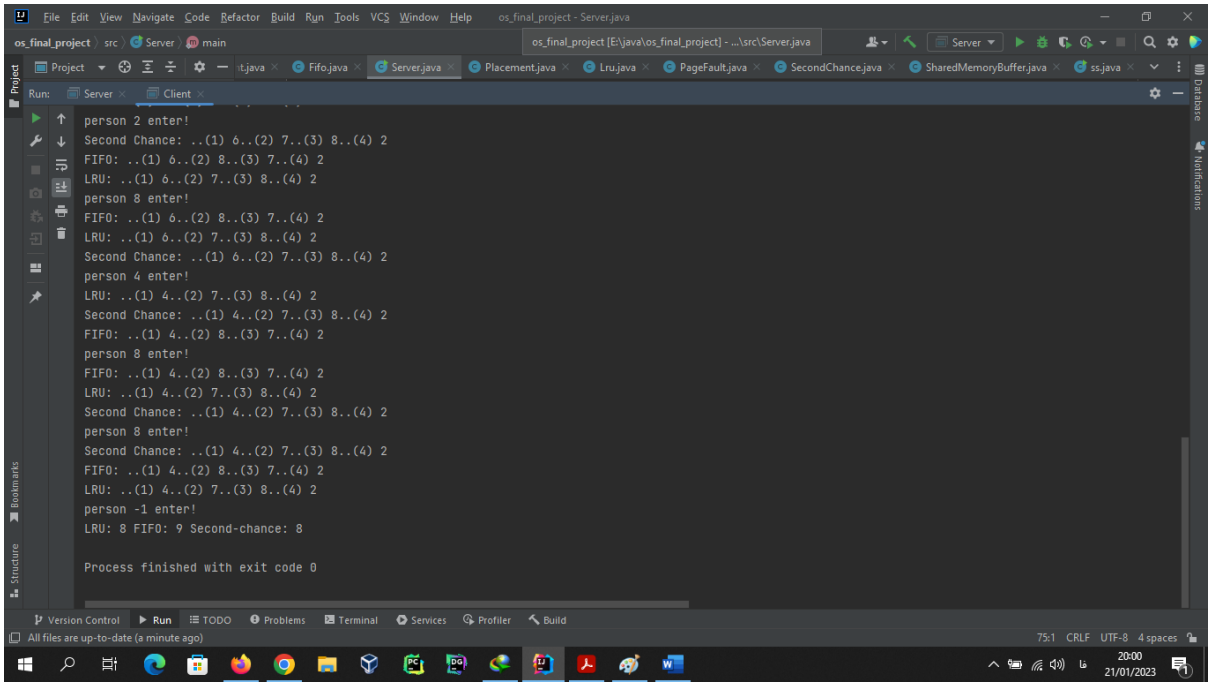
خروجی کلی:



اعداد وارد شده:

۸ ۸ ۴ ۸ ۲ ۸ ۷ ۷ ۸ ۶ ۷ ۶ ۹ ۹ ۱ ۰ ۷ ۸

نتیجه:



خروجی LRU سایت:

Generate: Summary

Column limit: 30

References: #1 8 7 10 9 9 6 7 6 8 7 7 8 2 8 4 8 8

Build schedule | Clear script | JSON

schedule:

Summary - LRU algorithm

- Total frames: 4
- Algorithm: LRU
- Reference string length: 17 references
- String: 8 7 10 9 9 6 7 6 8 7 7 8 2 8 4 8 8

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ref		8	7	10	9	9	6	7	6	8	7	7	8	2	8	4	8	8
f		8	7	10	9	9	6	7	6	8	7	7	8	2	8	4	8	8
f			8	7	10	10	9	6	7	6	8	8	7	8	2	8	4	4
f				8	7	7	10	9	9	7	6	6	6	7	7	2	2	2
f					8	8	7	10	10	9	9	9	9	6	6	7	7	7
hit		X	X	X	X	✓	X	✓	✓	X	✓	✓	✓	X	✓	X	✓	✓
v							8			10				9		6		

- Total references: 17
- Total distinct references: 7
- Hits: 9
- Faults: 8
- **Hit rate:** 9/17 = **52.941176470588%**
- **Fault rate:** 8/17 = **47.058823529412%**

خروجی FIFO سایت:

Summary - FIFO algorithm

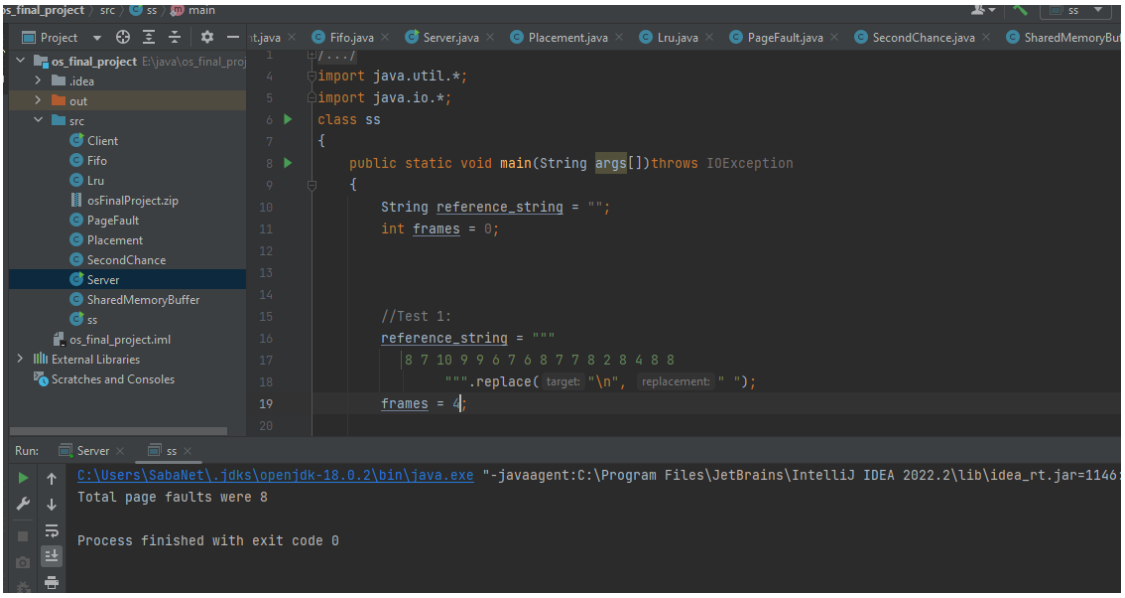
- Total frames: 4
- Algorithm: FIFO
- Reference string length: 17 references
- String: 8 7 10 9 9 6 7 6 8 7 7 8 2 8 4 8 8

Solution visualization

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
ref		8	7	10	9	9	6	7	6	8	7	7	8	2	8	4	8	8
f		8	7	10	9	9	6	6	6	8	7	7	7	2	2	4	4	4
f			8	7	10	10	9	9	9	6	8	8	8	7	7	2	2	2
f				8	7	7	10	10	10	9	6	6	6	8	8	7	7	7
f					8	8	7	7	7	10	9	9	9	6	6	8	8	8
hit		X	X	X	X	✓	X	✓	✓	X	X	✓	✓	X	✓	X	✓	✓
v							8			7	10			9		6		

- Total references: 17
- Total distinct references: 7
- Hits: 8
- Faults: 9
- **Hit rate:** 8/17 = **47.058823529412%**
- **Fault rate:** 9/17 = **52.941176470588%**

خروجی Second Chance کد gfg:



برای بقیه تست ها نیز همین نتیجه درست را میگیریم.