سوال سوم

در این سوال هدف ما پیاده‌سازی یک گراف جهت‌دار است. گراف یک ساختمان‌داده برای ذخیره داده‌های متصل به هم است. به عبارت دیگر یک گراف از مجموعه‌ای از راس‌ها و یال‌هایی که راس‌ها را به یکدیگر متصل می‌کنند تشکیل شده است. برای آشنایی بیشتر با مفهوم گراف می‌توانید لینک زیر را مطالعه کنید:

https://www.tutorialspoint.com/data_structures_algorithms/graph_data_structure.htm

در این‌جا ما یک کلاس به نام Graph داریم که شامل یک Hashmap است که یک راس را به لیستی از رئوس همسایه آن مپ می‌کند.

```
Hashmap<Integer, ArrayList<Integer>>
```

این کلاس شامل متدهایی است که می‌توانند اعمال زیر را انجام دهند:

```
void addVertex (int vertex){}
```

این متد یک راس را می‌گیرد و آن را به گراف اضافه می‌کند.

شکل ورودی:

```
addVertex k
```

```
void addEdge (int source, int destination) {}
```

این متد راس مبدا و مقصد را می‌گیرد و راس مقصد را به لیست همسایه‌های راس مبدا اضافه می‌کند.

شکل ورودی:

```
addEdge s d
```

```
void removeEdge (int source, int destination) {}
```

این متد راس مبدا و مقصد را می‌گیرد و راس مقصد را از لیست همسایه‌های راس مبدا حذف می‌کند.

شکل ورودی:

```
removeEdge s d
```

```
void printGraph(){}
```

این متد رئوس را به صورت مرتب شده به همراه همسایه‌های آن‌ها چاپ می‌کند.

شکل ورودی:

```
print
```

نمونه ورودی و خروجی به شکل زیر است:

با وارد کردن دستور exit برنامه خاتمه می‌یابد.

شکل ورودی:

```
exit
```

```
Input:
addVertex 2
addVertex 5
addVertex 11
addEdge 2 11
addVertex 25
addEdge 5 11
addEdge 11 2
print
removeEdge 2 11
print
addVertex 20
addEdge 2 20
addEdge 20 25
removeEdge 5 11
print
exit
```

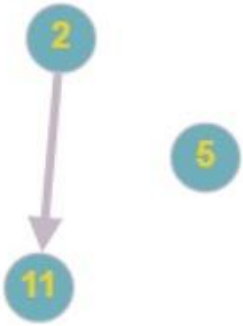
```
output:
Graph {
2 : 11
5 : 11
11 : 2
25 :
}
Graph {
2 :
5 : 11
11 : 2
25 :
}
Graph {
2 : 20
5 :
11 : 2
20 : 25
25 :
}
```

توضیح ساخته شدن گراف با توجه به نمونه ورودی که در بالا ذکر شد:

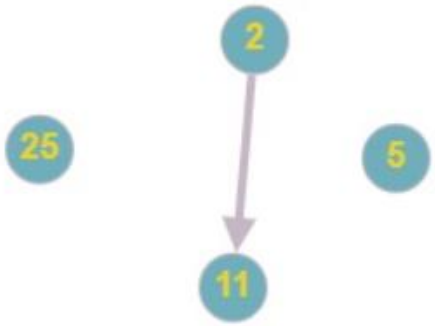
addVertex 2

addVertex 5

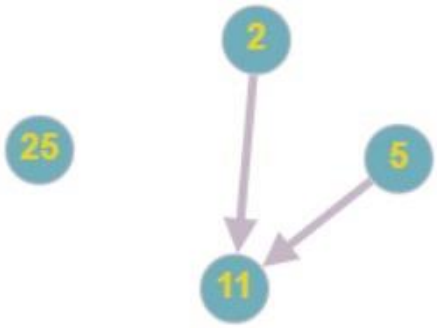
addVertex 11



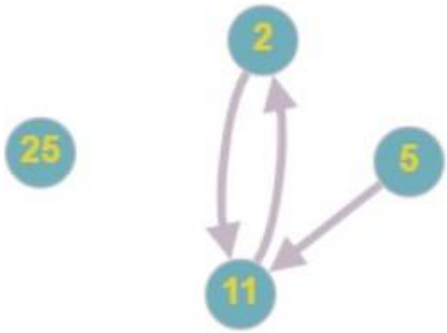
addEdge 2 11



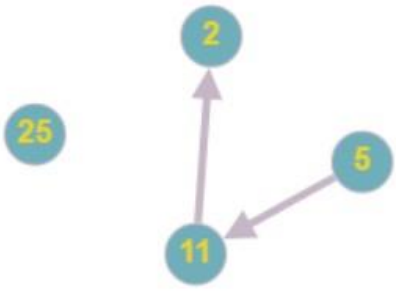
addVertex 25



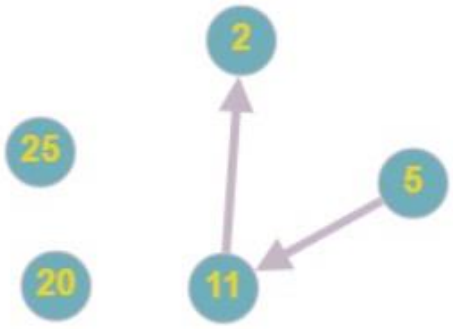
addEdge 5 11



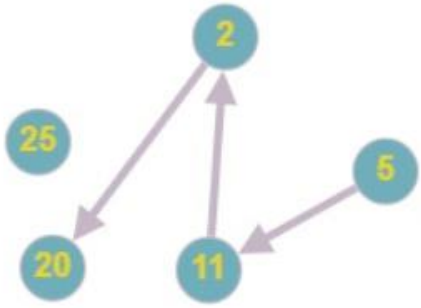
addEdge 11 2



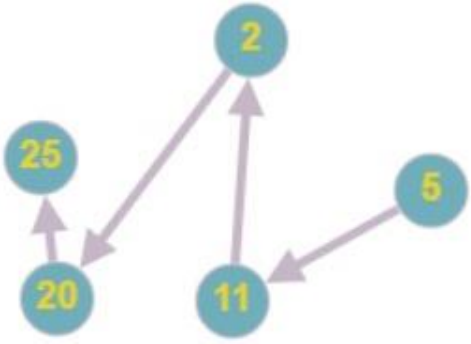
removeEdge 2 11



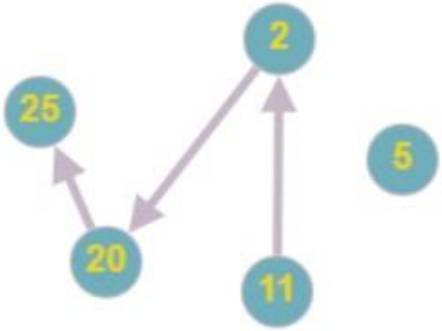
addVertex 20



addEdge 2 20



addEdge 20 25



removeEdge 5 11

سوال چهارم

در این سوال می‌خواهیم یک رستوران پیاده‌سازی کنیم که در ادامه توضیح کلاس‌های آن آمده است. در صورت نیاز می‌توانید متدهایی اضافه کنید که Signature آن را خودتان مشخص می‌کنید.

- **کلاس Person:**

اطلاعات هر فرد مانند نام، نام‌کاربری، موجودی پول و سبد خرید کاربر را در فیلدهای خود ذخیره می‌کند.

- نام دو فرد می‌تواند یکسان باشد اما نام‌کاربری یکتا است.
- موجودی کاربر می‌تواند عددی اعشاری باشد.
- سبد خرید، یک شی از کلاس Basket است.

- **کلاس Product:**

اطلاعات هر محصول از جمله نام، دسته بندی، قیمت، تاریخ و ساعت تولید و تاریخ و ساعت انقضا را در فیلدهای خود ذخیره می‌کند.

- نام یکتا است.

- تاریخ تولید و انقضا باید از کلاس LocalDateTime باشند.

- فرمت تاریخ باید به صورت yyyy-MM-dd HH:mm باشد.

- نباید تاریخ تولید بعد از تاریخ انقضا باشد.

- قیمت محصولات می‌تواند عددی اعشاری باشد.

- دسته بندی تنها شامل سه نوع می‌شود و انواع دیگر قابل پذیرش نیست:

Drink -

Food -

Snack -

- **کلاس Inventory:**

یک HashMap دارد و تمامی محصولات موجود در رستوران را به تعداد موجودی آن محصول map می‌کند. این کلاس باید توانایی کم و اضافه کردن محصولات جدید به انبار را داشته باشد و همچنین بتواند میزان موجودی یک محصول که قبلا به فروشگاه اضافه شده است را تغییر دهد.

- **کلاس Basket:**

سبد خرید شما است و باید قابلیت کم و اضافه کردن محصولات به سبد خرید را داشته باشد و همچنین مجموع قیمت محصولات داخل سبد خرید را نیز محاسبه کند.

- در پیاده‌سازی این کلاس از ArrayList استفاده کنید.

کلاس‌های Product، Person، Inventory و Basket را پیاده‌سازی کنید. همه کلاس‌ها باید متد toString را override کنند و در صورت نیاز با صدا زدن این متد اطلاعات مورد نیاز را چاپ کنند.

- **کلاس Store:**

تقریبا تمام پردازش‌ها و مدیریت منابع در این کلاس انجام می‌شود. به نوعی این کلاس، مدیریت رستوران شما را بر عهده دارد.

- یک Instance از کلاس Inventory با نام inventory دارد.
- یک hashSet از کاربران دارد.

- **کلاس Main:**

در این کلاس، تعامل با کاربر صورت میگیرد. تمام دستورات قابل فهم در این کلاس قرار دارد. تمام اطلاعات از کاربر گرفته می‌شوند و برای انجام پردازش به کلاس Store داده می‌شوند.

- با هربار شروع برنامه باید محصولات زیر را با اطلاعات داده شده مربوط به هر یک بسازید و همه آنها را با موجودی داده شده در HashMap موجود در Inventory ذخیره کنید.

Name	Short Name	Category	Price	Manufacture	Expiration	In Stock
Sheldon's Pizza	Pizza	Food	$19.99	Apr 5, 2022 09:00 PM	Apr 9, 2022 09:00 PM	10
Tom & Jerries Steak	Steak	Food	$100.00	Mar 6, 2022 09:30 PM	Mar 9, 2022 12:00 AM	5
Sponge Bob's Burger	Burger	Food	$15.75	Mar 3, 2022 08:30 AM	Mar 3, 2022 08:30 PM	100
Central Perk's Coffee	Coffee	Drink	$8.50	Mar 1, 2022 01:20 PM	Mar 1, 2022 03:20 PM	50
Ninja Turtles Chocolate Bar	Chocolate	Snack	$17.99	Jan 12, 2020 10:50 AM	Jan 12, 2023 10:50 AM	15
Scooby Doo's Scooby Snacks	Snack	Snack	$3.49	Jun 14, 2021 04:30 PM	Jun 14, 2022 04:30 PM	12
MacLaren's Soda	Soda	Drink	$7.00	Mar 11, 2022 03:30 PM	Jun 11, 2022 03:30 PM	80
Kung Fu Panda's Noodle Soup	Noodle	Food	$35.00	Feb 28, 2022 01:20 PM	Mar 1, 2022 12:00 AM	40

در صورت عدم وضوح تصویر، مشاهده نسخه اصلی

توجه کنید که برای ذخیره کردن نام productها از ستون Short Name این جدول استفاده کنید؛ به عبارتی فیلد نام در کلاس Product همان value ستون Short Name جدول بالا است.

- **نحوه تعامل با کاربر:**

در هنگام شروع برنامه لیستی از دستورات قابل فهم برنامه به کاربر نشان داده می‌شود:

```
[ add.p  balance  people  menu  order  checkout  basket  remove.b  inventory(remove & change)  exit ]
```

دستورات و نحوه استفاده از آن‌ها

- **add.p:**

برای اضافه کردن کاربر، به این صورت به برنامه داده می‌شود:

```
add.p name username balance
```

```
add.p Amirerfan amirerfant 100.2
```

اگر اضافه کردن کاربر موفقیت آمیز بود، پیغام زیر

```
Successfully added user.
```

و در غیر این صورت پیغام زیر چاپ می‌شود.

```
This username already exist!
```

- **balance:**

برای تغییر مقدار موجودی حساب کاربران استفاده می‌شود. به این صورت به برنامه داده می‌شود:

```
balance username amount
```

```
balance farhad1380 525
```

اگر جابجا کردن پول موفقیت آمیز بود، پیغام زیر

```
Successfully transmitted.
```

و اگر کاربر وجود نداشت، پیغام زیر چاپ می‌شود.

```
Invalid username.
```

- **people:**

اگر کاربر این دستور را وارد کند، لیستی از تمام کاربران شامل نام، موجودی و username آن‌ها نشان داده می‌شود.

نمونه:

```
1. name= 'Farhad'  username= 'farhad1380'  balance= 112.75
2. name= 'Amirerfan'  username= 'amirerfant'  balance= 51.0
```

اگر لیست افراد خالی بود، پیغام زیر چاپ می‌شود:

```
There is no one here!
```

- **menu:**

اگر کاربر این دستور را وارد کند، لیستی از تمام آیتم‌های منو شامل نام، دسته بندی، قیمت، تاریخ تولید و انقضاء آن‌ها نشان داده می‌شود.

نمونه:

```
1. name= 'Burger'  category= 'Food'  price= 15.75$  manufactureDate= 2022-03-03 08:30  expirationDate= 2022-03-03 20:30  instock: 100
2. name= 'Noodle'  category= 'Food'  price= 35.0$  manufactureDate= 2022-02-28 13:20  expirationDate= 2022-04-01 00:00  instock: 40
3. name= 'Snack'  category= 'Snack'  price= 3.49$  manufactureDate= 2021-01-14 16:30  expirationDate= 2022-01-14 16:30  instock: 12
4. name= 'Steak'  category= 'Food'  price= 100.0$  manufactureDate= 2022-03-06 21:30  expirationDate= 2022-03-09 00:00  instock: 5
5. name= 'Coffee'  category= 'Drink'  price= 8.5$  manufactureDate= 2022-03-01 13:20  expirationDate= 2022-03-01 15:20  instock: 55
6. name= 'Chocolate'  category= 'Snack'  price= 17.99$  manufactureDate= 2020-01-12 10:50  expirationDate= 2023-01-12 10:50  instock: 15
7. name= 'Soda'  category= 'Drink'  price= 7.0$  manufactureDate= 2022-03-11 15:30  expirationDate= 2022-06-11 15:30  instock: 80
8. name= 'Pizza'  category= 'Food'  price= 19.99$  manufactureDate= 2022-04-05 21:00  expirationDate= 2022-05-09 21:00  instock: 10
```

- **order:**

این دستور برای سفارش دادن استفاده می‌شود و زمانی که کاربر این دستور را وارد می‌کند، قصد دارد محصولی را به سبد خرید خود اضافه کند.

به این صورت به برنامه داده می‌شود:

```
order username food
```

```
order amirerfant Chocolate
```

اگر سفارش موفقیت آمیز بود، پیغام زیر چاپ می‌شود:

```
Successfully added to your basket.
```

اگر username اشتباه بود، پیغام زیر چاپ می‌شود:

```
Invalid username.
```

اگر اسم غذا اشتباه بود، پیغام زیر چاپ می‌شود:

```
Invalid food.
```

در نهایت اگر غذا موجود نبود، پیغام زیر چاپ می‌شود:

```
Out of stock.
```

- **checkout**:

این دستور برای تسویه حساب است. اگر کاربر این دستور را وارد کرد، باید مجموع ارزش سبد خرید او را با مالیت حساب کنید.

- نحوه محاسبه مالیات:

همانطور که گفته شد هر یک از محصولات عضوی از یکی از سه دسته‌ی Drink-Food-Snack است.

دسته‌بندی Drink شامل ۳۵ درصد، دسته‌بندی Food شامل ۱۰ درصد و دسته‌بندی Snack شامل ۲۰ درصد مالیات می‌شود.

دستور «checkout» به صورت زیر به برنامه داده می‌شود:

```
checkout username
```

اگر username اشتباه بود، پیغام زیر

```
Invalid username.
```

اگر موجودی کاربر کافی نبود، پیغام زیر

```
You can't afford your bill.
```

و در نهایت اگر تسویه حساب موفقیت آمیز بود، پیغامی شبیه به پیغام زیر چاپ شده و سبد خرید کاربر خالی می‌شود:

```
Amirerfan's bill= 119.45
tax = 12.45  net = 107.0
Your new balance= 130.55
Have a good day.
```

- **basket**:

محصولات موجود در سبد خرید کاربر را نشان می‌دهد که به صورت زیر به برنامه داده می‌شود:

```
basket username
```

```
basket amirerfant
```

اگر username اشتباه بود، پیغام زیر چاپ می‌شود:

```
Invalid username.
```

نمونه نمایش سبد خرید:

```
Farhad's Basket:
1. Noodle  Food  35.0$
2. Pizza  Food  19.99$
```

- **remove.b**:

این دستور برای حذف جنس موجود در سبد خرید استفاده می‌شود که به صورت زیر به برنامه داده می‌شود:

```
remove.b username index
```

- توجه داشته باشید که index، همان شماره ردیف محصول در دستور مشاهده سبد خرید است. برای مثال شماره ردیف Pizza، دو است.

```
remove.b amirerfant 2
```

اگر username اشتباه بود، پیغام زیر

```
Invalid username.
```

اگر ایندکس معتبر نبود، پیغام زیر

```
Invalid index.
```

در غیر این صورت، پیغام زیر چاپ می‌شود.

```
Successfully removed.
```

- **inventory**:

این دستور برای مدیریت انبار استفاده می‌شود. توجه داشته باشید که این بخش از دسترسی کاربران عادی خارج است و برای ورود به این بخش کاربر باید رمز عبور وارد کند.

- رمز عبور این بخش «ceit-2022» است.

نکته‌ی دیگر این است که بخش inventory دو دستور زیر مجموعه به نام‌های change و remove دارد.

- **دستور زیرمجموعه remove**:

برای حذف کردن یک محصول از منو استفاده می‌شود که به صورت زیر به برنامه داده می‌شود:

```
inventory password remove food
```

```
inventory ceit-2022 remove Coffee
```

اگر حذف کردن موفقیت آمیز بود، پیغام زیر نشان داده می‌شود:

```
Successfully removed.
```

- **دستور زیرمجموعه change**:

برای تغییر دادن موجودی یک محصول در منو استفاده می‌شود که به صورت زیر به برنامه داده می‌شود:

```
inventory password change food amount
```

```
inventory ceit-2022 change Coffee 5
```

اگر تغییر موفقیت آمیز بود، پیغام زیر نشان داده می‌شود:

```
Successfully changed the amount.
```

اگر تغییر تعداد، منفی بود و بزرگتر از مقدار موجودی رستوران بود، پیغام زیر نشان داده می‌شود:

```
Greater than available amount in stock
```

در هر کدام از دو دستور زیر مجموعه بالا اگر رمز عبور اشتباه وارد شد، پیغام زیر نشان داده می‌شود و اجازه ورود داده نمی‌شود:

```
Invalid password.
```

به همین ترتیب اگر در هر کدام از دو دستور بالا نام غذا نیز غیر معتبر باشد، پیغام زیر نشان داده می‌شود:

```
Invalid food.
```

- **exit:**

این دستور برای پایان دادن برنامه است. تا وقتی که کاربر این دستور را وارد نکرده، برنامه ادامه پیدا می‌کند و از کاربر ورودی دریافت می‌کند. نمونه خروجی دستور:

```
Have a nice day, chief.
```

توجه! مستندسازی به کمک Javadoc، کامنت‌گذاری و رعایت اصول کدنویسی خوانا برای همه کلاس‌های پیاده‌سازی شده الزامی است.