به نام هستی بخش

برنامه سازی پیشرفته

نیمسال دوم94-95

تمرین شماره هفتم

دانشکده مهندسی و علوم کامپیوتر

مدرس : دکتر وحیدی اصل

طراح : محسنی – سیدآقایی

تاریخ تحویل: 10 خرداد

سوال اول (35 نمره)

در این پرسش از شما دانشجویان درس برنامه‌سازی پیشرفته دانشگاه شهید بهشتی خواسته شده است که برنامه‌ای برای مدیریت واحدها در بازه حذف و اضافه بنویسید. سه عنصر اصلی این تمرین، دانشگاه، درس و دانشجو هستند.

هر دانشجو دارای سه فیلد نام، نام خانوادگی و شماره دانشجویی می‌باشد. شماره دانشجویی هر فرد یکتا می‌باشد.

هر درس دارای نام، ظرفیت، مجموعه دانشجویان ثبت‌نام شده و همچنین صفی از دانشجویان در حال انتظار می‌باشد. (برای پیاده‌سازی مجموعه از Vector و برای پیاده‌سازی صف انتظار از Queue استفاده کنید)

هر دانشگاه دارای Arraylist از تمام از درس‌های ارائه شده در یک ترم و یک Set از دانشجوها است. دانشگاه، شش تابع اصلی دارد که در تابع main فراخوانی میشوند .دقت کنید که امضای این توابع باید دقیقا منطبق با امضای مشخص‌شده باشد تا در هنگام تست به درستی عمل کند .این شش تابع به شرح زیر میباشند:

- University() سازنده بدون آرگومان

این تابع اطلاعات مربوط به دانشجویان و درسهای ارائه شده را از ورودی استاندارد میخواند. فرمت اطلاعات ورودی به این ترتیب است که ابتدا تعداد دانشجویان دانشگاه (N) در یک سطر آورده می‌شود. سپس در هر یک از N سطر بعدی، اطلاعات یکی از دانشجویان آورده می‌شود. نام، نام‌خانوادگی و شماره دانشجویی فرد که با فاصله از هم جدا شده‌اند، به ترتیب آورده داده می‌شوند. سپس تعداد درس‌های ارائه شده (C) در یک سطر جداگانه داده می‌شود و سپس اطلاعات مربوط به درس‌ها پشت سر هم آورده می‌شود. برای هر درس، ابتدا در یک سطر نام درس داده می‌شود. در سطر بعدی ظرفیت درس، با یک فاصله تعداد دانشجویان ثبت‌شده در درس (S) و سپس تعداد دانشجویان حاضر در لیست انتظار (W) داده می‌شود. در S سطر بعدی شماره دانشجویی افرادی داده می‌شود که با موفقیت در درس ثبت‌نام کرده‌اند و سپس در W سطر بعدی شماره دانشجویی متناظر با افراد منتظر در لیست انتظار داده می‌شود. برای سادگی می‌توانید فرض کنید که ورودی داده شده صحیح است.

- void enroll_course(String course_name, int student_id)

با استفاده از این تابع، دانشجو میتواند در درس ثبت‌نام کند .در صورتی که گنجایش درس تکمیل نشده باشد، دانشجوی مورد نظر به مجموعه‌ی دانشجویان ثبت شده در درس اضافه میشود و در غیر این صورت به صف انتظار وارد میشود.

- void unenroll_course(String course_name, int student_id)

با استفاده از این تابع، دانشجو میتواند درسی را که در آن ثبت‌نام کرده‌است و یا در لیست انتظار آن قرار دارد حذف کند. در صورتی که دانشجو در مجموعه‌ی دانشجویان ثبت‌نام شده باشد، از این مجموعه حذف میشود. و اگر در صف انتظار باشد، از صف خارج می‌شود(پر واضح است که با خارج شدن فرد از صف، افرادی که پشت سر او در صف هستند، یک نفر به جلو حرکت میکنند).

- void increase_capacity(String course_name, int amount)

این تابع، ظرفیت درسی که نام آن به عنوان آرگومان داده شده است، را به مقدار amount افزایش میدهد .با افزایش ظرفیت درس، به همان تعداد اضافه شده، دانشجو از اول صف انتظار خارج شده و به مجموعه‌ی دانشجویان ثبت‌نام شده در درس اضافه میشوند.

- int get_total_waiting()

این تابع، مجموع طول صف‌های انتظار درسهای مختلف را برمیگرداند.

- void registered_courses(int Student_id)

این تابع، نام درسهایی را که دانشجو با موفقیت توانسته در آنها ثبت‌نام کند، در خروجی استاندارد چاپ میکند. ابتدا در یک خط نام و نام خانوادگی دانشجو چاپ میشود .سپس برای هر یک از درسهایی که دانشجو در آن ثبت شده است، در یک سطر اطلاعات درس به فرمت "Name: X registered, Y in queue" چاپ می‌شود.Name نام درس، X تعداد افراد ثبت شده و Y تعداد موجود در صف انتظار است.

نمونه ای از تابع main که کد شما با آن تست می‌شود به شکل زیر است:

```
public static void main(String[] args)
{
    University sbu = new University();
    sbu.enroll_course("AP", 12345678);   // Student has already registered in Golestan.
    sbu.increase_capacity("AP", 1);   // Omits one student from waiting-list and registers him.
    sbu.registered_courses(12345678);
    sbu.registered_courses(87651234);
}
```

نمونه ورودی و خروجی (مطابق کد بالا) :

خروجی	ورودی
Mohamadreza Ahmadi AP: 3 registered, 0 in queue Farzad Rohanifar	5 Mohamadreza Ahmadi 12345678 Niusha Payandeh 87654321 Mehrshayyad Shapoori 43215678 Farzad Rohanifar 87651234 Nima Khamesian 65748392 1 AP 2 2 1 12345678 87654321 43215678

سوال دوم (10 نمره)

در صفحه محورهای مختصات سه بعدی هر نقطه با مولفه‌های x، y و z مشخص می‌شود. کلاسی به نام Vector3D بنویسید که نمایانگر یک نقطه در فضا باشد. N نقطه در فضا به ما داده می‌شود. این نقاط را بر حسب فاصله شان از مبدا به صورت نزولی مرتب کنید و در نهایت به همین ترتیب چاپشان کنید. در صورتی که دو نقطه فاصله یکسان داشتند نقطه ای بزرگتر است که مولفه x آن بزرگتر باشد و به همین ترتیب در صورت برابری x، به ترتیب مقادیر y و z را مقایسه کنید.

راهنمایی: تابع مقایسه گر را برای این کلاس بازنویسی کنید.

ورودی:

در خط اول تعداد نقاط را از کاربر دریافت کنید (N)

در N خط بعدی به ترتیب x و y و z هر نقطه را دریافت کنید.

خروجی:

تمامی نقاط را به صورت مرتب شده چاپ کنید(با حذف نقاط تکراری)

سوال سوم (5 نمره)

برنامه ای بنویسید که دو مجموعه A, B را از کاربر دریافت کند و مجموعه A – B را چاپ کند.

ورودی:

در خط اول N (طول مجموعه A) را از کاربر دریافت کنید.

در N خط بعدی اعضای مجموعه A را دریافت کنید.

در خط بعد M (طول مجموعه B) را از کاربر دریافت کنید.

در M خط بعدی اعضای مجموعه B را از کاربر دریافت کنید.

خروجی:

مجموعه A – B را چاپ کنید.

نمونه ورودی و خروجی:

خروجی	ورودی
3 3	3 1 3 3 2 1 2
3 4 7	8 1 3 3 4 2 7 -1 1 9 1 1 3 2 -1 6 6 6 8

پیشنهاد: سعی کنید از ساختاری استفاده کنید که کد شما از نظر پیچیدگی زمانی بهینه باشد.

سوال چهارم (10 نمره)

دفترچه تلفنی داریم که در آن نام افراد و شماره تلفن آنها موجود است .حال میخواهیم N مورد به موارد نوشته شده در این دفترچه اضافه کنیم.

ورودی:

ابتدا عدد N را از کاربر دریافت کنید.

در N خط بعدی، شماره تلفن و نام فرد به شما داده می‌شوند (با یک space فاصله) اما در هر خط ممکن است اول شماره تلفن داده شود یا اول نام داده شود. (طوری برنامه را بنویسید که Exception رخ ندهد.)

یک نفر ممکن است چندین شماره تلفن داشته باشد. (شماره های مختلف هر فرد در خطوط [کوئری های] مختلف داده می‌شود.)

تعداد ارقام شماره تلفن مشخص نیست.

خروجی:

موارد موجود در دفترچه تلفن را چاپ کنید. (ابتدا نام فرد سپس شماره تلفن های او)

سوال پنجم (40 نمره)

در این تمرین می‌خواهیم یک سیستم انبار داری نوین را برای یک شرکت تولیدکننده مواد غذایی پیاده سازی کنیم !این شرکت محصولات مختلفی تولید میکند و برای محصولات مختلف انبارهای مختلفی دارد (هر انبار فقط یک نوع محصول را در خود جای می‌دهد اما یک نوع محصول ممکن است چند انبار داشته باشد.) برای این که سیستم ما بتواند برای همه انبارها، مورد استفاده قرارگیرد تصمیم گرفته ایم در طراحی آن از کلاس Warehouse که یک کلاس generic است استفاده کنیم. شرح توابع این کلاس در ادامه خواهد آمد.

همه ی کالاهایی که قرار است در انبار های این کارخانه نگهداری شوند از لحاظ طراحی‌های نرم افزار ما از کلاس Product ارث خواهند برد. بنابر این همه ی انبارها فقط کالاهایی که از Product ارث ببرند را ذخیره خواهند کرد.

شرح توابع و اجزای سیستم انبارداری نوین :

کلاس Product:

```
abstract class Product;
```

این کلاس یک کلاس abstract است که هر کالای تولید کارخانه از این کلاس ارثبری می‌کند.

توابع کلاس Product:

```
public Product(int id, NovinDate productDate, NovinDate expiredDate)
```

این تابع constructor این کلاس است که id (این عدد برای هر کالا یکتا خواهد بود) و تاریخ تولید و تاریخ انقضای کالای تولیدشده (تاریخ ها از کلاس NovinDate هستند که توضیحاتش در ادامه می‌آید.) را می‌گیرد و یک کالا با این خصوصیات میسازد. همچنین وضعیت این کالا در هنگام تولید باید NOT_STORED قرار گیرد (در ادامه توضیحات مربوط به وضعیت خواهد آمد.)

وضعیت کالا

```
enum State;
```

این enum وضعیت کالا را نشان میدهد. هر کالا 4 وضعیت متفاوت می‌تواند داشته باشد که هر کدام در ادامه خواهد آمد:

- NOT_STORED: هر کالا از زمانی که تولید می‌شود تا زمانی که هنوز وارد انباری نشده این وضعیت را داراست.
- STORED: زمانی که کالایی به یک انبار منتقل شد این وضعیت را می‌گیرد.
- EXPIRED: زمانی که یک کالا تاریخ گذشته محسوب شد (در ادامه توضیحات آن خواهد آمد) این وضعیت را می‌گیرد.
- SOLD: زمانی که کالایی از انبار برای فروش منتقل می‌شود این وضعیت را می‌گیرد.

کلاس NovinDate:

class NovinDate;

توابع کلاس NovinDate:

public NovinDate(int year, int month, int day);

این تابع constructor این کلاس است که با گرفتن روز و ماه و سال یک تاریخ با این خصوصیات تولید می‌کند.

کلاس Warehouse که محصولات از نوع T را ذخیره میکند.

public class Warehouse ;

توابع کلاس Warehouse:

public Warehouse(String name, int capacity);

این تابع constructor این کلاس است و یک انبار جدید با نام و ظرفیت که به عنوان ورودی به آن داده میشود ایجاد میکند.

public void add(T[] products) ;

این تابع یک آرایه از محصولات نوع T می‌گیرد و این محصولات را به این انبار اضافه می‌کند. قاعدتا تنها محصولاتی که وضعیت آن ها NOT_STORED است می-توانند به انبار اضافه شوند و در غیر این صورت باید پیغام مناسب چاپ شود. در صورتی که با اضافه شدن محصولات جدید تعداد محصولات از ظرفیت انبار بیشتر شود نیز باید پیغام مناسب به کاربر داده شود.

public int control(NovinDate date);

این تابع یک تاریخ می‌گیرد و در این تاریخ همه محصولات درون انبار را بررسی کرده و محصولات تاریخ مصرف گذشته (محصولاتی که تاریخ انقضایشان همان روز است هنوز منقضی نشده اند) را از انبار خارج میکند. (دقت کنید وضعیت این کالا ها باید به روز شود.) در انتها تعداد محصولاتی که در انبار باقی مانده اند برگردانده می‌شوند.

public T[] sendForSale(int num, NovinDate date);

این تابع با گرفتن تعداد و یک تاریخ ابتدا کالاهای تاریخ مصرف گذشته در این تاریخ را از انبار خارج می‌کند. سپس به تعداد آمده در ورودی (num) از محصولات درون‌انبار به اولویت تاریخ انقضا (محصولاتی که تاریخ انقضای آن ها زودتر است زودتر خارج می‌شوند) خارج می‌کند و به لیست محصولات فروخته‌شده اضافه میکند. در صورتی که تعداد محصولات مجاز کمتر از num باشد، باید پیغام مناسب چاپ شود.

Public static Product[] getExpiredProducts();

این تابع همه کالاهای منقضی تولید شده تا آخرین عملیات را برمی‌گرداند.

Public static product[] getSoldProducts();

این تابع همه کالای فروخته شده تا آخرین عملیات را بر می‌گرداند

تاکنون سیستم ما برای هیچ کالای خاصی تعریف نشده بود و صرفا یک سیستم انبارداری بود که به هر کارخانه ای می‌توانستیم بفروشیم و این خاصیت را از این به بعد هم باید سیستم حفظ کند و هر کالایی بتواند به آن اضافه شود! اما با توجه به درخواست یک کارخانه که از گفتن نام آن خودداری می‌کنیم، قرار است دو کالای شیر و ماست را درون سیستم به طور پیش‌فرض قراردهیم. کلاس Milkمعرف کالای شیر(هر شی از این کلاس معادل یک بسته شیر است) و کلاس Yogurt معرف کالای ماست (هر شی از این کلاس معادل یک بسته ماست است!) بدیهی است که هر دو این کلاس ها از کلاسProduct ارث می‌برند. تنها فرقی که این کالاها

دارند این است که کارخانه ی مورد نظر با متد جدید خود که خیلی هم مورد تایید استانداردهای نظارتی نیست توانسته به ماده‌ای دست‌یابد که می‌تواند حداکثر 9 بار به هر بسته ماست اضافه شود. پس از اضافه شدن این ماده به هر بسته ماست تاریخ مصرف گذشته، به مدت 7 روز از تاریخ اضافه شدن این ماده، آن بسته قابلیت مصرف مجدد پیدا می‌کند (درواقع expiredDate برای آن تغییر می‌کند و 7 روز بعد از تاریخ اضافه شدن ماده خواهد شد.) این شرکت زمانی که هر بسته ی ماست، تاریخ گذشته تشخیص داده شد، از ماده فوق استفاده میکند و تاریخ انقضا را به 7 روز بعد از آن تاریخ تغییر می‌دهد (در تابع control یا sendForSale) و آن بسته را از انبار خارج نمی‌کند. شما باید دو کلاس Milk و Yogurt را با خصوصیات فوق پیاده سازی کنید.

توضیحات: همه تاریخ ها در عملیات‌هایی که نیاز به تاریخ دارند به صورت صعودی خواهد بود .

تابع های getter و setter برای فیلدهای state ، id ، productDate ، expiredDate و ... را می‌توانید بنویسید و نیز اضافه‌کردن تابع های دیگر به کلاس‌ها مجاز است(حتی ممکن است مجبور به این کار شوید)!