تاکسی

برای حل این تمرین نیازمند مباحث وراثت و کلاس‌های abstract هستید. (اسلاید ۱۰ و ابتدای اسلاید ۱۱)

برای یک شهرک کوچک، قصد داریم با کمک شما بخشی از یک برنامه‌ی سفارش تاکسی آنلاین را پیاده‌سازی کنیم.

ابتدا فایل را Source دانلود کرده و محتوای آن را ببینید. کارشناسان ترافیک، این شهرک را به ۵ ناحیه‌ی اصلی با شماره‌های ۰ تا ۴ تقسیم‌بندی کرده‌اند و همانطور که در جدول زیر دیده می‌شود، برای رفتن از هر ناحیه به ناحیه دیگر، ضریبی تعیین کرده‌اند که در محاسبه‌ی هزینه‌ی نهایی مورد استفاده قرار می‌گیرد.

Place	۰	۱	۲	۳	۴
۰	۱	۲	۲	۴	۳
۱	۲	۱	۴	۲	۳
۲	۳	۵	۱	۳	۲
۳	۴	۳	۳	۱	۲
۴	۳	۳	۲	۲	۱

برای سادگی، این اطلاعات در قالب ماتریس دوبعدی distance، در کلاس DistanceMap در اختیار شما قرار گرفته است. برای سفارش تاکسی از این برنامه، سه روش

مختلف (TripMethod) وجود دارد و با توجه به روش انتخاب‌شده توسط مسافر، هزینه‌ی سفر وی محاسبه شده و نمایش داده می‌شود.

	هزینه‌ی پایه	بارندگی و ساعت اوج ترافیک	ساعت اوج ترافیک	بارندگی
اقتصادی	۵۰۰۰	۱.۴	۱.۲	۱.۲
ویژه	۱۰۰۰۰	۳	۲	۲
موتور	۴۰۰۰	۱.۵	۲	۰.۸

محاسبه‌ی قیمت برای هر یک از این روش‌ها، با توجه به جدول ضرایب بالا و مقادیر موجود در ماتریس فاصله‌ها، به این صورت انجام می‌شود:

اگر مقدار (هزینه‌ی پایه * مقدار استخراج شده از ماتریس فاصله‌ها) برابر با X باشد، هزینه‌ی نهایی برابر است با

$$X \times ratio$$

که مقدار $ratio$ با توجه به ورودی‌های مسئله از جدول بالا انتخاب می‌شود.

به طور مثال، اگر کاربر قصد سفارش تاکسی اقتصادی برای رفتن از ناحیه ۲ به ۳ در هوای بارانی و ساعت غیراوج ترافیک را داشته باشد، هزینه‌ی سفارش برابر است با :

$$5000 \times 3 = 15000$$

$$15000 \times 1.2 = 18000$$

در پیاده‌سازی کلاس‌ها به نکات زیر توجه کنید:

- کلاس TripHandler باید singleton باشد و تنها نمونه‌ی آن، از طریق متد getInstance قابل دسترس باشد.
- در کلاس TripHandler متد calcPrice را باید پیاده‌سازی کنید که دو ورودی دارد:
  - type: نوع سفارش را تعیین می‌کند. مقادیر مجاز برای این ورودی برابر با bike یا economic و یا vip است. با توجه به مقدار این ورودی، باید نوع مناسب از TripMethod انتخاب شده و در محاسبه‌ی هزینه استفاده شود.
  - params: از نوع TripParam است که شامل ویژگی‌های لازم برای محاسبه‌ی هزینه‌ی سفر (شامل شماره ناحیه مبدا و مقصد، زمان اوج ترافیک بودن یا نبودن و بارانی بودن یا نبودن هوا) است. به سازنده‌ی این کلاس دقت کنید.
- به ازای هر روش سفارش تاکسی، یک پیاده‌سازی برای کلاس TripMethod باید انجام دهید:
  - VIPTripMethod، EconomicTripMethod و BikeTripMethod:
  - متد calcPrice در ورودی یک شی از نوع TripParam می‌گیرد و هزینه‌ی سفر را با توجه به ورودی و روش سفارش تاکسی، محاسبه می‌کند. به عبارت دیگر منطق محاسبه‌ی هزینه برای هر نوع سفارش در این متد پیاده‌سازی می‌شود و در متد calcPrice از TripHandler، این وظیفه به نمونه‌ی مناسب از یکی از پیاده‌سازی‌های TripMethod سپرده می‌شود.

مثال

```
TripHandler taxi = TripHandler.getInstance();

//سفر ویژه از ناحیه 1 به 1 در ساعت اوج ترافیک و هوای غیربارانی
int price = taxi.calcPrice("vip", new TripParam(1, 1, true, false));
int price2 = new VIPTripMethod().calcPrice(new TripParam(1, 1, true, false));

```

```
        System.out.println(price + " = " + price2);

        //سفر با موتور از ناحیه 2 به 4 در ساعت اوج ترافیک و در هوای بارانی
        price = taxi.calcPrice("bike", new TripParam(2, 4, true, true));
        price2 = new BikeTripMethod().calcPrice(new TripParam(2, 4, true, true));
        System.out.println(price + " = " + price2);
```

خروجی نمونه

```
20000 = 20000
12000 = 12000
```

آن چه که باید آپلود کنید

آنچه باید آپلود کنید یک فایل زیپ است که فقط شامل چهار کلاس زیر باشد:

کلاس VIPTripMethod

کلاس BikeTripMethod

کلاس EconomicTripMethod

کلاس TripHandler

پلی‌لیست

برای حل این تمرین نیازمند مباحث وراثت و کلاس‌های abstract هستید. (اسلاید ۱۰ و ۱۱)

می‌خواهیم برنامه‌ای برای مدیریت آهنگ‌هایمان بنویسیم.

فایل Source را دانلود کرده و محتویات آن را به دقت نگاه کنید. کلاس Music شامل فیلدهای زیر است :

- نام آهنگ
- خواننده
- آلبوم
- سال انتشار

کلاس PlayList را با متدهای زیر پیاده‌سازی کنید:

```
1 | public PlayList(int size)
```

برای ساختن یک شی پلی‌لیست باید حداکثر تعداد آهنگ‌هایی که قرار است در آن باشند، مشخص شود.

```
1 | public Music[] getMusics()
```

این متد تمام آهنگ‌های درون پلی‌لیست را به صورت آرایه‌ای از Musicها برمی‌گرداند.

```
public int getNumberOfMusics()
```

این متد تعداد آهنگ‌های موجود در پلی‌لیست را برمی‌گرداند.

```
public boolean addMusic(Music music)
```

شی Music دریافت شده را در صورت پر نبودن پلی‌لیست، به آن اضافه می‌کند و مقدار true را برمی‌گرداند. در غیر این صورت مقدار false را برمی‌گرداند.

```
public PlayList filter(Filter filter)
```

این متد براساس شی Filter دریافت شده، تعدادی از Musicهای آن را در یک شی PlayList جدید جدا، فیلتر می‌کند و پلی‌لیست جدید را برمی‌گرداند. شی PlayList کنونی **نباید** در این متد تغییر کند.
شما باید کلاس‌های FilterByAlbum ، FilterByArtist ، FilterByName و FilterByYear را طوری پیاده‌سازی کنید که هر کدام از این‌ها که به متد Filter داده شد، بر اساس **چند** نام آهنگ یا خواننده یا آلبوم و یا سال انتشار، آهنگ‌ها را فیلتر کند و لیست جدیدی ارائه دهد.

```
public Object[] collectData(DataCollector collector)
```

این متد براساس شی DataCollector دریافت شده، داده‌های موردنظر را از آهنگ‌های شی PlayList کنونی استخراج می‌کند. اندازه‌ی آرایه‌ی بازگردانده شده باید برابر تعداد آهنگ‌های موجود در پلی‌لیست باشد.

بر اساس این که این ورودی از نوع NamesCollector یا ArtistsCollector یا AlbumsCollector و یا ReleaseYearsCollector است، آرایه خروجی شامل اسم آهنگ‌ها یا خواننده‌ها یا آلبوم‌ها و یا سال انتشار آهنگ‌ها است.

```
public Object[] collectDataNoDuplicate(DataCollector collector)
```

این متد دقیقا مشابه متد collectData عمل می‌کند، با این تفاوت که در آرایه‌ی بازگردانده شده توسط آن، عضو تکراری وجود **ندارد** و ترتیب عناصر موجود در آن مهم **نیست**. **(امتیازی ۱۰۰ نمره)**

شما حق هیچ‌گونه تغییری در کلاس‌های DataCollector یا Filter یا Music را ندارید.

مثال

برای فهم بهتر سوال یک نمونه استفاده از آن را در زیر ببینید :

```
PlayList playList = new PlayList(50);
playList.addMusic(new Music("a", "b", "c", 2000));
playList.addMusic(new Music("aa", "bb", "cc", 2000));
playList.addMusic(new Music("aaa", "b", "cc", 2000));
playList.addMusic(new Music("aaa", "b", "ccc", 2010));


```

```
PlayList tmp = playList.filter(new FilterByAlbum("b"));
for (int i = 0; i < tmp.getNumberOfMusics(); i++) {
    System.out.println(tmp.getMusics()[i]);
}

System.out.println();

Object[] names = playList.collectData(new NamesCollector());
System.out.println("length of names : " + names.length);
for (int i = 0; i < names.length; i++) {
    System.out.println(names[i]);
}


System.out.println();

Object[] someYears = playList.filter(new FilterByArtist("c", "cc")).collectData(new ReleaseYearsCollec
System.out.println("length of someYears : " + someYears.length);
for (int i = 0; i < someYears.length; i++) {
    System.out.println(someYears[i]);
}

System.out.println();

//extra
Object[] years = playList.collectDataNoDuplicate(new ReleaseYearsCollector());
System.out.println("length of years : " + years.length);
```

```
        for (int i = 0; i < years.length; i++) {
            System.out.println(years[i]);
        }
```

خروجی

```
[ name : a  Artists : c  Album : b  released : 2000 ]
[ name : aaa  Artists : cc  Album : b  released : 2000 ]
[ name : aaa  Artists : ccc  Album : b  released : 2010 ]

length of names : 4
a
aa
aaa
aaa

length of someYears : 3
2000
2000
2000

length of years : 2
2000
2010
```

آن چه که باید آپلود کنید

یک فایل zip آپلود کنید که دقیقا در آن نه کلاس زیر قرار دارد:

۱. PlayList.java
۲. FilterByName.java
۳. FilterByAlbum.java
۴. FilterByArtist.java
۵. FilterByYear.java
۶. NamesCollector.java
۷. ArtistsCollector.java
۸. AlbumsCollector.java
۹. ReleaseYearsCollector.java

نمودار پلی‌لیست

برای حل این تمرین نیاز به تسلط بر وراثت و مبحث UML می‌باشد. (اسلاید ۱۰)

نمودار *UML Class Diagram* را برای **تمامی** کلاس‌های سوال پلی‌لیست رسم کنید.

لازم نیست که روابط composition و association و aggregation را مشخص نمایید.

آن چه که باید آپلود کنید

یک فایل pdf یا zip شامل تصویر نموداری که رسم کرده‌اید.

شی‌گرایی

برای حل این تمرین نیازمند مباحث وراثت و polymorphism هستید. (اسلاید ۱۰ و ۱۱)

کلاس‌های لازم را به گونه‌ای پیاده‌سازی کنید که متد main زیر اول بدون خطا کامپایل شده و ثانیا خروجی اجرای آن دقیقا به شکلی که گفته‌شده است باشد.

مثال

```
public static void main(String[] args) {
    A[] elements = { new D(), new A(), new C(), new B() };
    for (int i = 0; i < elements.length; i++) {
        System.out.println(elements[i].method1());
        System.out.println(elements[i].method2());
        System.out.println();
    }
}
```

خروجی

```
D1
D1B2
```

```
A1
A2

C1
C1B2

A1
A1B2
```

در پیاده‌سازی خود، موارد زیر را باید رعایت کنید:

- به جز کلاس A، سایر کلاس‌ها تنها یک متد می‌توانند داشته باشند.
- هیچ کلاس و متد اضافه‌ای ایجاد نکنید.
- هیچ ویژگی‌ای (Field) در هیچ کلاسی ایجاد نکنید.
- حتی متد main را هم در کلاس‌های خود قرار ندهید.
- **هدف سوال رسیدن به خروجی موردنظر با رعایت شروط بالا است و در صورتی که خروجی موردنظر تولید نشود یا شرایط برقرار نباشد، به کل سوال نمره‌ای تعلق نمی‌گیرد.**

آن چه که باید آپلود کنید

یک فایل zip است که وقتی آن را باز می‌کنیم در آن کلاس‌های A و B و C و D را ببینیم.

رنگ‌آمیزی (امتیازی) (C++)

- محدودیت زمان: ۲ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

جواد تصمیم گرفته است تغییراتی را در حیاط خانه‌اش ایجاد کند، برای همین از کاشی فروشی سر کوچه، n کاشی هم‌رنگ خریداری می‌کند و همه را در یک ردیف، کنار هم می‌چیند. اما جواد دوست دارد کاشی‌هایش دارای رنگ‌های متنوع باشند. به همین خاطر به رنگ‌فروشی رفته و m رنگ تهیه می‌کند. حال جواد می‌خواهد کاشی‌ها به صورتی رنگ کند که **دقیقا** k کاشی، که رنگ آن‌ها با رنگ کاشی **سمت چپشان** متفاوت است، وجود داشته‌باشد. (کاشی اول شمرده نمی‌شود)

او می‌خواهد بداند که به چند حالت مختلف می‌تواند کاشی‌ها را رنگ کند.

ورودی

در ابتدا اعداد n و m و k داده می‌شود که به ترتیب تعداد کاشی‌ها، تعداد رنگ‌ها و مقدار k (طبق توضیح سوال) هستند.

$$1 \leq n, m \leq 2000$$

$$0 \leq k \leq n - 1$$

خروجی

برای k بیان‌شده، باقی‌مانده‌ی تعداد راه‌های رنگ‌آمیزی کاشی‌ها را بر $10^9 + 1$ چاپ کنید.

ورودی نمونه ۱

```
3 3 0
```

خروجی نمونه ۱

```
3
```

در این مثال، $n = 3$ و $m = 3$ است. به ازای $k = 0$، رنگ هیچ کدام از کاشی‌ها با رنگ کاشی سمت چپیش نباید متفاوت باشد لذا تمام کاشی‌ها یکرنگ هستند که به ۳ حالت مختلف می‌توان آن‌ها را رنگ‌آمیزی کرد.

ورودی نمونه ۲

```
3 2 1
```

خروجی نمونه ۲

```
4
```

در این مثال، $n = 3$ و $m = 2$ و $k = 1$ است که یعنی باید فقط رنگ یک کاشی، با رنگ کاشی سمت چپیش متفاوت باشد.

کارخانه‌ی جواد (امتیازی) (C++)

- محدودیت زمان: ۱ ثانیه
- محدودیت حافظه: ۲۵۶ مگابایت

نمره‌ی سوال ۲۰۰ می‌باشد اما نمره‌دهی سوال، به صورت صفر و یک می‌باشد.

جواد به تازگی یک کارخانه ماشین‌سازی تاسیس کرده است.

از آنجا که در این قبیل کارخانه‌ها تقاضا برای محصولات ماه به ماه متفاوت است، جواد به دنبال یک استراتژی برای برنامه‌ریزی تولیدات کارخانه در n ماه آینده است. می‌توان پیش‌بینی کرد که برای ماه i ام، d_i تقاضا برای ساخت ماشین به کارخانه داده می‌شود. برای پاسخ‌گویی به درخواست‌ها در ماه iام، باید **حداقل** d_i ماشین تولید کرد. اگر در آن ماه تعداد ماشین‌های تولیدی بیش‌تر از d_i شود، تولیدات اضافه در انبار قرار می‌گیرند تا درصورت لزوم، ماه‌های بعد مورد استفاده قرار گیرند. هزینه‌ی نگهداری j ماشین در انبار در **هر ماه**، j ریال است.

کارخانه تعدادی کارگر *تمام‌وقت* دارد که می‌توانند در هر ماه **حداکثر** m ماشین تولید کنند، اما اگر در یک ماه نیاز به تولید بیش از m ماشین باشد، جواد مجبور است تعدادی کارگر *نیمه‌وقت* استخدام کند که به ازای تولید هر ماشین c ریال دستمزد دریافت می‌کنند.

برنامه‌ای بنویسید که براساس داده‌های مسئله، تعیین کند کارخانه‌ی جواد در هر ماه باید **دقیقا** چند ماشین تولید کند تا کم‌ترین هزینه رو بپردازد. شاید بتوان با بیش از یک روش به جواب بهینه رسید. هر جواب درستی، **نمره‌ی کامل** می‌گیرد.

ورودی

ابتدا اعداد n و m و c داده می‌شوند که به ترتیب تعداد ماه‌ها، تعداد ماشین‌های تولیدی توسط کارگران تمام‌وقت در هر ماه و دستمزد هر کارگر نیمه‌وقت هستند.

سپس در خط بعد، n عدد با فاصله می‌آیند که عدد iام، مقدار d_i است.

$$1 \leq n, m, c \leq 100$$

$$1 \leq d_i \leq 100$$

خروجی

در خط اول، $minimum$ هزینه‌ای که جواد باید بپردازد را چاپ کنید.

در خط بعد، n عدد با فاصله چاپ کنید که عدد iام، تعداد ماشین‌های تولیدی در ماه i ام است به‌گونه‌ای که به هزینه‌ی $minimum$ برسیم.

مثال

ورودی نمونه

```
4 5 3
2 6 1 9
```

خروجی نمونه

```
5
3 5 5 5
```