به نام خدا

دستور کار کارگاه برنامه‌نویسی پیشرفته

جلسه سوم

تعامل بین اشیا

مقدمه

جلسه قبل نحوه ساختن و کار با یک شی را یاد گرفتیم. اما برای ساختن یک برنامه کامل، ایجاد شی‌های که به طور مجزا کار می‌کنند، کافی نیست و باید ترکیبی از شی‌هایی را به وجود بیاوریم که با یکدیگر در تعامل هستند. در این جلسه ابتدا یک برنامه ساده را با چند شی می‌نویسیم و نحوه تعامل بین اشیا را فرا می‌گیریم. سپس گروه بندی آن‌ها و استفاده از لیست‌ها را یاد می‌گیریم.

برنامه‌ای که در این جلسه خواهیم نوشت، یک ساعت دیجیتال است که زمان را به صورت مدل ۲۴ساعته نمایش می‌دهد. به نظر شما به چه صورت شی‌ها و کلاس‌ها را در این مسئله تعریف و مدل کنیم؟ چه تعداد شی در این مسئله مورد نیاز است؟ چه ارتباطی باید بین اشیا وجود داشته باشد تا بتوان این مسئله را مدل و پیاده‌سازی کرد؟

برای حل این مسئله باید سعی کنیم طراحی مناسبی برای کلاس‌ها ارائه کنیم تا حل مسئله اصلی که پیچیدگی‌های زیادی دارد، به مدل‌سازی کلاس‌ها و حل مسائل کوچک‌تری تبدیل شود. هر چقدر که برنامه‌ها بزرگ‌تر و پیچیده‌تر می‌شوند، پیاده‌سازی آن‌ها در یک کلاس و اشکال‌زدایی از آن سخت‌تر و پیچیده‌تر می‌شود.

راه حلی که معمولا برای مقابله با پیچیدگی وجود دارد، استفاده از abstraction و modularization است. ما همیشه سعی می‌کنیم مسائل را به مسئله‌های کوچک‌تر و هر مسئله کوچک‌شده را به چند مسئله ساده‌تر تبدیل کنیم تا جایی که حل هر مسئله مجزا، ساده و قابل انجام باشد. به این عمل modularization می‌گویند. برای انجام این کار، باید در هر مرحله، جزئیاتی از مسئله را در نظر نگیریم و پیاده‌سازی آن جزئیات را در هنگام حل مسائل کوچک‌تر موکول کنیم. همچنین، وقتی که یک مسئله کوچک را حل کردیم، دیگر به جزئیات پیاده‌سازی آن توجه نمی‌کنیم؛ بلکه از پاسخ آن قسمت برای حل مسئله بزرگ‌تر استفاده می‌کنیم. به عمل در نظر گرفتن کلاس‌ها به عنوان یک واحد و عدم توجه به جزئیات داخلی آن abstraction می‌گویند.

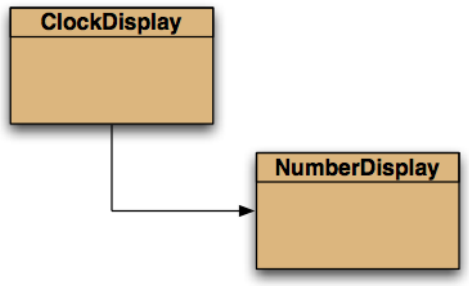
مراحل انجام کار

برای پیاده‌سازی ساعت دیجیتال، می‌توانیم بخش مربوط به ساعت‌شمار، دقیقه‌شمار و ثانیه‌شمار آن را به طور جداگانه مدل‌سازی کنیم. به دلیل نحوه عملکرد مشابه این بخش‌ها، یک کلاس به عنوان اعداد ساعت تعریف می‌کنیم که باید مقدار هر عدد پس از رسیدن به محدودیت مشخص‌شده (limit) مجددا به مقدار صفر بازگردد (برای عدد ساعت این محدودیت عدد ۲۴ است و برای دقیقه و ثانیه عدد ۶۰ است). از طرفی، نمایشگر از سه عدد ثانیه، دقیقه و ساعت تشکیل می‌شود. هر یک از بخش‌ها، باید یک متد increment داشته باشد که مقدار عدد موردنظر را یکی افزایش دهد. نمایشگر یک متد tick دارد که افزایش هر ثانیه از ساعت را مدل‌سازی می‌کند و ثانیه‌شمار را یکی بیشتر می‌کند و در صورت سپری‌شدن شصت ثانیه، دقیقه‌شمار را یکی افزایش می‌دهد و در صورت گذشتن ۶۰ دقیقه، ساعت‌شمار را یکی بیشتر می‌کند. در صورتی که مقدار ساعت‌شمار به ۲۴ برسد، مجدداً همه مقادیر به صفر برمی‌گردد.

برای یکسان‌سازی نحوه نمایش هر بخش، هر کدام از این اعداد باید یک متد print داشته باشند که برای چاپ عدد یک رقمی، ابتدای آن کاراکتر ۰ (صفر) را اضافه کند. نمایشگر بایستی ابتدا ساعت‌شمار را چاپ کرده و سپس یک علامت : چاپ کند و سپس دقیقه‌شمار را چاپ کرده و یک علامت : دیگر چاپ کند و در نهایت ثانیه شمار را چاپ کند. مثال: ۰۶:۲۳:۰۹)

در مرحله بعدی می‌خواهیم نمایشگر ساعت در هر اجرا، زمان فعلی سیستم را بگیرد و از آن زمان شروع به شمارش کند. برای این کار باید کلاس اعداد ساعت، constructorای با ورودی مقدار عدد داشته باشد و نمایشگر دیجیتال نیز باید constructorای با سه ورودی ثانیه، دقیقه و ساعت داشته باشد. نمودار کلاس این مدل‌سازی را در شکل زیر مشاهده می‌کنید.

انجام دهید: برای پروژه clock-display موجود در پیوست دستور کار، پیاده‌سازی ثانیه‌شمار را نیز اضافه کنید.



در ادامه می‌خواهیم روش‌های گروه‌بندی اشیا را (که به آن‌ها collection نیز گفته می‌شود) یاد بگیریم. به طور خاص، در پایان این بخش استفاده از آرایه‌ای از اشیا به عنوان یک collection با اندازه ثابت و استفاده از ArrayList را به عنوان یک collection با اندازه انعطاف‌پذیر فرا می‌گیریم.

در این حین با استفاده از کلاس‌هایی که در کتابخانه‌ها و پکیج‌های مختلف جاوا قرار دارند، خواهیم دید که abstraction به ما کمک می‌کند تا یک مسئله پیچیده را به مسئله‌ای ساده تبدیل کرده و آن را حل کنیم. اکثر برنامه‌نویسان همواره در حال بررسی این هستند که آیا مسائلشان را با استفاده از کتابخانه‌های مختلف موجود قادرند به مسائل ساده‌تری تبدیل کنند و با استفاده از این کار زمان مورد نیاز برای پیاده‌سازی یک برنامه را کاهش دهند.

پیاده‌سازی تعدادی از collectionها در پکیج‌های جاوا وجود دارد. یک collection گروهی از اشیا است که با استفاده از آن می‌توانیم همه اشیا را در قالب یک اجتماع واحد در نظر گرفت و مدیریت کرد.

انجام دهید

در این قسمت می‌خواهیم برنامه‌ای برای دسته‌بندی قطعات مختلف موسیقی بنویسیم. هدف از این برنامه، دسته‌بندی قطعات گوناگون موسیقی به دسته‌های Pop، Jazz، Rock و Country است. برای مشخص کردن هر قطعه موسیقی از آدرس آنها استفاده می‌کنیم. بنابراین باید برای هر قطعه موسیقی، فیلدی از نوع String داشته باشیم که آدرس آن موسیقی را نشان می‌دهد. بنابراین باید چهار ArrayList از نوع String در نظر گرفت که در هر کدام از آن‌ها، قطعه‌های موسیقی آن قرار می‌گیرد.

این برنامه باید اجازه اضافه‌کردن قطعات موسیقی را به صورت نامحدود بدهد (میزان حافظه RAM یک محدودیت خواهد بود. اما آیا می توان این محدودیت را رفع کرد؟). همچنین برنامه باید تعداد قطعه‌های موسیقی هر دسته را اعلام کند، لیست موسیقی‌ها را به کاربر بدهد و امکان حذف یک قطعه موسیقی از لیست را بدهد. همچنین برنامه باید بتواند موسیقی انتخاب‌شده توسط کاربر را پخش کند.

برای پیاده‌سازی این برنامه، باید یک کلاس با نام MusicCollection ساخته شود. این کلاس باید چه متدهایی داشته باشد؟ چه اشیایی باید در این کلاس ساخته شوند؟ (به برنامه پیوست‌شده به دستور کار مراجعه کنید)

برای استفاده از یک کلاس مربوط به یک پکیج، باید اصطلاحاً آن را import کرد. در مثال دسته‌بندی موسیقی‌ها باید کلاس ArrayList را import کنیم:

```
import java.util.ArrayList;
```

با این کار امکان استفاده از کلاس ArrayList و متدهای آن در کد فراهم می‌شود.

برای تعریف یک ArrayList از نوع String که آدرس فایل‌ها را نگه دارد، یک شی از نوع ArrayList می‌سازیم. برای ساختن این شی، باید مشخص کرد که اشیایی که در این collection نگه‌داری می‌شوند از چه نوعی (چه کلاسی) هستند:

```
private ArrayList<String> files;
files = new ArrayList<String>();
```

قسمت <String> مشخص‌کننده این است که اشیای موجود در ArrayList از نوع String خواهد بود.

توجه کنید که کلاس ArrayList یک کلاس عام‌منظوره (generic) است و برای نگه‌داری اشیایی از هر نوعی قابل استفاده است. البته زمانی که بخواهیم از ArrayList یک نمونه ایجاد کنیم، باید مشخص کنیم که این collection برای چه نوع کلاسی مورد استفاده قرار می‌گیرد. در بین < و > می‌تواند هر نوعی از اشیا مشخص شود.

کلاس ArrayList متدهای مختلفی دارد ولی در این بخش فقط با متدهای add، size، get و remove کار خواهیم کرد. در مثال دسته‌بندی موسیقی، این متدها به ترتیب برای اضافه‌کردن فایل جدید به یک دسته، نمایش تعداد فایل‌های یک دسته، پخش موسیقی از یک دسته یا نمایش کل فایل‌ها و حذف یک فایل از یک دسته استفاده می‌شود.

کلاس ArrayList ویژگی‌های زیر را دارد که در حل این مسئله به ما کمک می کند:

- این کلاس می‌تواند در صورت نیاز، ظرفیت خود را به طور پویا افزایش دهد تا تعداد بیشتری شی در خود جای دهد.
- این کلاس تعداد اشیا ذخیره‌شده در خود را در خود نگه‌داری می‌کند.
- این کلاس ترتیب ذخیره‌سازی اشیا را حفظ می‌کند و اشیا به همان ترتیبی که به آن اضافه شده باشند، ذخیره می‌شوند.

یکی دیگر از مزایای abstraction نیز در این‌جا مشخص می‌شود. کلاس MusicCollection برای تعیین تعداد فایل‌های ذخیره‌شده در یک دسته، صرفا از متد ()size کلاس ArrayList استفاده می‌کند.

هر شی‌ای که در ArrayList ذخیره می‌شود، یک index (شاخص) دارد که نشان‌دهنده موقعیت آن شی در collection است. index در آرایه‌ها از صفر (و نه از یک!) شروع می‌شود. بنابراین آخرین شی موجود در یک آرایه، indexای با مقدار 1 – ()size (یکی کمتر از طول آرایه) دارد.

سوال: در متدهای remove و get با استفاده از index می‌توان یک شی را به دست آورد و یا از مجموعه حذف کرد. اگر در این متدها، عددی منفی یا بیشتر از تعداد عناصر آرایه داده شود، چه اتفاقی می‌افتد؟

سوال: اگر از مجموعه ۲۰ تایی عنصر دهم را حذف کنید، index آخرین شی برابر با چه عددی خواهد شد؟ آیا امکان اضافه‌کردن یک شی در بین اشیای دیگر یک ArrayList وجود دارد؟ در این حالت index آخرین شی چه تغییری می‌کند؟

یکی از کارکردهای مورد انتظار از برنامه، چاپ همه قطعات موسیقی مربوط به یک دسته است. برای این کار باید کل آرایه موسیقی‌ها را پیمایش (iterate) کرد. چند راه برای پیمایش‌کردن یک لیست وجود دارد:

۱- استفاده از for-each:

```
for (ElementType element : collection) {
    loop body
}
```

استفاده از for-each راهی برای انجام‌دادن یک کار مشخص برای تک‌تک عناصر یک collection است.

۲- استفاده از for و تغییر index:

```
for (int i = 0; i < collection.size(); i++) {
    loop body
}
```

۳- استفاده از while و index: چگونه با استفاده از while می‌توان یک مجموعه را پیمایش کرد؟

۴- استفاده از نمونه‌ای از کلاس Iterator<>:

Iterator یک کلاس است که امکان حرکت روی عناصر یک collection را فراهم می‌کند.این کلاس در کتابخانه java.util تعریف شده است و یک کلاس generic است (چرا؟).

نحوه استفاده از Iterator به شکل زیر است:

```
Iterator<ElementType> it = myCollection.iterator();
while(it.hasNext()) {
    //call it.next() to get the next element
     ElementType temp = it.next();
    //do something with that element
     temp.doSomething();
}
```

Iterator را می‌توانید به شکل یک اشاره‌گر روی collection در نظر بگیرید. متد hasNext() بررسی می‌کند که آیا عنصر بعدی‌ای وجود دارد یا خیر. متد next() عنصر بعدی را برمی‌گرداند. بنابراین با یک حلقه while ساده می‌توان روی همه عناصر ArrayList یا هر collection دیگری حرکت کرد.

روش‌های index و روش Iterator به نظر شبیه به یکدیگر هستند و تفاوتی ندارند. حتی شاید استفاده از روش‌هایی که با index کار می‌کنند به نظر راحت‌تر باشند، اما جاوا کلاس‌های مختلفی برای collectionها ارائه می‌کند و کار با index روی بعضی از این collectionها مانند Map و Set امکان‌پذیر نیست و یا هزینه زیادی دارد. در حالی‌که روش Iterator در همه کلاس‌های collectionهای مختلف امکان پذیر است و زحمت برنامه‌نویسی را کم‌تر می‌کند.

نکته دیگری که در پیمایش روی collectionها باید در نظر گرفت، حذف عناصر در حین پیمایش است. فرض کنید می‌خواهیم همه قطعات موسیقی‌ای که با A شروع می‌شوند را حذف کنیم. یک روش ساده برای انجام این کار به شکل زیر است:

```
for(String track : myCollection) {
      if(track.startsWith(“A”))
            myCollection.remove(track);
}
```

اما اگر این کار را انجام دهید با خطای ConcurrentModificationException مواجه خواهید شد! زیرا اگر بخواهیم در حین پیمایش روی عناصر، آن‌ها را تغییر دهیم، چه اتفاقی برای فرایند پیمایش روی collection خواهد افتاد؟ اگر عنصر حذف‌شده عنصری باشد که در حال حاضر در حال کار روی آن هستیم، چه رفتاری باید در مورد پیمایش روی collection داشته باشیم؟ توجه کنید که ساختار for-each متوجه نمی‌شود که کدام عنصر حذف شده است؛ زیرا عمل حذف‌کردن روی collection رخ داده است.

روش درست انجام این کار، استفاده از Iterator است. با استفاده از متد remove از کلاس Iterator می توان به آن اعلام کرد که این عنصر قرار است که حذف شود و بنابراین رفتار مناسب برای پیمایش روی collection انجام می شود:

```
Iterator<String> it = tracks.iterator();
while(it.hasNext()) {
     String t = it.next();
     if(t.startsWith(“A”))
            it.remove();
}
```

به این نکته توجه داشته باشید که در هنگام کار کردن با Iterator هیچ‌گاه نباید از collection در داخل حلقه‌ای استفاده کنید و باید گرفتن و حذف عناصر با استفاده از iterator انجام شود.

حال باید در کلاس Main، ۴ نمونه از کلاس MusicCollection برای دسته‌های مختلف بسازید. کد ساخت اشیا به شکل زیر خواهد بود:

```
MusicCollection pop = new MusicCollection();
MusicCollection jazz = new MusicCollection();
MusicCollection rock = new MusicCollection();
MusicCollection country = new MusicCollection();
```

اما اگر تعداد دسته‌های موسیقی ۱۰۰ دسته بود، باز هم به این شکل اشیا مختلف برای هر دسته می‌ساختیم؟ برای این کار می‌توان از آرایه‌های ساده با اندازه ثابت استفاده کرد!

در ترم قبل، ساختن آرایه‌هایی از نوع primitiveها در زبان C را فرا گرفته‌اید. برای مثال برای تعریف آرایه‌ای از عددهای صحیح به صورت زیر عمل می کردید:

```
int arr[3];
```

همین کار در جاوا به شکل زیر انجام می شود:

```
int[] arr = new int[3];
```

و برای تعریف یک collection بر روی یک کلاس نیز به این ترتیب عمل می کنیم:

```
MusicCollection[] categories = new MusicCollection[4];
```

برای پیمایش روی آرایه‌های ساده نیز روش‌های مختلفی وجود دارد:

- استفاده از for-each: به همان شکلی که از for-each برای ArrayList استفاده می‌کردیم، می‌توانیم برای آرایه‌های ساده نیز استفاده کنیم.

```
for (int value : arr) {
    System.out.println(": " + value);
}
```

- می‌توانیم برای پیمایش روی آرایه‌های با طول ثابت نیز از روش‌های index (حلقه‌های for و while) استفاده کنیم.

توجه داشته باشید که برای دسترسی به عناصر در آرایه‌های ساده باید از [ ] استفاده کنیم. برای مثال، برای دسترسی به عنصر اول آرایه categories باید به شکل زیر عمل کرد:

```
categories[0]
```

انجام دهید

۱- برنامه نگه‌داری قطعات موسیقی خود را کامل کنید. در ابتدا فرض کردیم که هر موسیقی یک آدرس به فایل آن موسیقی است. حال سعی کنید هر موسیقی را به یک کلاس شامل فیلدهای آدرس فایل موسیقی، نام خواننده و سال انتشار گسترش دهید.
۲- امکان انتخاب‌کردن موسیقی‌های مورد علاقه و حذف از موسیقی‌های مورد علاقه را به برنامه خود اضافه کنید. همچنین متدی برای نمایش لیست موسیقی‌های مورد علاقه بنویسید.
۳- متدی برای جستجو بین موسیقی‌های یک دسته بنویسید که با گرفتن یک String ورودی، آن را بین موسیقی‌های آن دسته در نام خواننده و آدرس فایل جستجو کند و نتایج را در صفحه نمایش دهد.

اشکال‌زدایی

قطعه کد زیر سه اشکال دارد. آنها را بیابید و فرم صحیح آن را بنویسید.

```
import java.util.ArrayList;

public class MusicOrganizer {
	private ArrayList<String> tracks;

	public void removeTrack(String nameLike) {
		for (int i = 0; i <= tracks.size(); i++)
			if(tracks.get(i).contains(nameLike))
				tracks.remove(i);
	}
}
```

آیا استفاده از دستوری مشابه `tracks.get(i).contains(nameLike)` مجاز است؟ (راهنمایی: chaining method calls)

پاسخ دهید

۱- تفاوت این دو قطعه کد در چیست؟ (راهنمایی: anonymous objects)

قطعه کد اول

```
ArrayList<Student> studentList = new ArrayList<Student>(2);
Student std1 = new Student(“seyed”, 9031806);
studentList.add(std1);
Student std2 = new Student(“saleh”, 9131089);
stringList.add(std2);
```

قطعه کد دوم

```
ArrayList<Student> studentList = new ArrayList<Student>(2);
studentList.add(new Student(“seyed”, 9031806));
stringList.add(new Student(“saleh”, 9131089));
```

۲- می‌خواهیم سیستمی برای ذخیره و بازیابی اطلاعات دانشجویان و نمرات آن‌ها در درس‌های مختلف طراحی کنیم که اساتید هر درس به این سیستم دسترسی دارند. برای این سیستم چه کلاس‌هایی تعریف می کنید؟

۳- سه نمونه از کلاس‌های جاوا برای دسته‌بندی اشیا به همراه کاربرد آن‌ها ذکر کنید.

۴- یک کتابخانه جاوا برای خواندن فایل‌های excel (با فرمت xlsx.) پیدا کنید.