# به نام خدا

# **دستور کار کارگاه برنامه‌نویسی پیشرفته**

**جلسه سوم**

# **تعامل بین شی‌ها**

**مقدمه**

جلسه قبل نحوه ساختن و تعامل با یک شی را یاد گرفتیم. اما برای ساختن یک برنامه کامل ساختن شی های مجزایی که کار میکنند کافی نیست و باید ترکیبی از شی هایی که با یکدیگر تعامل می کنند را بوجود آوریم تا یک عمل را انجام دهیم.

در این جلسه ابتدا یک برنامه ساده با چند شی را می نویسیم و تعامل بین شی ها را فرا میگیریم. سپس گروه بندی شی ها و استفاده از لیست ها را فرا می گیریم.

برنامه ای که در این جلسه خواهیم نوشت یک ساعت دیجیتال است که زمان را به صورت مدل 24 ساعت نمایش خواهد داد. به نظر شما به چه صورت شی ها را در این مسئله تعریف کنیم؟ چه تعداد شی در این مسئله خواهیم داشت؟

راهی که تا بحال فراگرفته ایم استفاده از یک کلاس برای کل برنامه است اما این بار می خواهیم مسئله را طور دیگری حل کنیم. در این مسئله باید سعی کنیم که ببینیم آیا می توانیم کلاس ها را به قسمت های کوچک تری تقسیم کنیم یا خیر؟ دلیل این کار مقابله با پیچیدگی است. هرچقدر که برنامه ها بزرگتر و پیچیده تر می شوند پیاده سازی آنها در یکجا و مراقب بودن از آن برای رخ ندادن اشتباه سخت تر و پیچیده تر خواهد بود. پیاده سازی بعضی کار های ساده با یک کلاس امکان پذیر خواهد بود ولی هرچقدر که یک برنامه بزرگتر می شود مراقبت از رخ ندادن اشتباه و رعایت همه جزئیات کنار هم سخت و پیچیده تر خواهد شد.

راه حلی که معمولا برای مقابله با پیچیدگی وجود دارد استفاده از abstraction و modularization است. ما همیشه سعی می کنیم که مسائل را به مسئله های کوچک تر و هر مسئله کوچک شده را به چند مسئله کوچک تر تبدیل کنیم تا جایی که هر مسئله مجزا انقدر کوچک باشد که برخورد کردن با آن راحت باشد. به این عمل modularization می گویند. وقتی که یک مسئله کوچک را حل کردیم دیگر به جزئیات آن فکر نمیکنیم بلکه از آن برای حل مسائل دیگر استفاده می کنیم. به عمل در نظر گرفتن ماژول ها به عنوان یک شی واحد و توجه نکردن به جزئیات آن abstraction می گویند. به این روش بعضی مواقع divide and conquer هم می گویند.

**نکات آموزشی**

برای پرینت کردن یک متن در خروجی می توانید از تابع System.out.print استفاده کنید.

برای قرار دادن یک متن و عدد کنار یک دیگر در یک متن می توانید از عملگر + استفاده کنید.

**مراحل انجام کار**

برای پیاده سازی ساعت دیجیتال باید یک کلاس به عنوان عدد تعریف کنیم. اعداد ساعت باید هر عدد پس از رسیدن به limit به 0 بازگردد (برای عدد ساعت این limit عدد 24 است و برای دقیقه و ثانیه عدد 60 است). سپس خود نمایشگر از سه عدد ثانیه، دقیقه و ساعت تشکیل می شود. هر عدد یک تابع increment باید داشته باشد که عدد مورد نظر را یکی بیشتر کند. نمایشگر یک تابع tick خواهد داشت که هر ثانیه اجرا می شود و ثانیه شمار را یکی بیشتر می کند و در صورت صفر شدن ثانیه شمار دقیقه شمار را یکی بیشتر می کند و در صورت 0 شدن دقیقه شمار ساعت شمار را یکی بیشتر می کند.

هر کدام از این اعداد باید یک تابع print داشته باشند که در صورتی که عددی که چاپ می کنند یک رقمی باشد اول آن کاراکتر 0 را اضافه کند و در غیر این صورت خود عدد چاپ شود. نمایشگر بایستی ابتدا ساعت شمار را چاپ کرده و سپس یک علامت : چاپ کند و سپس دقیقه شمار را چاپ کرده و یک علامت : دیگر چاپ کند و در نهایت ثانیه شمار را چاپ کند.

در مرحله بعدی می خواهیم نمایشگر ساعت با هر اجرا زمان فعلی سیستم را بگیرد و از آن زمان شروع به شمارش کند. برای این کار باید اعداد تابع سازنده ای با ورودی عدد داشته باشند و نمایشگر دیجیتال باید تابع سازنده ای با سه عدد ورودی ثانیه، دقیقه و ساعت داشته باشد.

## گروه بندی اشیا

در این بخش می خواهیم که روش های گروه بندی اشیا که به آنها collection گفته می شود را یاد بگیریم. به طور خاص در پایان این بخش استفاده از ArrayList را به عنوان یک collection با اندازه قابل انعطاف پذیر و استفاده از آرایه ای از اشیا به عنوان یک collection با اندازه ثابت فرامیگیریم.

در این حین با استفاده از کلاس هایی که در کتابخانه های مختلف جاوا قرار دارند خواهیم دید که Abstraction چقدر به ما کمک میکند تا یک مسئله پیچیده را به مسئله ای ساده تبدیل کرده و آن را حل کنیم. اکثر برنامه نویسان همواره در حال بررسی این هستند که آیا مسائلشان را با استفاده از کتاب خانه های مختلف می توانند به مسائل ساده تری تبدیل کنند و با استفاده از این کار زمان مورد نیاز برای پیاده سازی یک برنامه را کاهش دهند؟

یکی از انواع abstraction ها استفاده از collection ها یا مجموعه هاست. یک مجموعه گروهی از اشیا است که با استفاده از آن می توانیم همه اشیا را هم مدیریت کنیم و به آنها اشاره کنیم. یک مجموعه می تواند خالی باشد، دارای یک شی باشد یا ده ها شی را شامل شود.

در این قسمت می خواهیم برنامه ای برای دسته بندی موسیقی ها بنویسیم. با استفاده از این برنامه باید بتوانیم موسیقی ها را به دسته های Pop، Jazz، Rock و Country تقسیم کنیم.

برای مشخص کردن موسیقی ها از آدرس آنها استفاده می کنیم بنابراین باید برای هر موسیقی یک String داشته باشیم که آدرس آن موسیقی را نشان می دهد.

بنابراین هدف داشتن چهار ArrayList از String است که نشاندهنده موسیقی های دسته های مختلف است.

این برنامه باید اجازه اضافه کردن موسیقی ها به صورت نامحدود را بدهد (میزان حافظه RAM یک محدودیت خواهد بود. اما آیا می توان این محدودیت را رفع کرد؟). همچنین برنامه باید تعداد موسیقی های هر دسته را اعلام کند و لیست موسیقی ها را به کاربر بدهد و امکان حذف یک موسیقی از لیست را بدهد. همچنین برنامه باید بتواند موسیقی انتخاب شده توسط کاربر را پخش کند.

برای پیاده سازی این بخش باید یک کلاس با نام MusicCollection ساخته شود. این کلاس باید چه توابعی داشته باشد؟ چه اشیایی باید در این کلاس ساخته شوند؟

برای استفاده از یک کلاس مربوط به یک کتابخانه باید آن را import کنید. در مثال دسته بندی موسیقی ها باید کلاس ArrayList را import کنیم.

import java.util.ArrayList;

با این کار امکان استفاده از کلاس ArrayList در کد فراهم خواهد شد.

برای تعریف یک ArrayList از نوع String که آدرس فایل ها را نگه دارد یک آبجکت از نوع ArrayList می سازیم. در حین ساخت این Object بایستی اعلام کنیم که اشیایی که در این مجموعه نگه داری خواهند شد از چه کلاسی هستند.

private ArrayList<String> files;

files = new ArrayList<String>();

قسمت <String> مشخص کننده این است که مجموعه از نوع String خواهد بود.

توجه کنید که کلاس ArrayList یک کلاس عام-منظوره است و برای هر کاربردی می توان از آن استفاده کرد ولی زمانی که بخواهیم از آن شی ای ایجاد کنیم باید مشخص کنیم که این مجموعه برای چه دسته کلاسی کاربرد خواهد داشت.

کلاس ArrayList توابع مختلفی دارد ولی در این بخش فقط با توابع add، size، get و remove کار خواهیم کرد.

این توابع در مثال دسته بندی موسیقی به ترتیب برای اضافه کردن فایل جدید، نمایش تعداد فایل ها، پخش موسیقی یا نمایش کل فایل ها و حذف یک فایل استفاده خواهند شد.

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

* این کلاس می تواند در صورت نیاز ظرفیت خود را افزایش دهد تا تعداد بیشتری شی در مجموعه خود جای دهد.
* این کلاس تعداد اشیا ذخیره شده در مجموعه را در خود نگه داری می کند.
* این کلاس ترتیب ذخیره سازی اشیا را حفظ می کند و اشیا به همان ترتیبی که به مجموعه اضافه شده باشند ذخیره می شوند.

یکی دیگر از مزایای Abstraction نیز در اینجا مشخص می شود. کلاس MusicCollection نیازی به ذخیره تعداد فایل هایی که ذخیره شدند ندارد و می تواند این کار را به عهده کلاس ArrayList بگذارد.

کلاس هایی که از <> استفاده می کنند کلاس های Generic هستند. این کلاس ها سعی کردند امکاناتی را برای هر نوع کلاسی فراهم کنند. برای مثال کلاس ArrayList یک کلاس Generic است که این امکان را به ما می دهد که از آن برای ساختن مجموعه ای از هر نوع شی ای استفاده کنیم

هر شی ای که در مجموعه ذخیره می شود یک عدد دارد که نشان دهنده موقعیت آن در مجموعه است که به آن index می گویند. index در مجموعه ها از 0 شروع می شود و بنابراین اولین شی ای که به مجموعه اضافه می شود index صفر را میگیرد و شی بعدی که اضافه شود index یک را می گیرد. بنابراین اخرین شی در مجموعه index=size – 1 دارد.

در توابع remove و get با استفاده از index می توان یک شی را بدست آورد و یا از مجموعه حذف کرد.

اگر در این توابع عددی منفی یا بیشتر از تعداد اعضا مجموعه بدهید چه اتفاقی می افتد؟

اگر از مجموعه 20 تایی عضو 10 ام را حذف کنید index اعضا دیگر جابجا می شود و بنابراین شی با index=19 مقدار index=18 را خواهد داشت. همچنین امکان اضافه کردن یک شی در وسط مجموعه با تابع add وجود دارد و در این حالت index اشیایی که بعد از عضو اضافه شده یکی بیشتر خواهد بود.

با استفاده از index می توان به شی بعد و قبل هر شی دسترسی داشت و از آنها استفاده کرد. همچنین می توان همه عناصر مجموعه را iterate کرد.

یکی از ویژگی هایی که برنامه باید داشته باشد چاپ همه موسیقی های مربوط به یک دسته است. برای این کار باید کل مجموعه موسیقی ها را iterate کرد. چند راه برای iterate کردن یک لیست وجود دارد:

1. استفاده از for-each:

for(ElementType element : collection) {loop body}

استفاده از for-each یک راه انجام دادن یک کار برای همه اعضا مجموعه است. for-each یکی از حالت های حلقه های for است که به شکل بالا نوشته می شود.

1. استفاده از for و index:

for(int i, i < collection.size(); i++) {

loop body

}

1. استفاده از while و index: (چگونه با استفاده از while می توان یک مجموعه را iterate کرد؟)
2. استفاده از Iterator<>:

Iterator یک کلاس است که امکان حرکت روی اعضا یک مجموعه را فراهم می کند.این کلاس در کتابخانه java.utils 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 elementdo something with that element}

Iterator را می توانید به شکل یک اشاره گر روی مجموعه ببینید. تابع hasNext نشان می دهد که آیا عضو بعدی ای وجود دارد یا خیر و تابع next عضو بعدی را برمیگرداند. بنابراین با یک حلقه while ساده می توان روی همه عناصر مجموعه حرکت کرد.

روش های index و روش Iterator به نظر شبیه به یکدیگر هستند و تفاوتی ندارند. حتی شاید استفاده از روش هایی که با index کار می کنند به نظر راحت تر باشند، اما جاوا کلاس های مختلفی برای مجموعه ها ارائه می دهد و کار با index روی بعضی از این مجموعه ها امکان پذیر نیست و یا هزینه زیادی دارد ولی روش Iterator در همه کلاس های مجموعه ها امکان پذیر است و به خوبی کار می کند.

نکته دیگری که در حرکت کردن روی مجموعه ها باید در نظر گرفت حذف عناصر در هنگام حرکت کردن روی مجموعه است. فرض کنید می خواهید همه عناصری که با A شروع می شوند را حذف کنید. یک روش راحت برای انجام این کار به شکل زیر است:

for each track in the collection {if track.startsWith(“A”):collection.remove(track)}

اما اگر این کار را انجام دهید با خطا ConcurrentModificationException مواجد خواهید شد. زیرا اگر ما بخواهیم در میان حرکت کردن روی اعضا مجموعه انها را تغییر دهیم چه اتفاقی برای فرآیند حرکت کردن روی مجموعه خواهد افتاد؟ اگر عضو حذف شده عضوی باشد که در حال حاضر در حال کار روی آن هستیم چه رفتاری باید در مورد حرکت روی مجموعه داشته باشیم؟ (توجه کنید که for-each متوجه نمی شود که کدام عنصر حذف شده است زیرا عمل حذف کردن روی collection رخ داده است.) حذف کردن یک عنصر در میان حرکت روی مجموعه حالت های گیج کننده ای را بوجود می آورد و بنابراین امکان پذیر نخواهد بود.

روش درست انجام دادن این عمل استفاده از Iterator است. با استفاده از تابع remove کلاس Iterator می توان به آن اعلام کرد که این عضو قرار است که حذف شود و بنابراین رفتار مناسب برای حرکت روی مجموعه 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 هیچ گاه نباید از collection در داخل حلقه استفاده کنید و باید گرفتن عناصر و حذف عناصر را با استفاده از شی iterator انجام دهید.

با اضافه کردن متغیر category به این کلاس و با استفاده از روش هایی که تا اینجا فراگرفته اید می توانید کلاس MusicCollection را کامل کنید.

حال باید 4 عدد شی از کلاس MusicCollection برای دسته های 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 دسته های موسیقی 100 دسته بود باز هم به این شکل اشیا مختلف برای هر دسته می ساختیم؟

برای این کار می توان از آرایه ها (که مجموعه هایی با اندازه ثابت هستند) استفاده کرد.

در ترم قبل ساخت آرایه هایی از نوع primitive ها در C را فرا گرفته اید. برای مثال برای تعریف آرایه ای از نوع عدد به صورت زیر عمل می کردید:

int i[3];

همین کار در جاوا به شکل زیر انجام می شود.

int[] i = new int[3];

و برای تعریف یک مجموعه بر روی یک کلاس نیز به همان شکل بالا عمل می کنیم.

MusicCollection[] m = new MusicCollection[4];

برای حرکت روی آرایه ها نیز روش های مختلفی وجود دارد:

* استفاده از for-each: به همان شکلی که از for-each برای ArrayList استفاده می کردیم می توانیم برای آرایه ها نیز استفاده کنیم.

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

* می توانیم برای حرکت روی آرایه ها از روش های index (for و while) استفاده کنیم.

توجه داشته باشید که برای دسترسی به عناصر در آرایه ها باید از [] استفاده کنیم. برای مثال برای دسترسی به عضو اول آرایه m باید به شکل زیر عمل کرد:

m[0]

حال می توانید برنامه دسته بندی 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 که شامل آدرس فایل موسیقی، نام خواننده و سال انتشار است تبدیل کنید.

تابعی برای جستجو بین موسیقی های یک دسته بنویسید که با گرفتن یک String آن را بین موسیقی های آن دسته برای نام خواننده و آدرس فایل جستجو کند و نتایج را در صفحه نمایش دهد.

**اشکال‌زدایی**

حداقل یک قطعه کد مرتبط با دستور کار و هدف از نوشتن آن کد مشخص شود. سپس از دانشجو خواسته شود تا پس از فهمیدن جزئیات دقیق کد، عیب‌یابی و اشکال‌زدایی کند تا هدف مورد نظر توسط کد برآورده شود. در صورتی که لازم است می‌توان از ایده pair assessment نیز استفاده کرد؛ یعنی کد دانشجوی دیگری را ارزیابی کند و اصول برنامه‌نویسی درست را بررسی کند.

# نکاتی که باید در همه مراحل رعایت شود

در حین انجام مراحل این دستورکار بخشی از هر یک از موارد زیر به دانشجویان آموزش داده شود و صحت انجام آن‌ها توسط مدرس کارگاه بررسی و ارزیابی شود:

۱. اصول و مفاهیم Clean Code

۲. استفاده از Git و commitهای منطقی

۳. نکاتی از Effective Java

\* ابتدای هر جلسه تمرین‌های مربوط به هفته گذشته بررسی شود و راه‌حل کلی هر یک از آن‌ها مرور شود. سپس دستور کار جدید بحث شود.