MyArrayList

در این تمرین قصد داریم تا کلاس ArrayList را توسعه دهیم.

همانطور که در نمودار فوق قابل مشاهده است کلاس MyArrayList از کلاس ArrayList ارث‌بری می‌کند و در نتیجه ویژگی‌های آن را دارا می‌باشد. حال علاوه بر ویژگی‌های کلاس ArrayList می‌خواهیم سه تابع زیر را نیز به MyArrayList اضافه کنیم:

- تابع search
- تابع getQueue
- تابع getStack

کلاس MyArrayList

کلاس MyArrayList از کلاس ArrayList ارث‌بری می‌کند اما همانطور که میدانید کلاس ArrayList یک کلاس generic است. در این تمرین تنها قصد داریم اعداد صحیح را در لیست خود ذخیره کنیم در نتیجه به صورت خاص از ArrayList<Integer> ارث‌بری می‌کنیم:

```
public class MyArrayList extends ArrayList<Integer> {
    //TODO
}
```

تابع search

این تابع دارای یک پارامتر ورودی به نام data است. خروجی این تابع یک لیست از تمام index هایی است که data در ArrayList ما وجود دارد.

تابع getQueue

این تابع ورودی ندارد و به عنوان خروجی یک Queue برمی‌گرداند که داده‌های ArrayList به ترتیب در آن push شده اند.

تابع getStack

این تابع ورودی ندارد و به عنوان خروجی یک Stack برمی‌گرداند که داده‌های ArrayList به ترتیب در آن push شده اند.

توضیحات

- با توجه به اینکه این تمرینات با test case تصحیح خواهد شد در نحوه اسم‌گذاری دقت کنید.
- در ابتدای فایل خود package تعریف نکنید.

Error

در برنامه‌نویسی **کلاینت-سرور** برنامه‌ها به دو دسته تقسیم میشن. برنامه سمت سرور و برنامه سمت کلاینت. برنامه‌ای که سمت سرور قرار داره یه برنامه واحد هست یعنی یک سرور داریم که برنامه روش در حال اجرا شدن هست اما برنامه سمت کلاینت روی دستگاه‌های مختلف اجرا میشن مثلا گوشی‌های اندروید رو در نظر بگیرید. یک برنامه اندروید مثل اینستاگرام داریم که روی گوشی‌های مختلف نصب شده. از طرفی یه برنامه سمت سرور داریم که همه اطلاعات روی اون قرار داره و روی یک سرور در حال اجراست. تمام برنامه‌های سمت کلاینت برای اینکه اطلاعاتی رو به کاربر نشون بدن اول یک **درخواست** به سرور می‌فرستند و مثلا ازش می‌خوان که عکس‌های مربوط به فلان اکانت رو براشون بفرسته. سرور درخواست رو دریافت می‌کنه و اگه مشکلی وجود نداشته باشه یک **پاسخ** به سمت کلاینت ارسال می‌کنه مثلا تو مثال اینستا یه تعداد عکس رو به سمت کلاینت می‌فرسته.

تو این فرآیند ارسال **درخواست** و گرفتن **پاسخ** ممکنه اشکالات متفاوتی پیش بیاد مثلا اینکه اینترنت کند باشه یا درخواستی که فرستادیم مشکل داشته باشه و ... . تو این شرایط سرور یک **Error** رو به عنوان پاسخ ارسال می‌کنه. حال ما به عنوان یک توسعه‌دهنده برنامه سمت کلاینت وظیفه داریم وقتی خطا رو دریافت می‌کنیم به صورت درست تفسیرش کنیم و مخاطب نشون بدیم. مثلا وقتی که اینترنت کند هست، سرور خطای Time Out می‌فرسته. وقتی که ما این پیام رو دریافت می‌کنیم باید به کاربر پیام بدیم که اینترنت مشکل داره و بعد از اتصال به اینترنت دوباره تلاش کنه.

حالا تو این تمرین قصد داریم که این سیستم خطا رو مدل‌سازی کنیم و با توجه به خطایی که از سرور میاد یک شیئ متناسب تولید کنیم. نمودار کلاس Error به صورت زیر خواهد بود:

بعضی از خطاها شناخته شده هستند مثلا همون خطای Time Out که به خاطر کندی اینترنت هست ما برای این خطاها یک کلاس مجزا ایجاد می‌کنیم که کد و پیام اونها از قبل مشخص هست.

کلاس‌های زیر از کلاس Error ارثبری می‌کنند:

- کلاس Bad Request
  - کد : 400
  - پیام : Bad Request
- کلاس Forbidden
  - کد : 403
  - پیام : Forbidden
- کلاس InternalServerError
  - کد : 500
  - پیام : Internal Server Error
- کلاس NotFound
  - کد : 404
  - پیام : Not Found
- کلاس RequestTimeOut
  - کد : 408
  - پیام : Request Time Out

توجه کنید که **سازنده** هر یک از کلاس‌های فرزند تنها یک پارامتر ورودی priority دارد. این کلاس‌ها هیچ تابع اضافه‌ای نسبت به تابع پدر ندارند و تنها پارامترهای سازنده آنها کمتر است.

کلاس ErrorList

ما این کلاس را برای ذخیره مجموعه‌ای از خطاها در نظر می‌گیریم.

همانطور که در دیاگرام این کلاس مشاهده می‌کنید، این کلاس شامل یک متغیر از جنس لیست است که خطاها رو در خود ذخیره می‌کنه. توابع این کلاس به صورت زیر هستند:

- تابع addError : یک Error به لیست خطاها اضافه می‌کند.
- تابع sortErrors : خطاها را با توجه به priority آنها مرتب می‌کند توجه کنید که هر چه اولویت بالاتر باشد خطا در موقعیت ابتدایی تر در لیست قرار می‌گیرد.
- تابع getErrors : لیست خطاها رو برمی‌گردونه.

توضیحات

- کد خود را به صورت زیر پوشه‌بندی کنید و آپلود نمایید:

```
.
└── error
  ├── BadRequest.java
  ├── Error.java
  ├── ErrorList.java
```

```
├── Forbidden.java
├── InternalServerError.java
├── NoutFound.java
└── RequestTimeOut.java
```

- در نامگذاری توابع و متغیرهای خود دقت کنید.

تشخیص خطا

این یک سوال تشریحی است که در آن قصد داریم خطای کدها را مشخص کنیم. در هر یک از موارد زیر خطای کد را مشخص کنید و تصحیح شده کد را ارائه دهید. هر یک از کدها را در داخل یک **فایل** جاوا در نظر بگیرید.

خطای اول

```
public static class ErrorDetection {

    private int error;

    public ErrorDetection(int error) {
        this.error = error;
    }

    public void setError(int error) {
        this.error = error;
    }

    public int getError() {
        return error;
    }
}
```

خطای دوم

```
public class ErrorDetection {

    ErrorMessage errorMessage;

    public ErrorDetection() {
        errorMessage = new ErrorMessage();
    }

    public static void main(String[] args) {
        ErrorDetection errorDetection = new ErrorDetection();
        ErrorMessage errorMessage = new ErrorMessage();
    }

    public class ErrorMessage {

    }
}
```

خطای سوم

```
public final class ErrorDetection {

    private int error;

    public void setError(int error) {
        this.error = error;
    }

    public int getError() {
        return error;
    }
}

class SpecialErrorDetection extends ErrorDetection {

    private int code;

    public void setCode(int code) {
        this.code = code;
    }

```

```
    public int getCode() {
        return code;
    }
}
```

خطای چهارم

```
public interface ErrorDetection {
    private int getMessage();
}

interface SpecialErrorDetection extends ErrorDetection {
    public int getSpecialCode();
}
```

سوالات وراثت

به سوالات زیر به صورت تشریحی پاسخ دهید:

سوال اول

دو کلاس زیر را در نظر بگیرید:

```java
public class A {
    private int i;
    protected int j;

    private int getFirstData() { return i; }
    protected int getSecondData() { return j; }
}
```

```java
public class B extends A {

}
```

- آیا می‌توان از متغیر i در کلاس B استفاده کرد؟ از متغیر j چطور؟
- آیا می‌توان تابع getFirstData را در کلاس B، بازنویسی (override) کرد؟ در مورد تابع getSecondData چطور؟
- آیا می‌توان تابع getFirstData را در کلاس B، فراخوانی کرد؟ در مورد تابع getSecondData چطور؟
- در صورت ساخت یک شیئ از کلاس A آیا می‌توان تابع getFirstData را برای آن فراخوانی کرد؟ درمورد getSecondData چطور؟

سوال دوم

دو کلاس زیر را در نظر بگیرید:

```java
public class A {
    private int i;

    public void setI(int i) {
        this.i = i;
    }

    public void printData() {
        System.out.println(i);
    }
}
```

```java
public class B extends A{

    private int k;

    public void setK(int k) {
        this.k = k;
    }

    @Override
    public void printData() {
        System.out.println(k);
    }
}
```

حال تابع main زیر را در نظر بگیرید:

```java
public static void main(String[] args) {
    B b = new B();

    b.setI(1);
    b.setK(2);
    b.printData();
}
```

- خروجی کد فوق چه خواهد بود؟
- برای آنکه بخواهیم در تابع printData ابتدا تابع printData از کلاس پدر صدا زده شود و بعد سایر بدنه متد چه کاری می‌توانیم انجام دهیم؟ (یعنی اول i چاپ شود و بعد k)

حال تابع main زیر را در نظر بگیرید:

```
public static void main(String[] args) {
    A a = new B();
    a.setK(3);
}
```

- آیا کد فوق صحیح است؟ در صورتی که کد فوق دارای خطا است، خطای مربوطه را توضیح دهید.