

task 1 流API

这个任务中我们学习流API，不过在这之前，我们会先向你介绍一些很基本的方法。要使用流API,我们需要一个**Stream**对象。如果我们有一个类似List的集合，它没有实现**Stream**。不过**Collection**接口有一个**Stream**方法，会返回对应的那个Collection的一个Stream对象。现在我们尝试以下的操作：

```
List<String> strings = List.of("I", "am", "a", "list", "of", "Strings");
Stream<String> stream = strings.stream();
//调用流API的方法，例如我们希望最多有4个元素
Stream<String> limit = stream.limit(4);
//最后我们打印结果
System.out.println("limit = " + limit);
```

我们会得到什么结果，结合资料思考一下为什么会得到这样的结果，而不是我们所期望的结果？

- 答：因为流是lazy的，只有在等到终止操作后才会执行中间操作，而上述代码没有终止操作，所以limit中间操作不会执行，所以得不到所期望的结果。
- 代码如下：

```
import java.util.List;
import java.util.stream.Stream;

public class Limit_ {
    public static void main(String[] args) {
        List<String> strings = List.of("I", "am", "a", "list", "of",
"Strings");
        Stream<String> stream = strings.stream();
        ////调用流API的方法，例如我们希望最多有4个元素
        //Stream<String> limit = stream.limit(4);
        ////最后我们打印结果
        //System.out.println("limit = " + limit);

        stream.limit(4).forEach(System.out::println);
    }
}
```

```
• D:\Java\jdk-17\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2024.2.3
I
am
a
list

Process finished with exit code 0
```

任务：再学习一下流API,修改上面的代码使之生成我们想要的结果。你可以尝试创建自己的流，并总结一下使用流的规则。

tips: 你可以看到在处理一些流的过程中我们将操作堆叠在一起，放在一个流管道里，就像这样：

```
List<Integer> squaresList = numbers.stream()
    .map(i -> i * i) //你可能会对这种表达式有兴趣
    .sorted((x, y) -> y - x) //你可能会对这种表达式有
    兴趣
    .collect(Collectors.toList());
```

学习一下这种操作堆叠的原理，并结合上面使用流API的规则总结规律。

- **答：**流API代表了一系列的数据，但它不储存数据。在执行代码时，它会按照顺序沿管道前进，只有等一个数据完整地通过管道下一个数据才会进入。流的这种代码大大简化了原来的代码，也增强了代码的可读性。

- 代码如下：

```
List<Integer> numbers = List.of (1, 2, 3, 4, 5, 6, 7, 8, 9, 10 );
List<Integer> squaresList = numbers.stream()
    .map(i -> i * i) //你可能会对这种表达式有兴趣
    .sorted((x, y) -> y - x) //你可能会对这种表达式有兴趣
    .collect(Collectors.toList());
System.out.println(squaresList);
```

```
D:\Java\jdk-17\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2024.2.3\
[100, 81, 64, 49, 36, 25, 16, 9, 4, 1]

Process finished with exit code 0
```

或许你会对上面注释的部分感兴趣，这叫做**Lambda**表达式，强烈建议你深入掌握这种技巧，这会对你的代码质量有很大的提升：[详解Java中的Lambda表达式](#)，[Java Lambda 表达式](#)

要求：将实现的代码、运行截图和自己的学习总结推送到GitHub仓库上，此处提交链接：

进阶挑战——应用流API

现在我们又成为了音乐厅的员工，这次我们需要再次改进我们的自动点歌机，以满足老板永无止境的要求。我们对点歌机进行了升级，现在的点歌机的**Song**类如下：

```
public class Song{
    private String title;
    private String artist;
    private int genre;
    private int year;
    private int timesPlayed;
    // 利用注解或者自己创建构造器和get方法
}
```

同时我们也给出了模拟代码：

```

class Songs {
    public List<Song> getSongs() {
        return List.of(
            new Song("$10", "Hitchhiker", "Electronic", 2016, 183),
            new Song("Havana", "Camila Cabello", "R&B", 2017, 324),
            new Song("Cassidy", "Grateful Dead", "Rock", 1972, 123),
            new Song("50 ways", "Paul Simon", "Soft Rock", 1975, 199),
            new Song("Hurt", "Nine Inch Nails", "Industrial Rock", 1995, 257),
            new Song("Silence", "Delerium", "Electronic", 1999, 134),
            new Song("Hurt", "Johnny Cash", "Soft Rock", 2002, 392),
            new Song("Watercolour", "Pendulum", "Electronic", 2010, 155),
            new Song("The Outsider", "A Perfect Circle", "Alternative Rock", 2004,
312),
            new Song("With a Little Help from My Friends", "The Beatles", "Rock", 1967,
168),
            new Song("Come Together", "The Beatles", "Blues rock", 1968, 173),
            new Song("Come Together", "Ike & Tina Turner", "Rock", 1970, 165),
            new Song("With a Little Help from My Friends", "Joe Cocker", "Rock", 1968,
46),
            new Song("Immigrant Song", "Karen O", "Industrial Rock", 2011, 12),
            new Song("Breathe", "The Prodigy", "Electronic", 1996, 337),
            new Song("What's Going On", "Gaye", "R&B", 1971, 420),
            new Song("Hallucinate", "Dua Lipa", "Pop", 2020, 75),
            new Song("Walk Me Home", "P!nk", "Pop", 2019, 459),
            new Song("I am not a woman, I'm a god", "Halsey", "Alternative Rock", 2021,
384),
            new Song("Pasos de cero", "Pablo Alborán", "Latin", 2014, 117),
            new Song("Smooth", "Santana", "Latin", 1999, 244),
            new Song("Immigrant song", "Led Zeppelin", "Rock", 1970, 484));
    }
}

```

你的任务：

1. 注意到更新后的歌曲包含歌曲的流派（**genre**），你需要找出所有包含“**rock**”流派的歌曲。

tip: 你需要过滤（**filter**）数据，只保留某一个流派的**Song**，你可以把目标歌曲收集（**collect**）到一个新的**List**中。

- 答：代码如下：

```

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;
import java.util.stream.Collectors;
import java.util.stream.Stream;

@Data
@AllArgsConstructor
@NoArgsConstructor
public class Song{
    private String title;
    private String artist;
    private String genre;
}

```

```

private int year;
private int timesPlayed;
// 利用注解或者自己创建构造器和get方法

public static void main (String[] args){
    Stream<Song> song = Songs.getSongs().stream();
    List<Song> song_ = song.filter(song1 ->
song1.genre.equals("Rock")).toList();
    song_.forEach(System.out::println);
}

}

class Songs {
    public static List<Song> getSongs() {
        return List.of(
            new Song("$10", "Hitchhiker", "Electronic", 2016, 183),
            new Song("Havana", "Camila Cabello", "R&B", 2017, 324),
            new Song("Cassidy", "Grateful Dead", "Rock", 1972, 123),
            new Song("50 ways", "Paul Simon", "Soft Rock", 1975, 199),
            new Song("Hurt", "Nine Inch Nails", "Industrial Rock", 1995,
257),
            new Song("Silence", "DeLerium", "Electronic", 1999, 134),
            new Song("Hurt", "Johnny Cash", "Soft Rock", 2002, 392),
            new Song("Watercolour", "Pendulum", "Electronic", 2010,
155),
            new Song("The Outsider", "A Perfect Circle", "Alternative
Rock", 2004, 312),
            new Song("With a Little Help from My Friends", "The
Beatles", "Rock", 1967, 168),
            new Song("Come Together", "The Beatles", "Blues rock", 1968,
173),
            new Song("Come Together", "Ike & Tina Turner", "Rock", 1970,
165),
            new Song("With a Little Help from My Friends", "Joe Cocker",
"Rock", 1968, 46),
            new Song("Immigrant Song", "Karen O", "Industrial Rock",
2011, 12),
            new Song("Breathe", "The Prodigy", "Electronic", 1996, 337),
            new Song("What's Going On", "Gaye", "R&B", 1971, 420),
            new Song("Hallucinate", "Dua Lipa", "Pop", 2020, 75),
            new Song("Walk Me Home", "P!nk", "Pop", 2019, 459),
            new Song("I am not a woman, I'm a god", "Halsey",
"Alternative Rock", 2021, 384),
            new Song("Pasos de cero", "Pablo Alborán", "Latin", 2014,
117),
            new Song("Smooth", "Santana", "Latin", 1999, 244),
            new Song("Immigrant song", "Led Zeppelin", "Rock", 1970,
484));
    }
}

```

```
D:\Java\jdk-17\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=52284:D:\IntelliJ IDEA 2024.2.3\bin" -Dfile.encoding=UTF-8
Song(title=Cassidy, artist=Grateful Dead, genre=Rock, year=1972, timesPlayed=123)
Song(title=With a Little Help from My Friends, artist=The Beatles, genre=Rock, year=1967, timesPlayed=168)
Song(title=Come Together, artist=Ike & Tina Turner, genre=Rock, year=1970, timesPlayed=165)
Song(title=With a Little Help from My Friends, artist=Joe Cocker, genre=Rock, year=1968, timesPlayed=46)
Song(title=Immigrant song, artist=Led Zeppelin, genre=Rock, year=1970, timesPlayed=484)
|
Process finished with exit code 0
```

-

1. 列出所有的歌曲流派。

tip: `map`方法接受一个`Function`，可以接受一个类型的对象并返回不同的对象，这意味着`Lambda`表达式可能会派上用场。

- **答：**我重写了`hashCode`和`equals`方法，再用`distinct`方法选出所有`genre`不一样的`song`放进`list`中，再用`lambda`表达式输出`genre`，就达到了题目所要求的目的。
- **其它想法：**也可以先将`genre`提取出来，再用`Stream`筛选，这样似乎还会更简单一些？

- ```
import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

import java.util.List;
import java.util.Objects;
import java.util.stream.Collectors;
import java.util.stream.Stream;

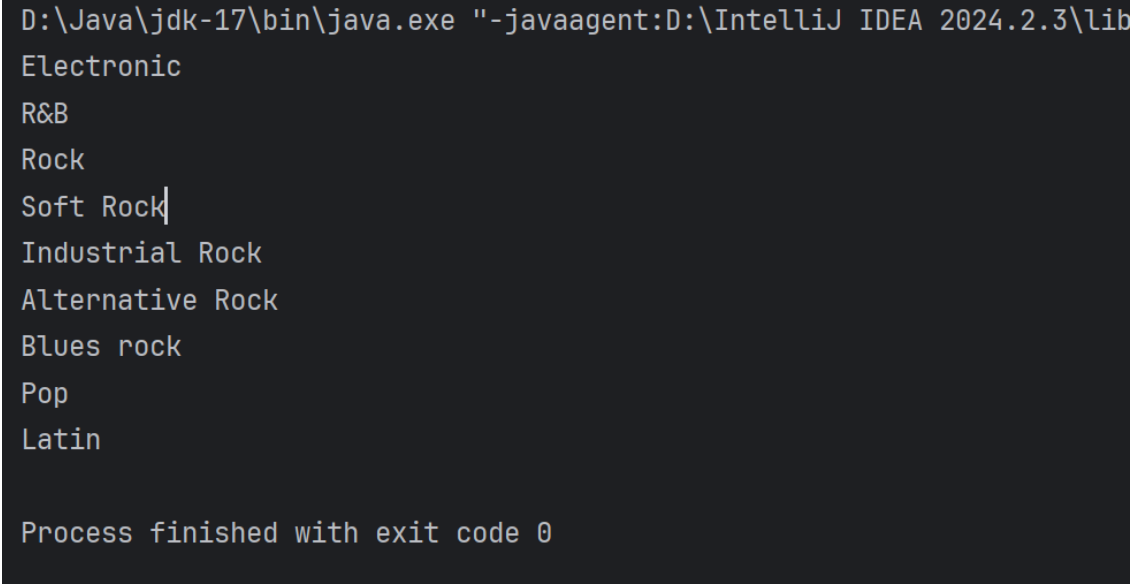
@Data
@AllArgsConstructor
@NoArgsConstructor
public class Song{
 private String title;
 private String artist;
 private String genre;
 private int year;
 private int timesPlayed;
 // 利用注解或者自己创建构造器和get方法

 @Override
 public int hashCode() {return Objects.hashCode(getGenre());}

 @Override
 public boolean equals(Object o) {
 if (this == o) return true;
 if (o == null || getClass() != o.getClass()) return false;
 Song song = (Song) o;
 if (!Objects.equals(genre, song.genre)) return false;
 return true;
 }

 public static void main (String[] args){
 Stream<Song> song = Songs.getSongs().stream();
 //List<Song> song_ = song.filter(song1 ->
 song1.genre.equals("Rock")).toList();
 }
}
```

```
List<Song> song_ = song.distinct().toList();
song_.forEach(song1 -> System.out.println(song1.genre));
}
}
```

- D:\Java\jdk-17\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2024.2.3\lib  
Electronic  
R&B  
Rock  
Soft Rock  
Industrial Rock  
Alternative Rock  
Blues rock  
Pop  
Latin  
  
Process finished with exit code 0

要求：将实现的代码、运行截图、自己学习过程中的笔记和思考以pdf格式推送到GitHub仓库（在笔记中写出对题目思考部分的学习和想法可以加分），此处提交链接：

---

## task 2 串行化

---

对象有行为和状态。行为在类中，但状态在各个对象中，那么保存一个对象的状态时会发生什么呢？

事实上，如何保存Java程序的状态，一般有以下两种选择：

1. 如果你的数据只由生成它的Java程序使用：你可以选择串行化并保存到某个文件当中，这样当你再需要用这个状态的时候，又可以读取串行化对象，把它变成一个可以使用的对象。
2. 如果你的数据由其他程序使用：你必须遵循其他程序的约定，写一个纯文本文件以方便读取。

**任务：**结合之前学习的流的知识，了解一下`InputStream/OutputStream`类，这里有教程可供参考：[Java 流\(Stream\)](#)、[文件\(File\)](#)和[IO](#)

你需要做的是：尝试将上文的歌曲作为串行化对象写入文件，再通过逆串行化将对象的状态读取出来。

- **答：**可以使用对象流实现对于对象的输出和输入。
  - 一定要注意Song和Songs均要实现Serializable的接口

- ```
// 串行化输出
ObjectOutputStream oos = new ObjectOutputStream(
    new FileOutputStream("Songs.dat"));

oos.writeObject(Songs.getSongs());
System.out.println("串行化输出成功~");

// 反串行化读取
ObjectInputStream ois = new ObjectInputStream(
    new FileInputStream("Songs.dat"));

List<Song> song_ = (List<Song>)ois.readObject();
song_.forEach(System.out::println);
```

- ```
D:\Java\jdk-17\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2024.2.3\lib\idea_rt.jar=54324:D:\IntelliJ IDEA 2024.2.3\bin"
串行化输出成功~
Song(title=$10, artist=Hitchhiker, genre=Electronic, year=2016, timesPlayed=183)
Song(title=Havana, artist=Camila Cabello, genre=R&B, year=2017, timesPlayed=324)
Song(title=Cassidy, artist=Grateful Dead, genre=Rock, year=1972, timesPlayed=123)
Song(title=50 ways, artist=Paul Simon, genre=Soft Rock, year=1975, timesPlayed=199)
Song(title=Hurt, artist=Nine Inch Nails, genre=Industrial Rock, year=1995, timesPlayed=257)
Song(title=Silence, artist=Delerium, genre=Electronic, year=1999, timesPlayed=134)
Song(title=Hurt, artist=Johnny Cash, genre=Soft Rock, year=2002, timesPlayed=392)
Song(title=Watercolour, artist=Pendulum, genre=Electronic, year=2010, timesPlayed=155)
Song(title=The Outsider, artist=A Perfect Circle, genre=Alternative Rock, year=2004, timesPlayed=312)
Song(title=With a Little Help from My Friends, artist=The Beatles, genre=Rock, year=1967, timesPlayed=168)
Song(title=Come Together, artist=The Beatles, genre=Blues rock, year=1968, timesPlayed=173)
Song(title=Come Together, artist=Ike & Tina Turner, genre=Rock, year=1970, timesPlayed=165)
Song(title=With a Little Help from My Friends, artist=Joe Cocker, genre=Rock, year=1968, timesPlayed=46)
Song(title=Immigrant Song, artist=Karen O, genre=Industrial Rock, year=2011, timesPlayed=12)]
Song(title=Breathe, artist=The Prodigy, genre=Electronic, year=1996, timesPlayed=337)
Song(title=What's Going On, artist=Gaye, genre=R&B, year=1971, timesPlayed=420)
Song(title=Hallucinate, artist=Dua Lipa, genre=Pop, year=2020, timesPlayed=75)
Song(title=Walk Me Home, artist=P!nk, genre=Pop, year=2019, timesPlayed=459)
Song(title=I am not a woman, I'm a god, artist=Halsey, genre=Alternative Rock, year=2021, timesPlayed=384)
Song(title=Pasos de cero, artist=Pablo Alborán, genre=Latin, year=2014, timesPlayed=117)
Song(title=Smooth, artist=Santana, genre=Latin, year=1999, timesPlayed=244)
Song(title=Immigrant song, artist=Led Zeppelin, genre=Rock, year=1970, timesPlayed=484)

Process finished with exit code 0
```

**tip:** 了解一下Serializable接口，掌握它的功能。

要求：将实现的代码、运行截图、自己学习过程中的笔记和思考以pdf格式推送到GitHub仓库（在笔记中写出对题目思考部分的学习和想法可以加分），此处提交链接：

---

## 进阶挑战——文件I/O

---

写文本数据（实际上就是String）和写对象类似，只不过你要写一个String而不是一个对象，而且要用FileWriter而不是FileOutputStream：

```
//就像这样
class WriteAFile {
 public static void main(String[] args) {
 // 所有I/O代码都有可能抛出一个IOException
 try {
 FileWriter writer = new FileWriter("Foo.txt");// 如果文件“Foo.txt”不存在，则会
 自动创建这个文件
 writer.write("hello foo!");
 writer.close();
 } catch (IOException ex) {
 ex.printStackTrace();
 }
 }
}
```

### 任务：

请自行学习关于文件写入和读入的操作，实现自动点歌机中将Song类写入和读出文件中的功能，参考上文链接。

- 答：因为题目没有明说写入和读出的具体效果，默认为以输入输出歌曲名称及作者的字符串代表写入和读出歌曲。
- 代码如下：

```
•
 try (BufferedWriter bw = new BufferedWriter(
 new FileWriter("Songs.txt"));) {
 Songs.getSongs().forEach(song -> {
 try {
 bw.write(song.getTitle() +
 " --- " + song.getArtist());
 bw.newLine();
 } catch (IOException e) {
 throw new RuntimeException(e);
 }
 });

 bw.close();
 }

 BufferedReader br = new BufferedReader(
 new FileReader("Songs.txt"));
 String line;
 while((line = br.readLine()) != null) {
 System.out.println(line);
 }
 br.close();
```

•



```
D:\Java\jdk-17\bin\java.exe "-javaagent:D:\IntelliJ IDEA 2024.2.3\l
串行化输出成功~
$10 --- Hitchhiker
Havana --- Camila Cabello
Cassidy --- Grateful Dead
50 ways --- Paul Simon
Hurt --- Nine Inch Nails
Silence --- Delerium
Hurt --- Johnny Cash
Watercolour --- Pendulum
The Outsider --- A Perfect Circle
With a Little Help from My Friends --- The Beatles
Come Together --- The Beatles
Come Together --- Ike & Tina Turner
With a Little Help from My Friends --- Joe Cocker
Immigrant Song --- Karen O
Breathe --- The Prodigy
What's Going On --- Gaye
Hallucinate --- Dua Lipa
Walk Me Home --- P!nk
I am not a woman, I'm a god --- Halsey
Pasos de cero --- Pablo Alborán
Smooth --- Santana
Immigrant song --- Led Zeppelin

Process finished with exit code 0
```