2024090905014-林仪-Java-09

Task1

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

public class Main {
    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);
        limit.forEach(System.out::println);
    }
}
```

直接打印stream对象得的是它的类型和地址信息
通过foreach可以遍历每一个元素并输出

Lambda表达式

lam实际是在运行时创建的一个含唯一方法的接口的匿名内部类并直接重写方法，

```
((String s1,String s2)->{  //s1,s2为接口方法的参数
    System.out.println(s1);//可为多条语句
})
//可简化为
((s1,s2)->System.out.println(s1))


((String s1)->{  //s1,s2为接口方法的参数
    return s1;
})
//只含return这一条时可简化为
(s1->s1);
```

其中传入参数的类型不能只省略其中一个
((s1-s2)->s1-s2)实际上是一个重写接口方法的类

方法引用

方法引用符 **::** 用于引用对象的方法，构造器，类的实例方法
静态方法引用 (类名：： 静态方法) 如(Integer::parseInt()) == (s->Integer.parseInt())
特定类型的任意对象的实例方法引用 (类名::实例方法名)， 如 (String::length)
特定对象的实例方法引用 (对象引用::实例方法名) 如 (str::toUpperCase)， 这里 str 是一个字符对象
方法引用类似与 **Lambda**表达式，实质上也是一个实现接口的匿名内部类

应用流API

```
import java.util.List;
public class Main {
    public static void main(String[] args) {

        List<Song> songs = (new Songs()).getSongs();
        List<Song> list = songs.stream().filter(s ->(s.getGenre().equals("Rock")
)).toList();
        list.forEach(System.out::println);
        System.out.println();
       songs.stream().map(Song::getGenre).distinct().forEach(System.out::println);
    }
}
 class Song{
    private String title;
    private String artist;
    private String genre;
    private int year;
    private int timesPlayed;

     public Song(String title, String artist, String genre, int year, int
timesPlayed) {
         this.title = title;
         this.artist = artist;
         this.genre = genre;
         this.year = year;
         this.timesPlayed = timesPlayed;
     }
```

```
    public String getGenre() {return genre;}
    public String getTitle() {return title;}

    @Override
    public String toString() {
        return "Song{" +
                "title='" + title + '\'' +
                ", artist='" + artist + '\'' +
                ", genre='" + genre + '\'' +
                ", year=" + year +
                ", timesPlayed=" + timesPlayed +
                '}';
    }
    // 利用注解或者自己创建构造器和get方法
}
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));
    }
}
```

```
C:\Users\Administrator\.jdks\openjdk-23\bin\java.exe "-javaagent:D:\idea\Intel
Song{title='Cassidy', artist='Grateful Dead', genre='Rock', year=1972, timesPl
Song{title='With a Little Help from My Friends', artist='The Beatles', genre='
Song{title='Come Together', artist='Ike & Tina Turner', genre='Rock', year=197
Song{title='With a Little Help from My Friends', artist='Joe Cocker', genre='R
Song{title='Immigrant song', artist='Led Zeppelin', genre='Rock', year=1970, t

Electronic
R&B
Rock
Soft Rock
Industrial Rock
Alternative Rock
Blues rock
Pop
Latin
```

distinct实际通过equal方法来比较，要转成非song类才好比较

task 2 串行化

```
import java.io.*;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        String path ="d:\\haha.dat";
        List<Song> list = new Songs().getSongs();
        File file = new File(path);
        ObjectOutputStream oos=null;
        ObjectInputStream ois=null;
        try {
            oos = new ObjectOutputStream(new FileOutputStream(file));
            for (Song s :list) {
                oos.writeObject(s);
```

```
            }
            ois = new ObjectInputStream(new FileInputStream(file));
            Object obj=null;
            while ((obj=ois.readObject())!=null)
            System.out.println(obj);

        } catch (EOFException e){}
        catch (IOException | ClassNotFoundException e) {
           e.printStackTrace();
        } finally {
            try {
                assert oos != null;
                oos.close();
                assert ois != null;
                ois.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
}
 class Song implements Serializable {
    private String title;
    private String artist;
    private String genre;
    private int year;
    private int timesPlayed;

     public Song(String title, String artist, String genre, int year, int
timesPlayed) {
         this.title = title;
         this.artist = artist;
         this.genre = genre;
         this.year = year;
         this.timesPlayed = timesPlayed;
     }

     public String getGenre() {return genre;}
     public String getTitle() {return title;}
```

```
    @Override
    public String toString() {
        return "Song{" +
                "title='" + title + '\'' +
                ", artist='" + artist + '\'' +
                ", genre='" + genre + '\'' +
                ", year=" + year +
                ", timesPlayed=" + timesPlayed +
                '}';
    }
    // 利用注解或者自己创建构造器和get方法
}
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));
    }
}
```

```
Song{title='The Outsider', artist='A Perfect Circle', genre='Alternative Rock',
Song{title='With a Little Help from My Friends', artist='The Beatles', genre='R
Song{title='Come Together', artist='The Beatles', genre='Blues rock', year=1968
Song{title='Come Together', artist='Ike & Tina Turner', genre='Rock', year=1970
Song{title='With a Little Help from My Friends', artist='Joe Cocker', genre='Rc
Song{title='Immigrant Song', artist='Karen O', genre='Industrial Rock', year=20
Song{title='Breathe', artist='The Prodigy', genre='Electronic', year=1996, time
Song{title='What's Going On', artist='Gaye', genre='R&B', year=1971, timesPlaye
Song{title='Hallucinate', artist='Dua Lipa', genre='Pop', year=2020, timesPlaye
Song{title='Walk Me Home', artist='P!nk', genre='Pop', year=2019, timesPlayed=4
Song{title='I am not a woman, I'm a god', artist='Halsey', genre='Alternative R
Song{title='Pasos de cero', artist='Pablo Alborán', genre='Latin', year=2014, t
Song{title='Smooth', artist='Santana', genre='Latin', year=1999, timesPlayed=24
Song{title='Immigrant song', artist='Led Zeppelin', genre='Rock', year=1970, ti

Process finished with exit code 0
```

readline每次读取后会指向下一行，判断下一行是否为空时不能直接使用，readline读到文件末尾时会抛出EOF异常

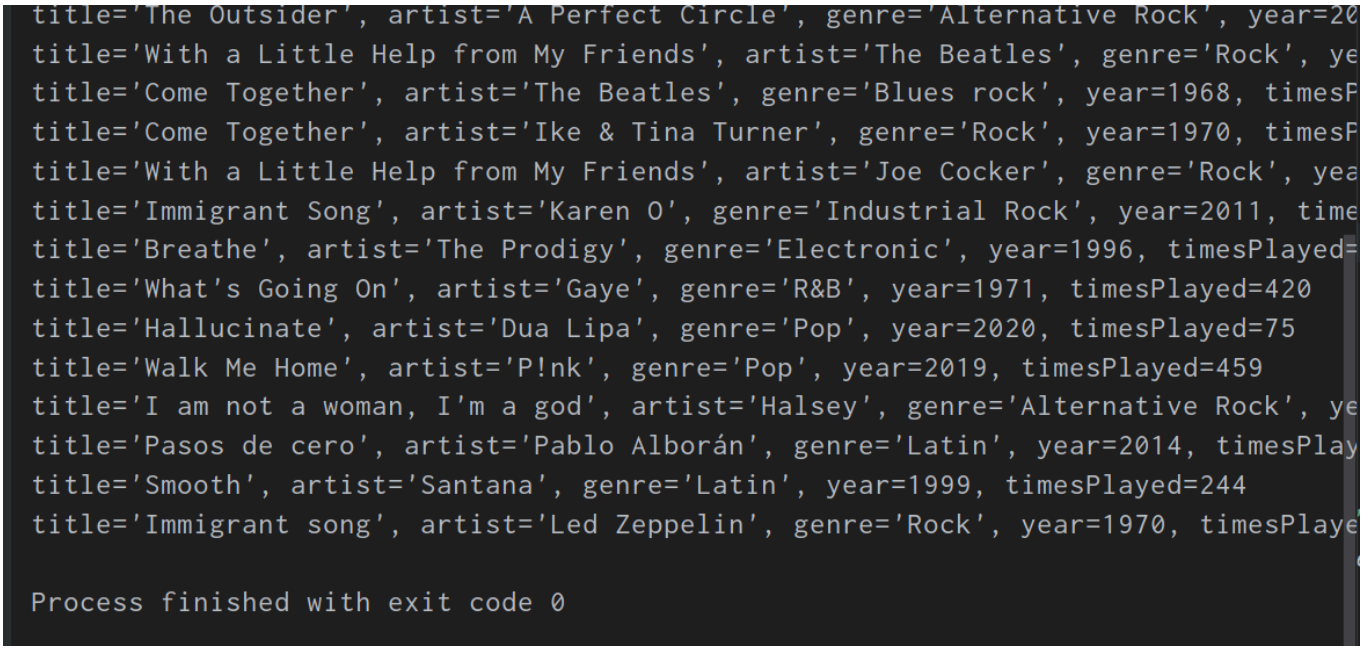
文件I/O

```
import java.io.*;
import java.util.ArrayList;
import java.util.List;
public class Main {
    public static void main(String[] args) {
        String path ="d:\\haha.txt";
        List<Song> list = new Songs().getSongs();
        List<Song> list2 = new ArrayList<>();
        File file = new File(path);
        BufferedWriter bw = null;
        BufferedReader br = null;
        try {
            bw = new BufferedWriter(new FileWriter(path));
            for (Song s :list) {
                bw.write(s.getall());
                bw.newLine();
                bw.flush();
```

```
            }
            br = new BufferedReader(new FileReader(file));
            String line;
            while ((line = br.readLine()) != null) {
                list2.add(new Song(line.split("/")[0],line.split("/")
[1],line.split("/")[2],
                        Integer.parseInt(line.split("/")
[3]),Integer.parseInt(line.split("/")[4])));
            }


        } catch (EOFException e){}
        catch (IOException e) {
           e.printStackTrace();
        } finally {
            try {
                assert bw != null;
                bw.close();
                assert br != null;
                br.close();
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
        list2.stream().forEach(System.out::println);
    }
}
 class Song implements Serializable {
    private String title;
    private String artist;
    private String genre;
    private int year;
    private int timesPlayed;

     public Song(String title, String artist, String genre, int year, int
timesPlayed) {
         this.title = title;
         this.artist = artist;
         this.genre = genre;
         this.year = year;
```

```
        this.timesPlayed = timesPlayed;
    }
    public String getGenre() {return genre;}
    public String getTitle() {return title;}
    public String getall() {return title+ "/" + artist + "/" + genre + "/" + year
+ "/" + timesPlayed;}

    @Override
    public String toString() {
        return
                "title='" + title + '\'' +
                ", artist='" + artist + '\'' +
                ", genre='" + genre + '\'' +
                ", year=" + year +
                ", timesPlayed=" + timesPlayed;
    }
    // 利用注解或者自己创建构造器和get方法
}
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));
    }
}
```



使用缓存流来输入输出文本内容
在Song中创建getall方法来取出文本内容
利用spilt方法来分割不同类型的数据以用读取的文本内容来创建Song对象