Ch17 練習

鄭安翔

ansel_cheng@hotmail.com

練習一 串流過濾操作

1. 開啟 StreamTest 專案
2. 檢視下列程式
   - com.example.Person 類別 / Gender 列舉 / City 列舉
   - com.example.ContactService 類別
   - com.example.StreamFilterTest 主類別
3. 修改 StreamFilterTest 類別
   - 寄送Email給所有成員：將集合轉為串流，使用forEach()寄送Email
   - 寄送Email 給小於30歲的成員：filter()過濾小於30歲的成員
   - 傳簡訊給南部成員：filter()過濾台南或高雄的成員
   - 傳簡訊給高雄男性成員：宣告Predicte過濾高雄男性成員
   - 使用鍊式呼叫Email給男性成員
   - 使用鍊式呼叫傳簡訊給台南的女性成員
   - 使用鍊式呼叫傳簡訊給台北25歲以上的女性成員
4. 測試、執行

Person類別/ Gender列舉/ City 列舉

Person.java

```java
package com.example;

import java.util.*;

public class Person {
    private String name;
    private City city;
    private int age;
    private Gender gender;
    private String email;
    private String phone;

    public Person(String name, City city, int age,
                  Gender gender, String email, String phone) {
        this.name = name;
        this.city = city;
        this.age = age;
        this.gender = gender;
        this.email = email;
        this.phone = phone;
    }

```

City.java

```java
package com.example;

public enum City {
    Taipei, Tainan, Kaohsiung
}
```

Gender.java

```java
package com.example;

public enum Gender {
    MALE, FEMALE
}
```

Person 類別

```
23      public String getName() {
26
27      public City getCity() {
30
31      public Gender getGender() {
34
35      public int getAge(){
38
39      public String getEmail() {
42
43      public String getPhone() {
46
47      @Override
48      public String toString(){
49          return String.format("%s(%d)%s", name, age, city);
50      }
51
52      public static List<Person> createList() {
53          List<Person> people = new ArrayList<>();
54          people.add(new Person("Sean", City.Taipei, 43, Gender.MALE, "sean@gmail.com", "02-2876-5432"));
55          people.add(new Person("Bob", City.Kaohsiung, 21, Gender.MALE, "bob@gmail.com", "07-6655-4433"));
56          people.add(new Person("Jane", City.Tainan, 35, Gender.FEMALE, "jane@gmail.com", "06-654-321"));
57          people.add(new Person("John", City.Taipei, 27, Gender.MALE, "john@gmail.com", "02-2345-6789"));
58          people.add(new Person("James", City.Tainan, 45, Gender.MALE, "james@gmail.com", "06-275-7575"));
59          people.add(new Person("Betty", City.Kaohsiung, 33, Gender.FEMALE, "betty@gmail.com", "07-9876-8787"));
60          people.add(new Person("Ivy", City.Taipei, 23, Gender.FEMALE, "ivy@gmail.com", "02-2777-1234"));
61          people.add(new Person("Nicole", City.Tainan, 42, Gender.FEMALE, "nicole@gmail.com", "06-666-4545"));
62          return people;
63      }
64
65  }
```

ContactService 類別

ContactService.java

```java
package com.example;

public class ContactService {
    public void sendEmail(Person p) {
        System.out.printf("Emailing: %s(%d) at %s%n", p.getName(), p.getAge(), p.getEmail());
    }

    public void sendMessage(Person p) {
        System.out.printf("Texting: %s(%d) at %s%n", p.getName(), p.getAge(), p.getPhone());
    }
}

```

StreamFilterTest類別

StreamFilterTest.java

```java
package com.example;

import java.util.List;

public class StreamFilterTest {

    public static void main(String[] args) {
        List<Person> personList = Person.createList();
        ContactService service = new ContactService();

        // 將集合轉為串流物件
        // 使用forEach()寄送Email給成員
        System.out.println("Email 給成員");


        //使用filter()過濾小於30歲的成員後,使用forEach()寄送Email
        System.out.println("Email 給小於30歲的成員");


        //傳簡訊給南部成員
        System.out.println("傳簡訊給南部成員");


        //宣告Predicte過濾高雄男性成員, 使用forEach()傳簡訊給過濾後成員
        System.out.println("傳簡訊給高雄男性成員");


        //使用鍊式呼叫Email給男性成員
        System.out.println("Email給男性成員");


        //使用鍊式呼叫傳簡訊給台南的女性成員
        System.out.println("傳簡訊給台南的女性成員");


        //使用鍊式呼叫傳簡訊給台北25歲以上的女性成員
        System.out.println("傳簡訊給台北25歲以上的男性成員");


    }

}
```

StreamFilterTest 類別

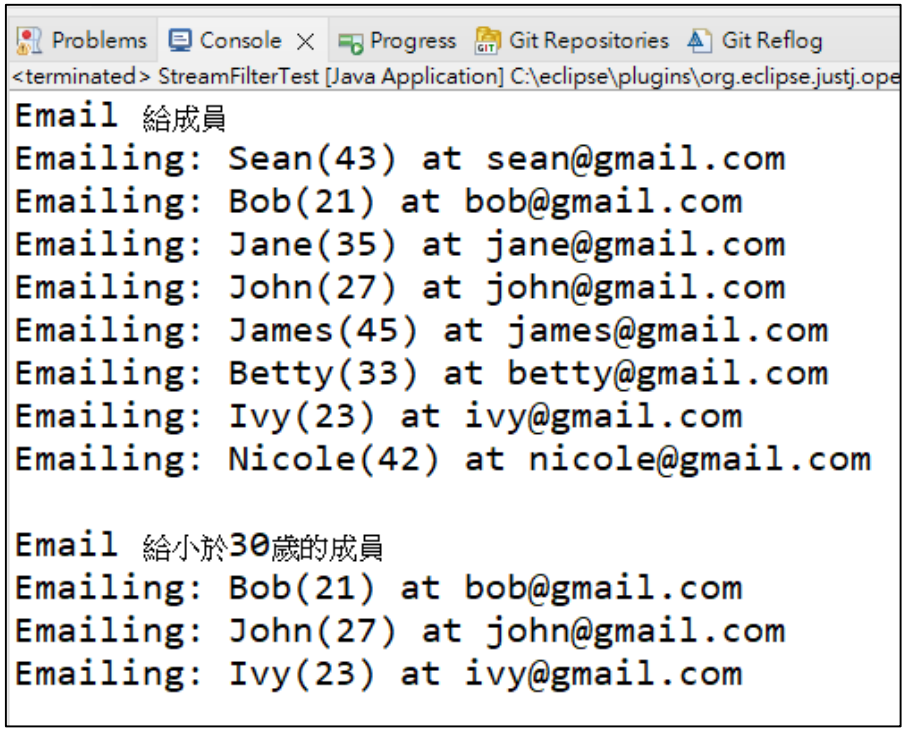
StreamFilterTest.java

```java
package com.example;

import java.util.List;

public class StreamFilterTest {

    public static void main(String[] args) {
        List<Person> personList = Person.createList();
        ContactService service = new ContactService();

        // 將集合轉為串流物件
        // 使用forEach()寄送Email給成員
        System.out.println("Email 給成員");
        personList.stream().forEach(p -> service.sendEmail(p));

        //使用filter()過濾小於30歲的成員後,使用forEach()寄送Email
        System.out.println("Email 給小於30歲的成員");
        personList.stream().filter(p->p.getAge()<30).forEach(p->service.sendEmail(p));

        //傳簡訊給南部成員
        System.out.println("傳簡訊給南部成員");
        personList.stream().filter(p->p.getCity()!=City.Taipei).forEach(p->service.sendMessage(p));

        //宣告Predicte過濾高雄男性成員, 使用forEach()傳簡訊給過濾後成員
        System.out.println("傳簡訊給高雄男性成員");
        personList.stream()
                  .filter(p->p.getGender()==Gender.MALE)
                  .filter(p->p.getCity()==City.Kaohsiung)
                  .forEach(p->service.sendMessage(p));
```

StreamFilterTest 類別

```
32
33          //使用鍊式呼叫Email給男性成員
34          System.out.println("Email給男性成員");
35          personList.stream().filter(p->p.getGender()==Gender.MALE)
36                    .forEach(p->service.sendEmail(p));
37
38          //使用鍊式呼叫傳簡訊給台南的女性成員
39          System.out.println("傳簡訊給台南的女性成員");
40          personList.stream().filter(p->p.getCity()==City.Tainan)
41                    .filter(p->p.getGender()==Gender.FEMALE)
42                    .forEach(p->service.sendMessage(p));
43
44          //使用鍊式呼叫傳簡訊給台北25歲以上的女性成員
45          System.out.println("傳簡訊給台北25歲以上的男性成員");
46          personList.stream()
47                    .filter(p->p.getCity()==City.Taipei)
48                    .filter(p->p.getAge()>25)
49                    .filter(p->p.getGender()==Gender.MALE)
50                    .forEach(p->service.sendMessage(p));
51
52      }
53
54  }
```

測試執行




```
傳簡訊給南部成員
Texting: Bob(21) at 07-6655-4433
Texting: Jane(35) at 06-654-321
Texting: James(45) at 06-275-7575
Texting: Betty(33) at 07-9876-8787
Texting: Nicole(42) at 06-666-4545

傳簡訊給高雄男性成員
Texting: Bob(21) at 07-6655-4433

Email給男性成員
Emailing: Sean(43) at sean@gmail.com
Emailing: Bob(21) at bob@gmail.com
Emailing: John(27) at john@gmail.com
Emailing: James(45) at james@gmail.com

傳簡訊給台南的女性成員
Texting: Jane(35) at 06-654-321
Texting: Nicole(42) at 06-666-4545

傳簡訊給台北25歲以上的男性成員
Texting: Sean(43) at 02-2876-5432
Texting: John(27) at 02-2345-6789
```

練習二 串流運算

1. 開啟 StreamTest 專案
2. 檢視下列程式
   - com.example.Gender / City 列舉
   - com.example. StreamOperationTest 主類別
3. 修改com.example.Person 類別
   - 實作Comparable，compareTo(Person p)方法，以name由小到大排序
   - 撰寫compareAgeTo(Person p) ，以age由大到小排序
   - 撰寫compareCityTo(Person p) ，以city由小到大排序
4. 修改 StreamOperationTest 類別，使用串流運算
   - 顯示台北成員的性別稱謂, 以map()方法將性別轉換為先生或小姐
   - 問候所有台南的成員(你好, Sean 先生), 以peek()方法在性別稱謂前列印問候語
   - 使用findFirst()方法取得第一個住台南的年齡小於35歲的女性
   - 南部成員個數

3. 修改 StreamOperationTest 類別，使用串流運算
   - 取得年紀最大的成員
   - 取得年紀最小的成員
   - 取得成員年紀總和
   - 取得成員年紀平均
   - 女性成員排序
   - 男性成員依城市排序
   - 所有成員依年紀反向排序
   - 所有成員依城市->年紀兩階段排序
   - 以收集器取得將女性成員排序後轉為新序列
   - 以收集器取得所有台北成員的電話序列
   - 收集器產生計算台南成員平均年紀
   - 收集器取得所有台北成員email字串用,隔開
   - 成員依城市分組
   - 成員依城市計數
   - 成員依年齡分組
4. 測試、執行

Person.java

```
1  package com.example;
2
3  import java.util.*;
4
5  public class Person implements Comparable<Person>{
6      private String name;
7      private City city;
8      private int age;
9      private Gender gender;
10     private String email;
11     private String phone;
12
13     public Person(String name, City city, int age,
22
23     public String getName() {
26
27     public City getCity() {
30
31     public Gender getGender() {
34
35     public int getAge(){
38
39     public String getEmail() {
42
43     public String getPhone() {
46
48     public String toString(){
51
52     public static List<Person> createList() {
64
```

```
65     @Override
66     public int compareTo(Person p) {
67         return name.compareTo(p.getName());
68     }
69
70     public int compareAgeTo(Person p){
71         return p.getAge() - age;
72     }
73
74     public int compareCityTo(Person p){
75         return this.city.compareTo(p.city);
76     }
77
78 }
79
```

Gender 列舉 / City 列舉

Gender.java

```java
package com.example;

public enum Gender {
    MALE("先生"), FEMALE("女士");

    String prefix;

    private Gender(String prefix) {
        this.prefix = prefix;
    }

    public String getPrefix() {
        return prefix;
    }

}
```

City.java

```java
package com.example;

public enum City {
    Taipei, Tainan, Kaohsiung
}
```

StreamOperationTest 類別

StreamOperationTest.java

```java
package com.example;

import java.util.*;

public class StreamOperationTest {

    public static void main(String[] args) {
        List<Person> personList = Person.createList();

        // 顯示台北成員的性別稱謂, 以map()方法將性別轉換為先生或小姐

        // 問候所有台南的成員(你好, Sean 先生), 以peek()方法在性別稱謂前列印問候語

        // 使用findFirst()方法取得第一個住台南的年齡小於35歲的女性,

        // 南部成員個數

        //取得年紀最大的成員

        //取得年紀最小的成員

        //取得成員年紀總和

        //取得成員年紀平均

        //女性成員排序

        //男性成員依城市排序

        //所有成員依年紀反向排序

        //所有成員依城市->年紀兩階段排序

        //以收集器取得將女性成員排序後轉為新序列

        //以收集器取得所有台北成員的電話序列

        //收集器產生計算台南成員平均年紀

        //收集器取得所有台北成員email字串用,隔開

        //成員依城市分組

        //成員依城市計數

        //成員依年齡分組

    }
}
```

StreamOperationTest 類別

*StreamOperationTest.java

```java
package com.example;

import java.util.*;
import java.util.stream.Collectors;

public class StreamOperationTest {

    public static void main(String[] args) {
        List<Person> personList = Person.createList();

        // 顯示台北成員的性別稱謂, 以map()方法將性別轉換為先生或小姐
        System.out.println("顯示台北成員的性別稱謂");
        personList.stream().filter(p->p.getCity()==City.Taipei)
                .map(p -> p.getGender().getPrefix())
                .forEach(System.out::println);

        // 問候所有台南的成員(你好, Sean 先生), 以peek()方法在性別稱謂前列印問候語
        System.out.println("\n問候所有台南的成員");
        personList.stream().filter(p->p.getCity()==City.Tainan)
                .peek(p -> System.out.printf("您好,%s", p.getName()))
                .map(p -> p.getGender().getPrefix())
                .forEach(System.out::println);

        // 使用findFirst()方法取得第一個住台南的年齡小於35歲的女性,
        System.out.println("\n取得住台南的年齡小於35歲的女性");

        Optional<Person> result = personList.stream().filter(p->p.getCity()==City.Tainan)
                .filter(p->p.getGender()==Gender.FEMALE).findFirst();
        if(result.isPresent()){
            System.out.println(result.get());
        } else {
            System.out.println("無符合條件成員");
        }

```

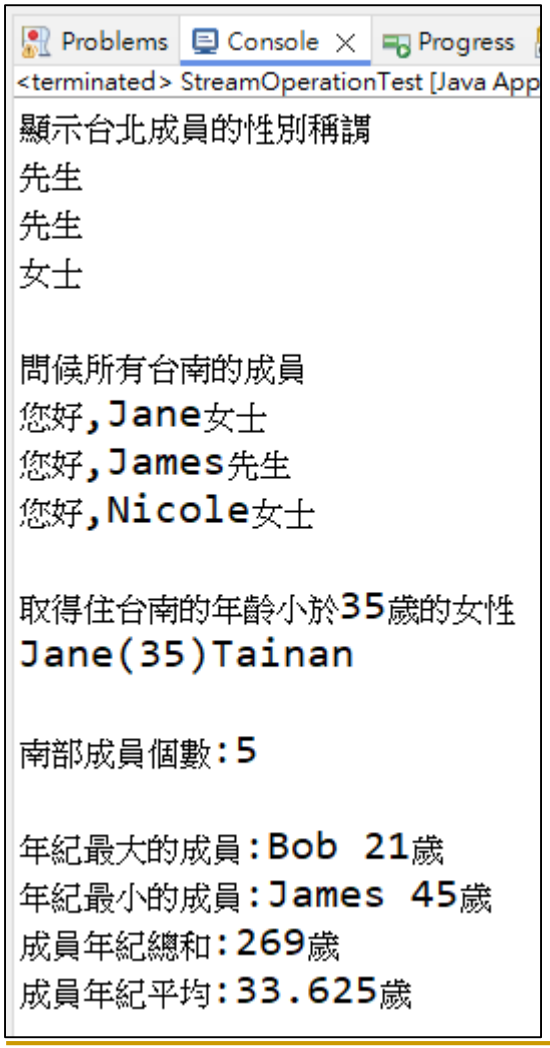
StreamOperationTest 類別

```
35          // 南部成員個數
36          long southCount = personList.stream().filter(p->p.getCity()==City.Tainan||p.getCity()==City.Kaohsiung).count();
37          System.out.println("\n南部成員個數:"+southCount);
38
39          //取得年紀最大的成員
40          Person oldest = personList.stream().max(Person::compareAgeTo).get();
41          System.out.println("\n年紀最大的成員:"+oldest.getName()+" "+oldest.getAge()+"歲");
42          //取得年紀最小的成員
43          Person youngest = personList.stream().min(Person::compareAgeTo).get();
44          System.out.println("年紀最小的成員:"+youngest.getName()+" "+youngest.getAge()+"歲");
45          //取得成員年紀總和
46          int ageSum = personList.stream().mapToInt(p->p.getAge()).sum();
47          System.out.println("成員年紀總和:"+ageSum+"歲");
48          //取得成員年紀平均
49          double ageAvg = personList.stream().mapToInt(p->p.getAge()).average().getAsDouble();
50          System.out.println("成員年紀平均:"+ageAvg+"歲");
51
52          //女性成員排序
53          System.out.println("\n女性成員排序");
54          personList.stream().filter(p->p.getGender()==Gender.FEMALE).sorted()
55                  .forEach(p->System.out.printf("%s(%d)%s%n", p.getName(), p.getAge(), p.getCity()));
56          //男性成員依城市排序
57          System.out.println("\n男性成員依城市排序");
58          personList.stream().filter(p->p.getGender()==Gender.MALE).sorted((p1,p2)->p1.compareCityTo(p2))
59                  .forEach(p->System.out.printf("%s(%d)%s%n", p.getName(), p.getAge(), p.getCity()));
60          //所有成員依年紀反向排序
61          System.out.println("\n所有成員依年紀反向排序");
62          personList.stream().sorted(Comparator.comparing(Person::getAge).reversed())
63                  .forEach(p->System.out.printf("%s(%d)%s%n", p.getName(), p.getAge(), p.getCity()));
64          //所有成員依城市->年紀兩階段排序
65          System.out.println("\n所有成員依城市後年紀排序");
66          personList.stream().sorted(Comparator.comparing(Person::getCity).thenComparing(Person::getAge))
67                  .forEach(p->System.out.printf("%s(%d)%s%n", p.getName(), p.getAge(), p.getCity()));
68
```

StreamOperationTest 類別

```
69          //以收集器取得將女性成員排序後轉為新序列
70          List<Person> newList = personList.stream().filter(p->p.getGender()==Gender.FEMALE).sorted().collect(Collectors.toList());
71          System.out.println("\n女性成員排序之新序列:"+newList);
72          //以收集器取得所有台北成員的電話序列
73          List<String> phoneList = personList.stream().filter(p->p.getCity()==City.Taipei).map(Person::getPhone).collect(Collectors.toList());
74          System.out.println("\n台北成員電話序列:"+phoneList);
75          //收集器產生計算台南成員平均年紀
76          double avgAge = personList.stream().filter(p->p.getCity()==City.Tainan).collect(Collectors.averagingDouble(Person::getAge));
77          System.out.println("\n台南成員平均年紀:"+avgAge);
78          //收集器取得所有台北成員email字串用,隔開
79          String emailStr = personList.stream().filter(p->p.getGender()==Gender.FEMALE).map(Person::getEmail).collect(Collectors.joining(", "));
80          System.out.println("\n女性成員email:"+emailStr);
81
82          //成員依城市分組
83          Map<City, List<Person>> cityMap = personList.stream().collect(Collectors.groupingBy(Person::getCity));
84          System.out.println("\n成員依城市分組:");
85          cityMap.forEach((k,v)->System.out.println(k+":"+v));
86
87          //成員依城市計數
88          Map<City, Long> cityCount = personList.stream().collect(Collectors.groupingBy(p->p.getCity(), Collectors.counting()));
89          System.out.println("\n成員依城市計數:");
90          cityCount.forEach((k,v)->System.out.println(k+":"+v+"人"));
91          //成員依年齡分組
92          Map<Boolean, List<Person>> ageMap = personList.stream().collect(Collectors.partitioningBy(p->p.getAge()>30));
93          System.out.println("\n成員依年齡分組:");
94          ageMap.forEach((k,v)->System.out.println((k?"大於":"小於")+"30歲:"+v));
95
96      }
97  }
```

測試執行



女性成員排序
Betty(33)Kaohsiung
Ivy(23)Taipei
Jane(35)Tainan
Nicole(42)Tainan

男性成員依城市排序
Sean(43)Taipei
John(27)Taipei
James(45)Tainan
Bob(21)Kaohsiung

所有成員依年紀反向排序
James(45)Tainan
Sean(43)Taipei
Nicole(42)Tainan
Jane(35)Tainan
Betty(33)Kaohsiung
John(27)Taipei
Ivy(23)Taipei
Bob(21)Kaohsiung

所有成員依城市後年紀排序
Ivy(23)Taipei
John(27)Taipei
Sean(43)Taipei
Jane(35)Tainan
Nicole(42)Tainan
James(45)Tainan
Bob(21)Kaohsiung
Betty(33)Kaohsiung

測試執行

```
女性成員排序之新序列:[Betty(33)Kaohsiung, Ivy(23)Taipei, Jane(35)Tainan, Nicole(42)Tainan]

台北成員電話序列:[02-2876-5432, 02-2345-6789, 02-2777-1234]

台南成員平均年紀:40.666666666666664

女性成員email:jane@gmail.com, betty@gmail.com, ivy@gmail.com, nicole@gmail.com

成員依城市分組:
Kaohsiung:[Bob(21)Kaohsiung, Betty(33)Kaohsiung]
Taipei:[Sean(43)Taipei, John(27)Taipei, Ivy(23)Taipei]
Tainan:[Jane(35)Tainan, James(45)Tainan, Nicole(42)Tainan]

成員依城市計數:
Kaohsiung:2人
Taipei:3人
Tainan:3人

成員依年齡分組:
小於30歲:[Bob(21)Kaohsiung, John(27)Taipei, Ivy(23)Taipei]
大於30歲:[Sean(43)Taipei, Jane(35)Tainan, James(45)Tainan, Betty(33)Kaohsiung, Nicole(42)Tainan]
```