

```
1 /*
2 * To change this license header, choose License Headers in Project Properties.
3 * To change this template file, choose Tools | Templates
4 * and open the template in the editor.
5 */
6 package ex11;
7
8 import ex11.Person.Gender;
9 import java.util.ArrayList;
10 import java.util.List;
11 import java.util.Map;
12 import java.util.Set;
13 import java.util.SortedSet;
14 import java.util.TreeSet;
15 import java.util.stream.Collectors;
16
17 /**
18 *
19 * @author cwtng
20 */
21 public class CollectorTest {
22
23     public static void main(String[] args) {
24
25         List<Person> persons = Person.createShortList();
26
27         List<String> names = persons.stream()
28             .map(Person::getName)
29             .collect(ArrayList::new,
30                 ArrayList::add,
31                 ArrayList::addAll);
32
33         names.forEach(name -> System.out.println(name));
34         System.out.println("");
35
36         // using Collectors below
37         List<String> nameList = persons.stream()
38             .map(Person::getName)
39             .collect(Collectors.toList());
40
41         nameList.forEach(name -> System.out.println(name));
42         System.out.println("");
43
44         // To obtain a set of unique names (removing duplicates)
45         Set<String> nameSet = persons.stream()
46             .map(Person::getName)
47             .collect(Collectors.toSet());
48
49         nameSet.forEach(name -> System.out.println(name));
50         System.out.println("");
51
52         // To obtain a set of unique names sorted in natural order
53         SortedSet<String> uniqueSortedNames = persons.stream()
54             .map(Person::getName)
55             .collect(Collectors.toCollection(TreeSet::new));
56
57         uniqueSortedNames.forEach(name -> System.out.println(name));
58         System.out.println("");
59     }
```

```
60 // To obtain a map of email to name
61 Map<String, String> email2Name = persons.stream()
62     .collect(Collectors.toMap(
63         Person::getEmail,
64         Person::getName));
65 // keyMapper
66 // valueMapper
67
68 email2Name.forEach((email, name) -> System.out.println(email + " : " +
69     name));
70 System.out.println("");
71
72 // To join the name of each person with a comma separator
73 String joinedNames = persons.stream().map(Person::getName)
74     .collect(Collectors.joining(", "));
75 System.out.println(joinedNames);
76 System.out.println("");
77
78 // To group people by their gender
79 Map<Gender, List<Person>> peopleByGender = persons.stream()
80     .collect(Collectors.groupingBy(Person::getGender));
81 System.out.println(peopleByGender);
82 System.out.println("");
83
84 // To count people by their gender
85 Map<Gender, Long> countPeopleByGender = persons.stream()
86     .collect(Collectors.groupingBy(Person::getGender,
87         Collectors.counting()));
88 System.out.println(countPeopleByGender);
89 System.out.println("");
90
91 // To group people by their gender, then by surname
92 Map<Gender, Map<String, List<Person>>> peopleByGenderBySurname =
93     persons.stream()
94         .collect(Collectors.groupingBy(Person::getGender,
95             Collectors.groupingBy(Person::getSurname)));
96 System.out.println(peopleByGenderBySurname);
97 System.out.println("");
98
99 // To partition people by their age
100 Map<Boolean, List<Person>> peopleByAge = persons.stream()
101     .collect(Collectors.groupingBy(person -> person.getAge() > 50));
102 System.out.println(peopleByAge);
103 System.out.println("");
```

```
1 package ex11;
2
3 import java.util.Collections;
4 import java.util.List;
5 import java.util.Comparator;
6
7 /**
8  * @author Mikew
9  */
10 public class ComparatorTest {
11
12     public static void main(String[] args) {
13
14         List<Person> personList = Person.createShortList();
15
16         // Sort with Inner Class
17         Collections.sort(personList, new Comparator<Person>() {
18             public int compare(Person p1, Person p2) {
19                 return p1.getSurName().compareTo(p2.getSurName());
20             }
21         });
22
23         System.out.println("=== Sorted Asc SurName ===");
24         for (Person p : personList) {
25             p.printName();
26         }
27
28         // Use Lambda instead
29         // Print Asc
30         System.out.println("=== Sorted Asc SurName ===");
31         Collections.sort(personList, (Person p1, Person p2) ->
32             p1.getSurName().compareTo(p2.getSurName()));
33
34         for (Person p : personList) {
35             p.printName();
36         }
37
38         // Note that the type of p1 and p2 are inferred
39         // Print Desc
40         System.out.println("=== Sorted Desc SurName ===");
41         Collections.sort(personList, (p1, p2) ->
42             p2.getSurName().compareTo(p1.getSurName()));
43
44         for (Person p : personList) {
45             p.printName();
46         }
47     }
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex11;
7
8 import static ex11.Person.Gender.MALE;
9 import java.util.List;
10 import java.util.Optional;
11
12 /**
13  *
14  * @author vanting
15  */
16 public class FindAndMatchStream {
17
18     public static void main(String[] args) {
19
20         List<Person> persons = Person.createShortList();
21
22         // Check if all persons in the list are male
23         boolean allMales = persons.stream().allMatch(person -> person.getGender() ==
24             MALE);
25
26         // Check if any person is at 25
27         boolean anyoneAt25 = persons.stream().anyMatch(person -> person.getAge() ==
28             25);
29
30         // Find the first male
31         Optional<Person> firstMale
32             = persons.stream()
33                 .filter(person -> person.getGender() == MALE)
34                 .findFirst();
35
36         System.out.println(allMales);
37         System.out.println(anyoneAt25);
38         System.out.println(firstMale);
39     }
40 }
41
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex11;
7
8 import java.util.ArrayList;
9 import java.util.List;
10 import java.util.function.Consumer;
11 import java.util.function.Function;
12 import java.util.function.Predicate;
13
14 /**
15  *
16  * @author vanting
17  */
18 public class FunctionUtil {
19
20     // Apply an action on each item in a list
21     public static <T> void forEach(List<T> list,
22                                   Consumer<? super T> action) {
23         for (T item : list) {
24             action.accept(item);
25         }
26     }
27
28     // Apply a filter to a list, returned the filtered list items
29     public static <T> List<T> filter(List<T> list,
30                                     Predicate<? super T> predicate) {
31         List<T> filteredList = new ArrayList<>();
32         for (T item : list) {
33             if (predicate.test(item)) {
34                 filteredList.add(item);
35             }
36         }
37         return filteredList;
38     }
39
40     // Map each item of type T in a list to a value of type R
41     public static <T, R> List<R> map(List<T> list,
42                                     Function<? super T, R> mapper) {
43         List<R> mappedList = new ArrayList<>();
44         for (T item : list) {
45             mappedList.add(mapper.apply(item));
46         }
47         return mappedList;
48     }
49
50     }
51 }
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex11;
7
8 import static ex11.Person.Gender.MALE;
9 import java.util.List;
10
11 /**
12  *
13  * @author vanting
14  */
15 public class FunctionUtilTest {
16
17     public static void main(String[] args) {
18
19         List<Person> list = Person.createShortList();
20
21         // Use forEach() method to print each person in the list
22         System.out.println("Original list of persons:");
23         FunctionUtil.forEach(list, p -> System.out.println(p));
24
25         // Filter only males
26         List<Person> maleList = FunctionUtil.filter(list, p -> p.getGender() ==
27 MALE);
28
29         System.out.println("\nMales only:");
30         FunctionUtil.forEach(maleList, p -> System.out.println(p));
31
32         // Map each person to his/her email
33         List<String> emailList = FunctionUtil.map(list,
34 p -> p.getEmail());
35
36         System.out.println("\nPersons mapped to their email:");
37         FunctionUtil.forEach(emailList, email -> System.out.println(email));
38
39         // Apply an action to each person in the male list
40         // Add one to each male's age
41         FunctionUtil.forEach(maleList,
42 p -> p.setAge(p.getAge() + 1));
43
44         System.out.println(
45             "\nMales only after adding 1 to age:");
46
47         FunctionUtil.forEach(maleList, p -> System.out.println(p));
48     }
49 }
50 }
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex11;
7 /**
8  *
9  * @author vanting
10 */
11 */
12 interface MyInterface {
13
14     /* This is a default method so we need not
15      * to implement this method in the implementation
16      * classes
17      */
18     default void newMethod() {
19         System.out.println("Newly added default method");
20     }
21
22     /* This is a static method. Static method in interface is
23      * similar to default method except that we cannot override
24      * them in the implementation classes.
25      * Similar to default methods, we need to implement these methods
26      * in implementation classes so we can safely add them to the
27      * existing interfaces.
28      */
29     static void anotherNewMethod() {
30         System.out.println("Newly added static method");
31     }
32
33     /* Already existing public and abstract method
34      * We must need to implement this method in
35      * implementation classes.
36      */
37     void existingMethod(String str);
38 }
39
40 public class Java8InterfaceExample implements MyInterface {
41
42     // implementing abstract method
43     public void existingMethod(String str) {
44         System.out.println("String is: " + str);
45     }
46
47     public static void main(String[] args) {
48         Java8InterfaceExample obj = new Java8InterfaceExample();
49         //calling the default method of interface
50         obj.newMethod();
51         //calling the static method of interface
52         MyInterface.anotherNewMethod();
53         //calling the abstract method of interface
54         obj.existingMethod("Java 8 is easy to learn");
55
56     }
57 }
58
```

```
1 package ex11;
2
3 import java.awt.BorderLayout;
4 import java.awt.event.ActionEvent;
5 import java.awt.event.ActionListener;
6 import javax.swing.JButton;
7 import javax.swing.JFrame;
8
9 /**
10 *
11 * @author Mikew
12 */
13 public class ListenerTest {
14
15     public static void main(String[] args) {
16
17         JButton testButton = new JButton("Test Button");
18         testButton.addActionListener(new ActionListener() {
19             @Override
20             public void actionPerformed(ActionEvent ae) {
21                 System.out.println("Click Detected by Anon Class");
22             }
23         });
24
25         testButton.addActionListener(e -> System.out.println("Click Detected by
26             Lambda Listener"));
27
28         // Swing stuff
29         JFrame frame = new JFrame("Listener Test");
30         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31         frame.add(testButton, BorderLayout.CENTER);
32         frame.pack();
33         frame.setVisible(true);
34     }
35 }
36
```

2021/12/10 上午2:48

Person.java

```
1 package ex11;
2
3 import java.util.ArrayList;
4 import java.util.List;
5
6 /**
7  * @author Mikew
8  */
9 public class Person {
10
11     enum Gender {
12         MALE, FEMALE
13     }
14
15     private String givenName;
16     private String surName;
17     private int age;
18     private Gender gender;
19     private String email;
20     private String phone;
21     private String address;
22
23     public static class Builder {
24
25         private String givenName = "";
26         private String surName = "";
27         private int age = 0;
28         private Gender gender = Gender.FEMALE;
29         private String email = "";
30         private String phone = "";
31         private String address = "";
32
33         public Person.Builder givenName(String givenName) {
34             this.givenName = givenName;
35             return this;
36         }
37
38         public Person.Builder surName(String surName) {
39             this.surName = surName;
40             return this;
41         }
42
43         public Person.Builder age(int val) {
44             age = val;
45             return this;
46         }
47
48         public Person.Builder gender(Gender val) {
49             gender = val;
50             return this;
51         }
52
53         public Person.Builder email(String val) {
54             email = val;
55             return this;
56         }
57
58         public Person.Builder phoneNumber(String val) {
59             phone = val;
60         }
61     }
62 }
```

localhost:4649/?mode=click

1/4

2021/12/10 上午2:48

Person.java

```
60         return this;
61     }
62
63     public Person.Builder address(String val) {
64         address = val;
65         return this;
66     }
67
68     public Person build() {
69         return new Person(this);
70     }
71
72     private Person() {
73         super();
74     }
75
76     private Person(Person.Builder builder) {
77         givenName = builder.givenName;
78         surName = builder.surName;
79         age = builder.age;
80         gender = builder.gender;
81         email = builder.email;
82         phone = builder.phone;
83         address = builder.address;
84     }
85
86     public String getName() {
87         return givenName + " " + surName;
88     }
89
90     public String getGivenName() {
91         return givenName;
92     }
93
94     public String getSurName() {
95         return surName;
96     }
97
98     public int getAge() {
99         return age;
100     }
101
102     public Gender getGender() {
103         return gender;
104     }
105
106     public String getEmail() {
107         return email;
108     }
109
110     public void setAge(int age) {
111         this.age = age;
112     }
113
114     public void print() {
115         System.out.println(
116             "\nName: " + givenName + " " + surName + "\n"
117             + "Age: " + age + "\n"
118         );
119     }
120 }
```

localhost:4649/?mode=click

2/4

2021/12/10 上午2:48	Person.java	2021/12/10 上午2:48	Person.java
120	+ "Gender: " + gender + "\n"	179	.surName("Johnson")
121	+ "eMail: " + eMail + "\n"	180	.age(45)
122	+ "Phone: " + phone + "\n"	181	.gender(Gender.MALE)
123	+ "Address: " + address + "\n"	182	.email("james.johnson@example.com")
124	);	183	.phoneNumber("333-456-1233")
125	}	184	.address("201 2nd St, New York, NY 12111")
126		185	.build()
127	public void printName() {	186	);
128	System.out.println(	187	people.add(
129	"Name: " + givenName + " " + surName);	188	new Person.Builder()
130	}	189	.givenName("Joe")
131		190	.surName("Baker")
132	@Override	191	.age(67)
133	public String toString() {	192	.gender(Gender.MALE)
134	return "Name: " + givenName + " " + surName + "\n" + "Age: " + age + "	193	.email("joebob.bailey@example.com")
135	Gender: " + gender + "\n" + eMail: " + eMail + "\n" + "Address: " + address + "\n";	194	.phoneNumber("112-111-1111")
136	}	195	.address("111 1st St, Town, CA 11111")
137		196	.build()
138	public static List<Person> createShortList() {	197	);
139	List<Person> people = new ArrayList<>();	198	
140		199	people.add(
141	new Person.Builder()	200	new Person.Builder()
142	.givenName("Bob")	201	.givenName("Phil")
143	.surName("Baker")	202	.surName("Smith")
144	.age(21)	203	.age(55)
145	.gender(Gender.MALE)	204	.gender(Gender.MALE)
146	.email("bob.baker@example.com")	205	.email("phil.smith@example.com")
147	.phoneNumber("201-121-4678")	206	.phoneNumber("222-33-1234")
148	.address("44 4th St, Smallville, KS 12333")	207	.address("22 2nd St, New Park, CO 222333")
149	.build()	208	.build()
150	);	209	
151		210	);
152	people.add(	211	
153	new Person.Builder()	212	people.add(
154	.givenName("Jane")	213	new Person.Builder()
155	.surName("Jones")	214	.givenName("Betty")
156	.age(25)	215	.surName("Jones")
157	.gender(Gender.FEMALE)	216	.age(85)
158	.email("jane.doe@example.com")	217	.gender(Gender.FEMALE)
159	.phoneNumber("202-123-4678")	218	.email("betty.jones@example.com")
160	.address("33 3rd St, Smallville, KS 12333")	219	.phoneNumber("211-33-1234")
161	.build()	220	.address("22 4th St, New Park, CO 222333")
162	);	221	.build()
163		222	);
164	people.add(	223	
165	new Person.Builder()	224	return people;
166	.givenName("John")	225	}
167	.surName("Baker")	226	
168	.age(25)	227	}
169	.gender(Gender.MALE)	228	
170	.email("john.doe@example.com")		
171	.phoneNumber("202-123-4678")		
172	.address("33 3rd St, Smallville, KS 12333")		
173	.build()		
174	);		
175			
176	people.add(		
177	new Person.Builder()		
178	.givenName("James")		
localhost:4649/?mode=click		localhost:4649/?mode=click	
3/4			4/4

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex11;
7
8 /**
9  *
10  * @author vanting
11  */
12 public class PrimeUtil {
13
14     private long lastPrime = 1L;
15
16     public long next() // instance method
17     {
18         lastPrime = next(lastPrime); // call static method
19         return lastPrime;
20     }
21
22     public static long next(long after) {
23         long counter = after;
24         while (!isPrime(++counter))
25             ;
26         return counter;
27     }
28
29     public static boolean isPrime(long num) {
30         if (num <= 1) {
31             return false;
32         }
33
34         if (num == 2) {
35             return true;
36         }
37
38         if (num % 2 == 0) {
39             return false;
40         }
41
42         long maxDivisor = (long) Math.sqrt(num);
43         for (int k = 3; k <= maxDivisor; k += 2) {
44             if (num % k == 0) {
45                 return false;
46             }
47         }
48
49         return true;
50     }
51
52 }
53
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex11;
7
8 import java.util.stream.Stream;
9
10 /**
11  *
12  * @author vanting
13  */
14 public class PrimeUtilTest {
15
16     public static void main(String[] args) {
17         // Print the first 5 prime numbers: 2, 3, 5, 7, 11
18         // Data type L for the seed value must be provided.
19         // Compiler uses it to infer the data type of stream elements.
20         // Use static method next(long)
21         Stream.iterate(2L, PrimeUtil::next) // n -> PrimeUtil.next(n)
22             .limit(5)
23             .forEach(System.out::println);
24
25         // Alternative design
26         Stream.iterate(2L, n -> n + 1)
27             .filter(PrimeUtil::isPrime) // n -> PrimeUtil.isPrime(n)
28             .limit(5)
29             .forEach(System.out::println);
30
31         // Design using the Stream.generate(Supplier) method,
32         // and skip the first 100 prime numbers
33         // use member method primeUtilObj.next()
34         Stream.generate(new PrimeUtil()).skip(100)
35             .limit(5)
36             .forEach(System.out::println);
37
38         /*
39         Supplier<Long> s = new Supplier()
40         {
41             PrimeUtil pu = new PrimeUtil();
42             public Long get()
43             {
44                 return pu.next();
45             }
46         };
47
48         Stream.generate(s)
49             .skip(100)
50             .limit(5)
51             .forEach(System.out::println);
52         */
53     }
54
55 }
56
57
```

```
1 package ex11;
2 /**
3  * @author Mikew
4  */
5 public class RunnableTest {
6
7     public static void main(String[] args) {
8
9         System.out.println("=== RunnableTest ===");
10
11         // Anonymous Runnable
12         Runnable r1 = new Runnable() {
13
14             @Override
15             public void run() {
16                 System.out.println("Hello world one!");
17             }
18         };
19
20         // Lambda Runnable
21         Runnable r2 = () -> System.out.println("Hello world two!");
22
23         // Run em!
24         r1.run();
25         r2.run();
26
27     }
28 }
29
30
```

```
1 /*
2  * To change this license header, choose License Headers in Project Properties.
3  * To change this template file, choose Tools | Templates
4  * and open the template in the editor.
5  */
6 package ex11;
7
8 import java.util.DoubleSummaryStatistics;
9 import java.util.List;
10
11 /**
12  *
13  * @author wanting
14  */
15 public class StatisticsTest {
16
17     public static void main(String[] args) {
18
19         // External iteration
20         DoubleSummaryStatistics stats = new DoubleSummaryStatistics();
21
22         List<Person> persons = Person.createShortList();
23         for (Person p : persons) {
24             stats.accept(p.getAge());
25         }
26
27         // Get statistics
28         System.out.println("Count: " + stats.getCount());
29         System.out.println("Average: " + stats.getAverage());
30         System.out.println("Sum: " + stats.getSum());
31         System.out.println("Min: " + stats.getMin());
32         System.out.println("Max: " + stats.getMax());
33
34
35         // Alternative design using Stream
36         DoubleSummaryStatistics stats2
37             = Person.createShortList()
38                 .stream()
39                 .map(Person::getAge)
40                 .collect(DoubleSummaryStatistics::new,
41                     DoubleSummaryStatistics::accept,
42                     DoubleSummaryStatistics::combine);
43
44         // Get statistics
45         System.out.println("Count: " + stats2.getCount());
46         System.out.println("Average: " + stats2.getAverage());
47         System.out.println("Sum: " + stats2.getSum());
48         System.out.println("Min: " + stats2.getMin());
49         System.out.println("Max: " + stats2.getMax());
50
51     }
52 }
```



2021/12/10 上午2:49	TestOptional.java	2021/12/10 上午2:49	TestOptional.java
1 /*		60 }	
2 * To change this license header, choose License Headers in Project Properties.		61 }	
3 * To change this template file, choose Tools   Templates		62 }	
4 * and open the template in the editor.		63 }	
5 */			
6 package ex11;			
7			
8 import java.util.List;			
9 import java.util.Optional;			
10			
11 /**			
12 *			
13 * @author wanting			
14 */			
15 public class TestOptional {			
16			
17     public static void main(String[] args) {			
18			
19         List<Person> list = Person.createShortList();			
20         testAge(list, 10);			
21         testAge(list, 25);			
22     }			
23			
24     public static void testAge(List<Person> list, int age) {			
25         // If age is not found, output:			
26         //     Not Found			
27         // If age is found, output:			
28         //     name			
29			
30         Optional<String> result = findAge(list, age);			
31			
32         if (result.isPresent()) {			
33             // if Optional has a value			
34             System.out.println(result.get()); // get the value			
35         } else {			
36             System.out.println("Not Found");			
37         }			
38			
39         // Alternative implementation using orElse()			
40         System.out.println(result.orElse("Not Found"));			
41			
42         // If age is found, output name in upper case			
43         System.out.println(result.map(String::toUpperCase).orElse("Not Found"));			
44     }			
45			
46     static Optional<String> findAge(List<Person> list, int age) {			
47         for (Person s : list) {			
48             if (s.getAge() == age) {			
49                 return Optional.of(s.getName());			
50             }			
51         }			
52			
53         return Optional.empty(); // name not found			
54     }			
55			
56     static void testFn(List<Person> list) {			
57			
58         Optional<String> result = findAge(list, 1234);			
59			
localhost:4649/?mode=click	1/2	localhost:4649/?mode=click	2/2