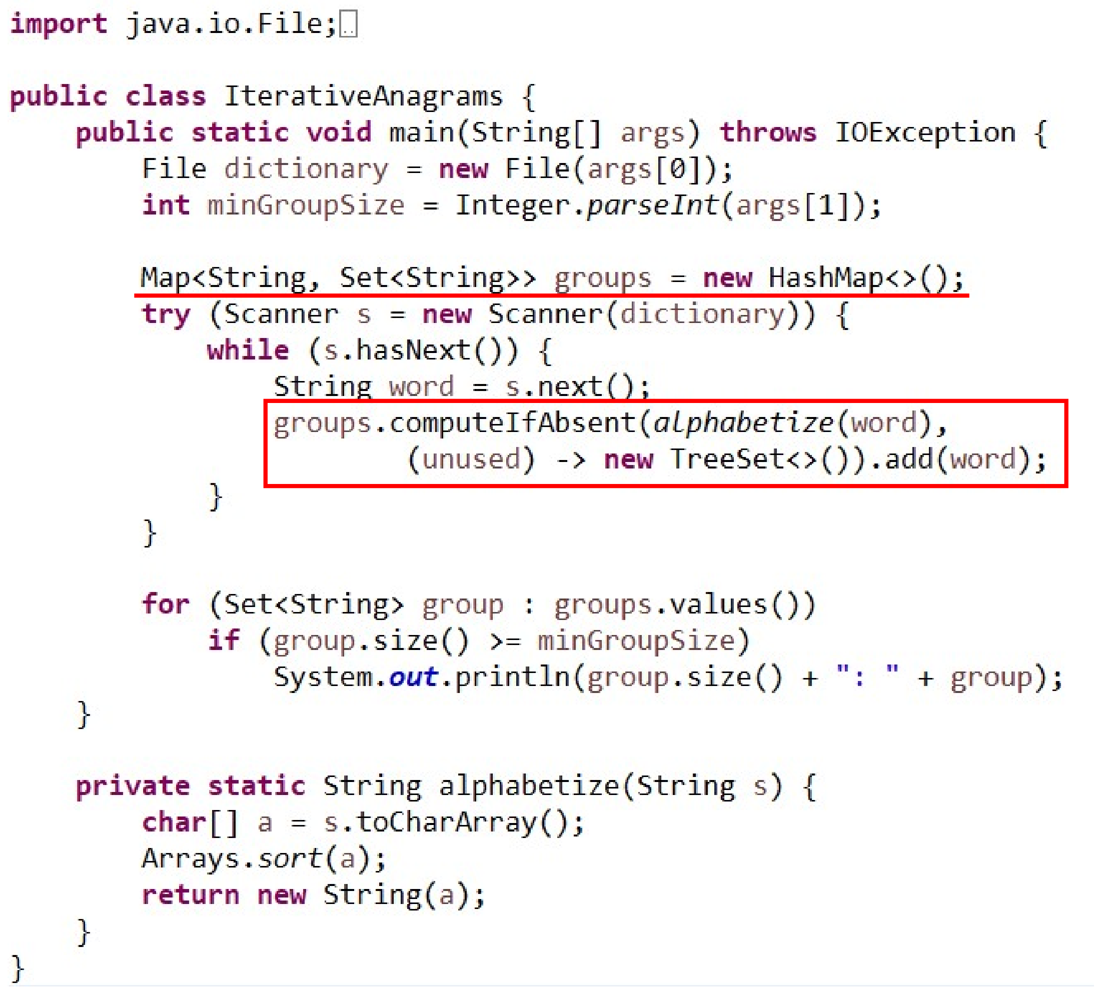
스트림

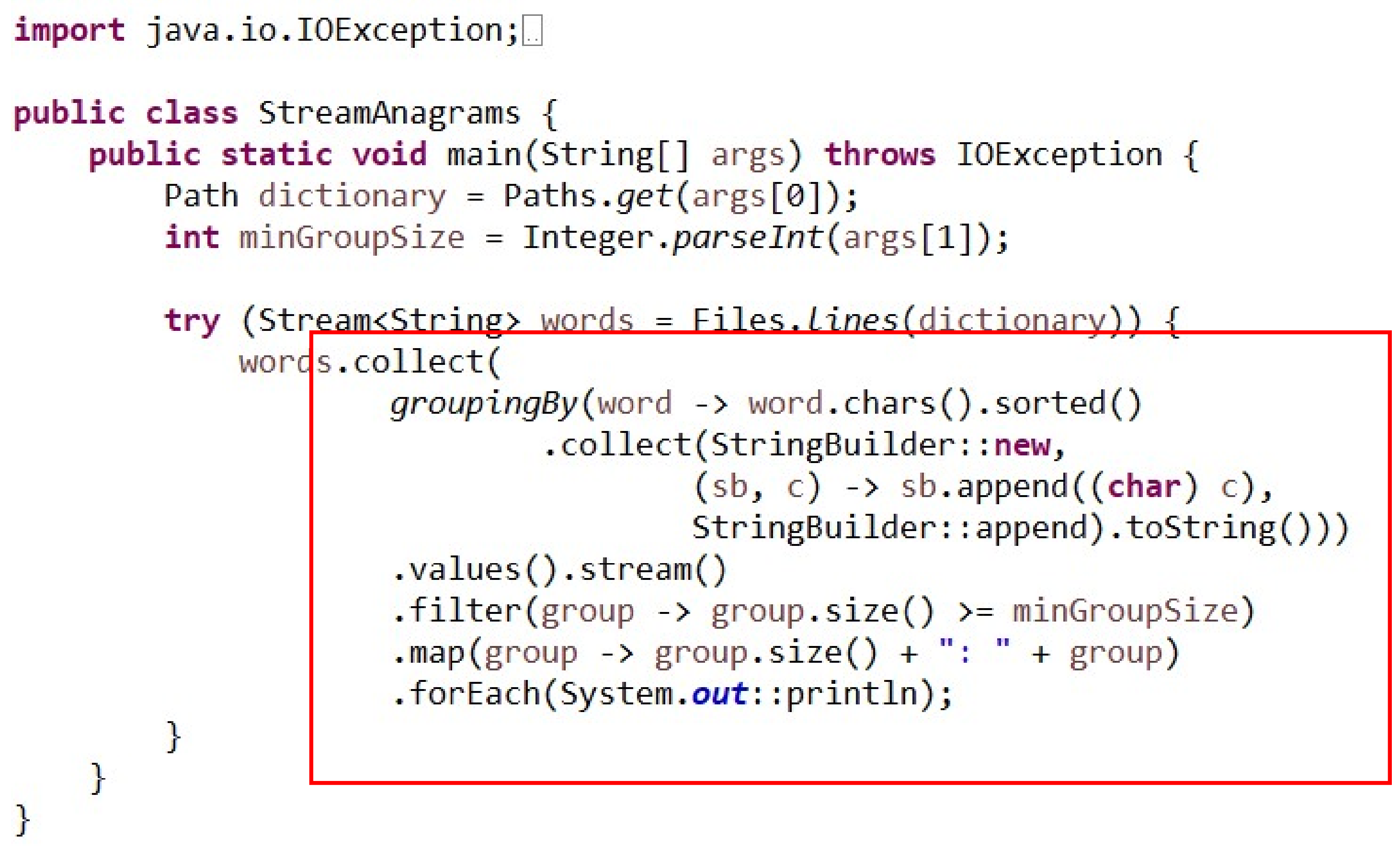
Effective Java

Code 1



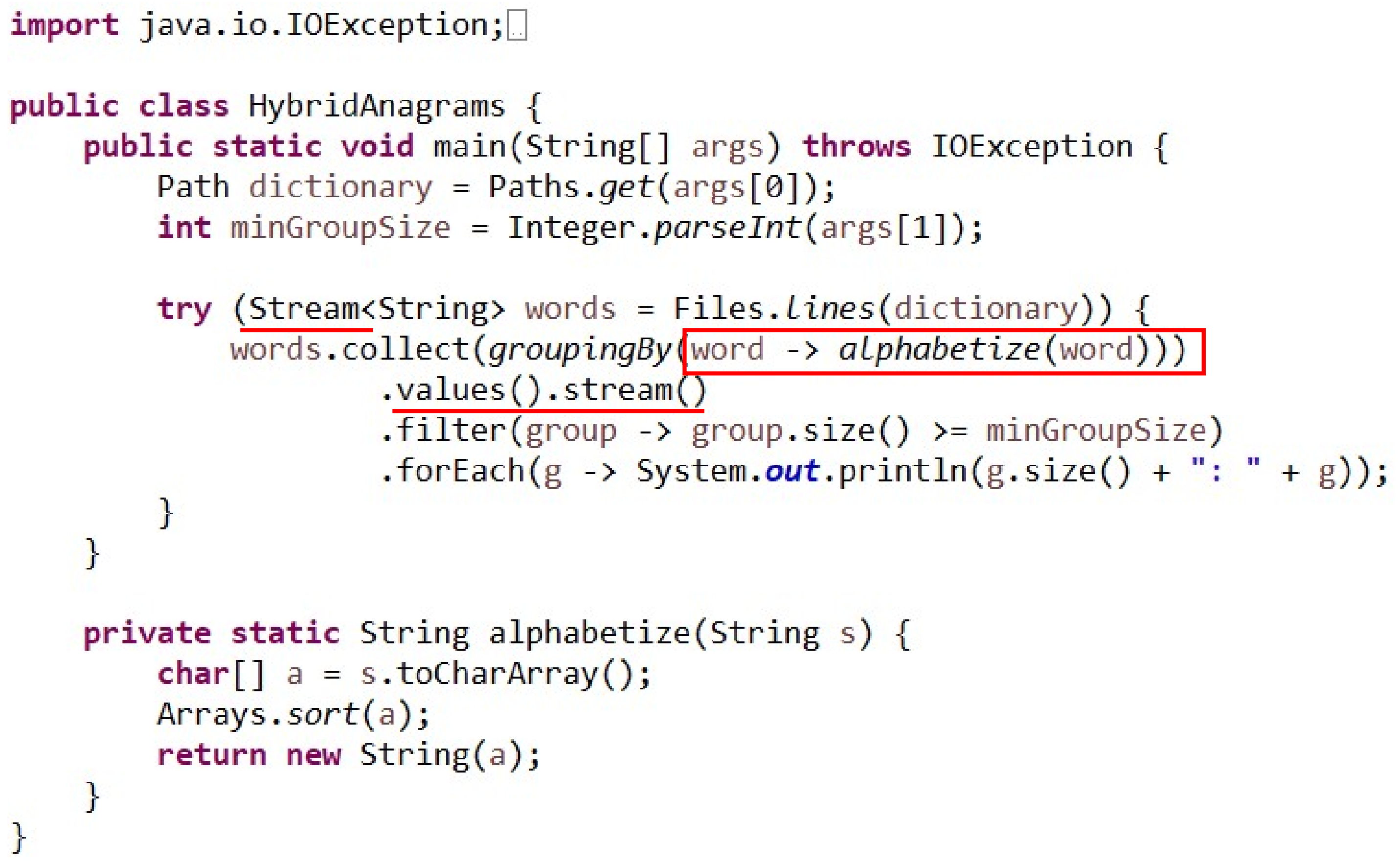
```java
import java.io.File;

public class IterativeAnagrams {
    public static void main(String[] args) throws IOException {
        File dictionary = new File(args[0]);
        int minGroupSize = Integer.parseInt(args[1]);

        Map<String, Set<String>> groups = new HashMap<>();
        try (Scanner s = new Scanner(dictionary)) {
            while (s.hasNext()) {
                String word = s.next();
                groups.computeIfAbsent(alphabetize(word),
                        (unused) -> new TreeSet<>()).add(word);
            }
        }

        for (Set<String> group : groups.values())
            if (group.size() >= minGroupSize)
                System.out.println(group.size() + ": " + group);
    }

    private static String alphabetize(String s) {
        char[] a = s.toCharArray();
        Arrays.sort(a);
        return new String(a);
    }
}
```

Code 2



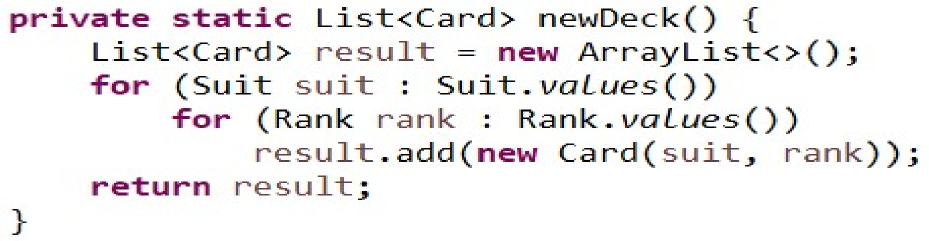
```java
import java.io.IOException;

public class StreamAnagrams {
    public static void main(String[] args) throws IOException {
        Path dictionary = Paths.get(args[0]);
        int minGroupSize = Integer.parseInt(args[1]);

        try (Stream<String> words = Files.lines(dictionary)) {
            words.collect(
                    groupingBy(word -> word.chars().sorted()
                            .collect(StringBuilder::new,
                                    (sb, c) -> sb.append((char) c),
                                    StringBuilder::append).toString()))
                    .values().stream()
                    .filter(group -> group.size() >= minGroupSize)
                    .map(group -> group.size() + ": " + group)
                    .forEach(System.out::println);

        }
    }
}
```

Code 3

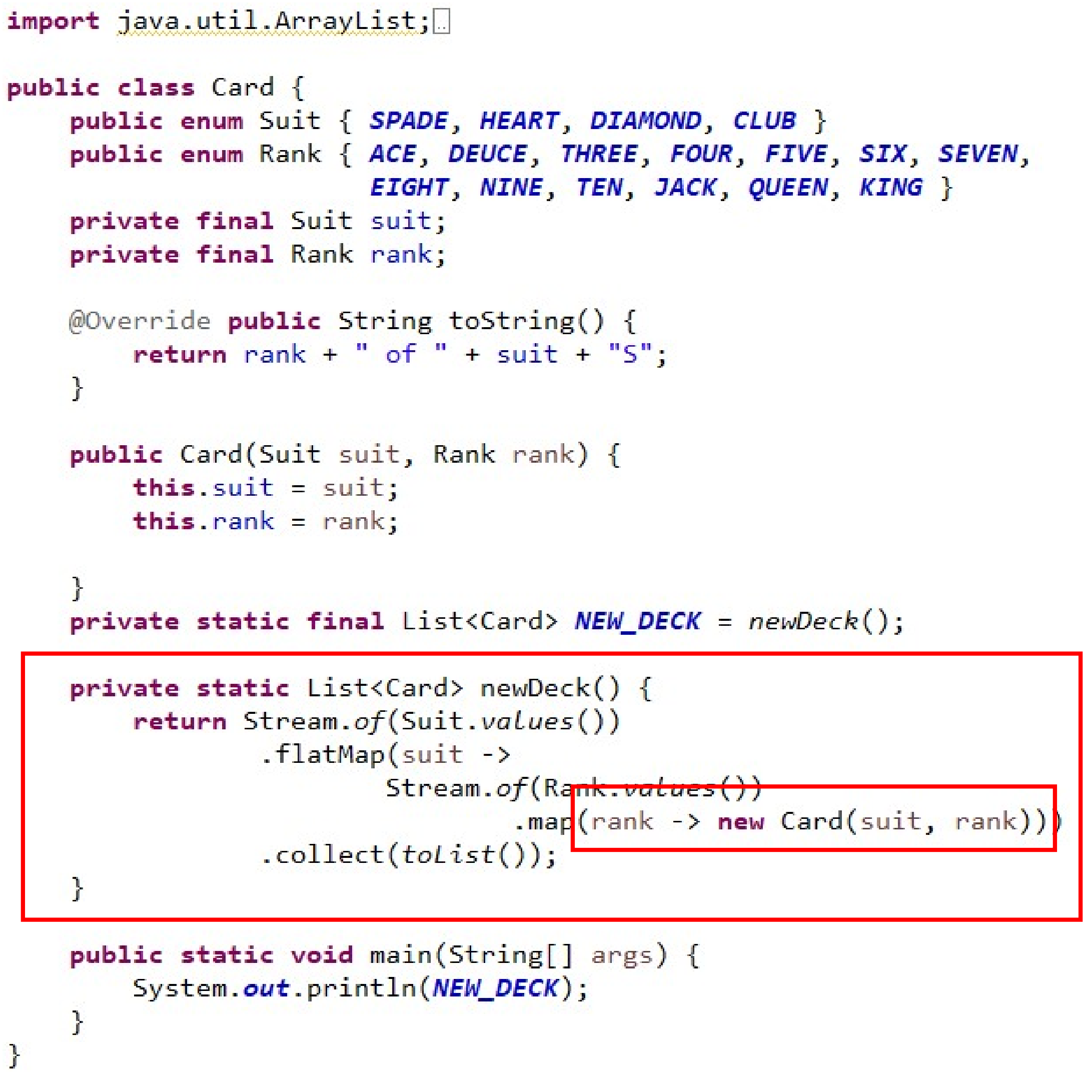


```java
import java.io.IOException;

public class HybridAnagrams {
    public static void main(String[] args) throws IOException {
        Path dictionary = Paths.get(args[0]);
        int minGroupSize = Integer.parseInt(args[1]);

        try (Stream<String> words = Files.lines(dictionary)) {
            words.collect(groupingBy(word -> alphabetize(word)))
                    .values().stream()
                    .filter(group -> group.size() >= minGroupSize)
                    .forEach(g -> System.out.println(g.size() + ": " + g));
        }
    }

    private static String alphabetize(String s) {
        char[] a = s.toCharArray();
        Arrays.sort(a);
        return new String(a);
    }
}
```

Code 4

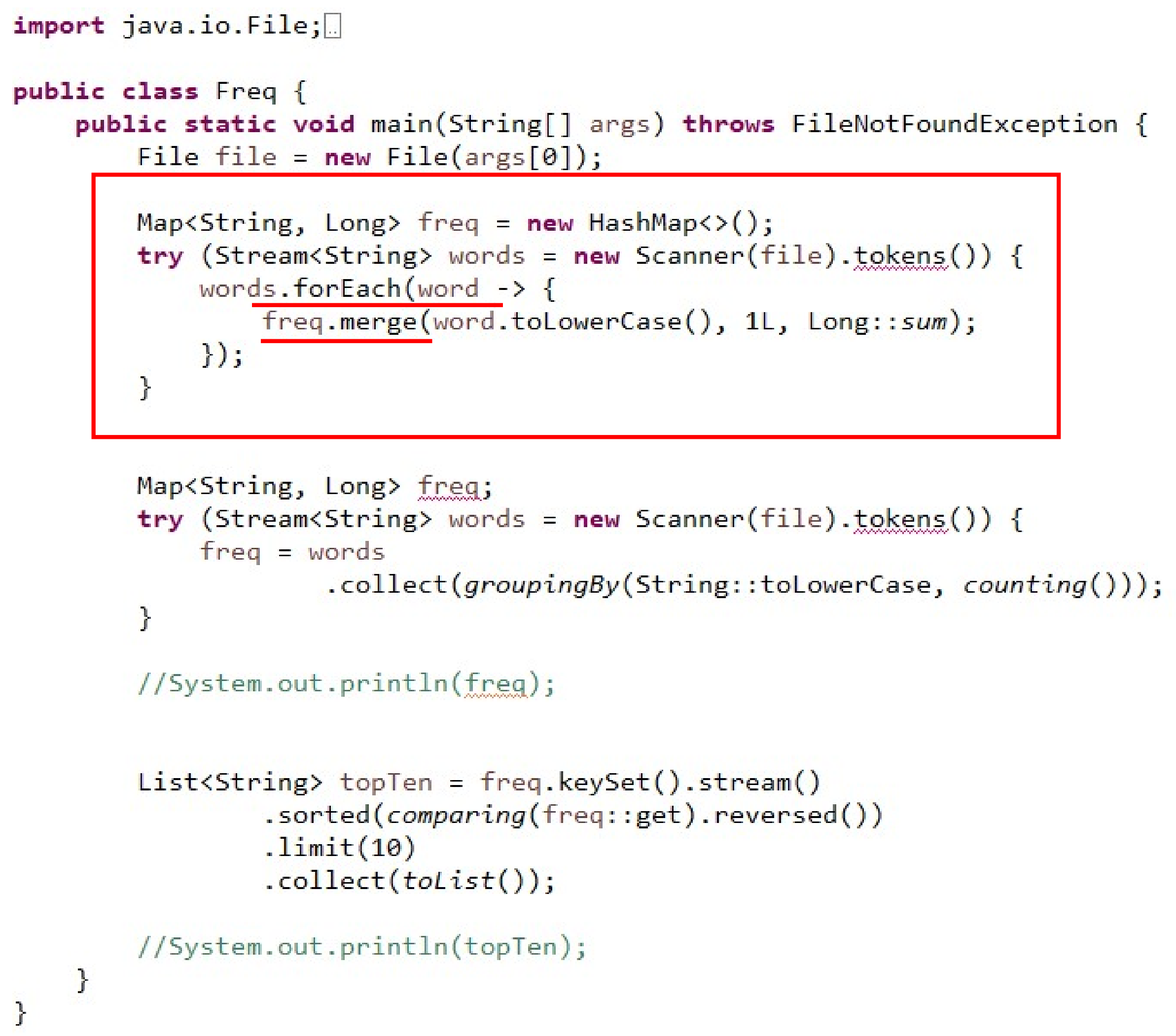


```java
private static List<Card> newDeck() {
    List<Card> result = new ArrayList<>();
    for (Suit suit : Suit.values())
        for (Rank rank : Rank.values())
            result.add(new Card(suit, rank));
    return result;
}
```



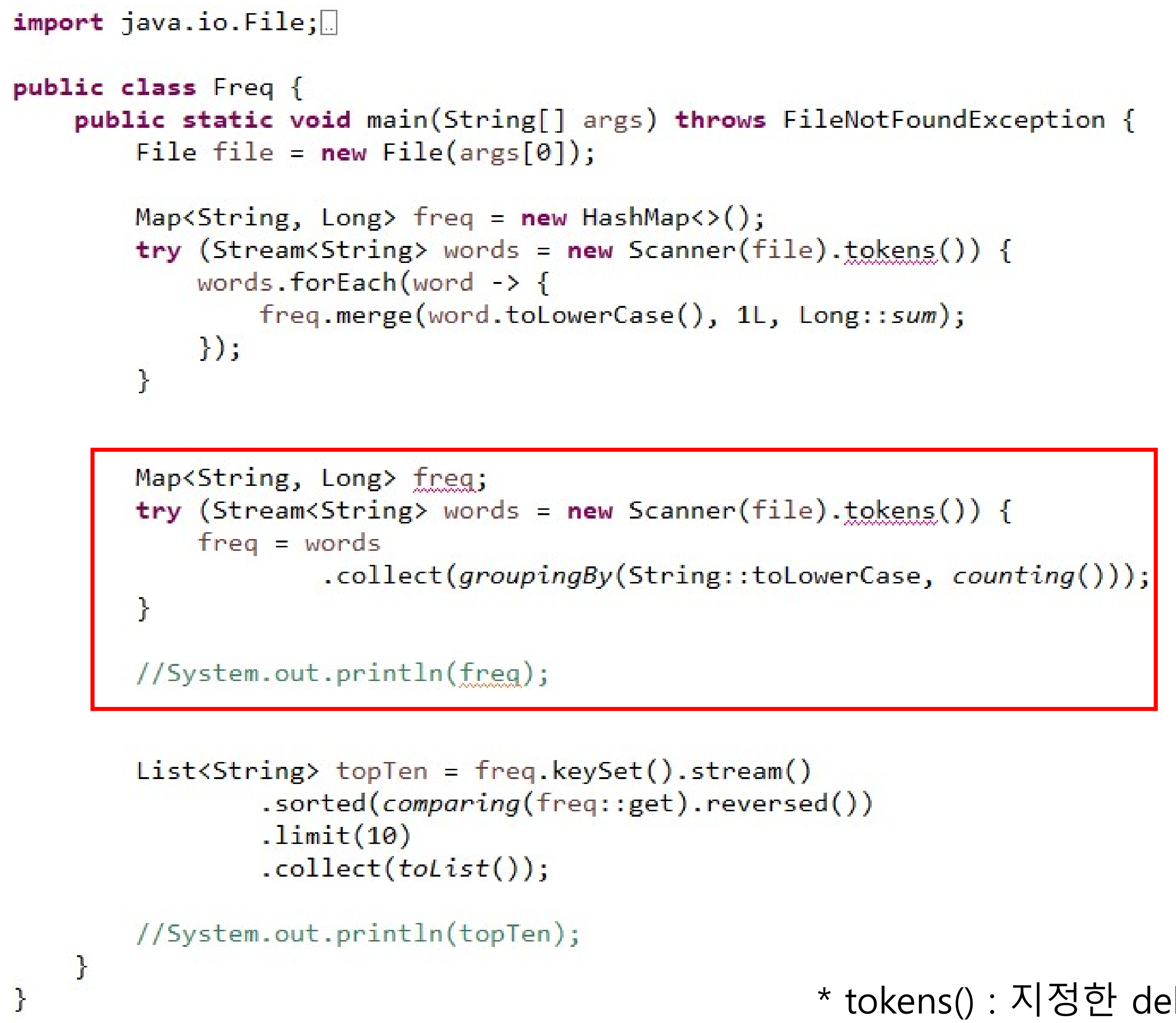
```java
import java.util.ArrayList;

public class Card {
    public enum Suit { SPADE, HEART, DIAMOND, CLUB }
    public enum Rank { ACE, DEUCE, THREE, FOUR, FIVE, SIX, SEVEN,
                       EIGHT, NINE, TEN, JACK, QUEEN, KING }
    private final Suit suit;
    private final Rank rank;

    @Override public String toString() {
        return rank + " of " + suit + "S";
    }

    public Card(Suit suit, Rank rank) {
        this.suit = suit;
        this.rank = rank;

    }
    private static final List<Card> NEW_DECK = newDeck();

    private static List<Card> newDeck() {
        return Stream.of(Suit.values())
                .flatMap(suit ->
                        Stream.of(Rank.values())
                                .map(rank -> new Card(suit, rank)))
                .collect(toList());
    }

    public static void main(String[] args) {
        System.out.println(NEW_DECK);
    }
}
```

Code 5~7



```java
import java.io.File;

public class Freq {
    public static void main(String[] args) throws FileNotFoundException {
        File file = new File(args[0]);

        Map<String, Long> freq = new HashMap<>();
        try (Stream<String> words = new Scanner(file).tokens()) {
            words.forEach(word -> {
                freq.merge(word.toLowerCase(), 1L, Long::sum);
            });
        }

        Map<String, Long> freq;
        try (Stream<String> words = new Scanner(file).tokens()) {
            freq = words
                    .collect(groupingBy(String::toLowerCase, counting()));
        }

        //System.out.println(freq);


        List<String> topTen = freq.keySet().stream()
                .sorted(comparing(freq::get).reversed())
                .limit(10)
                .collect(toList());

        //System.out.println(topTen);
    }
}
```

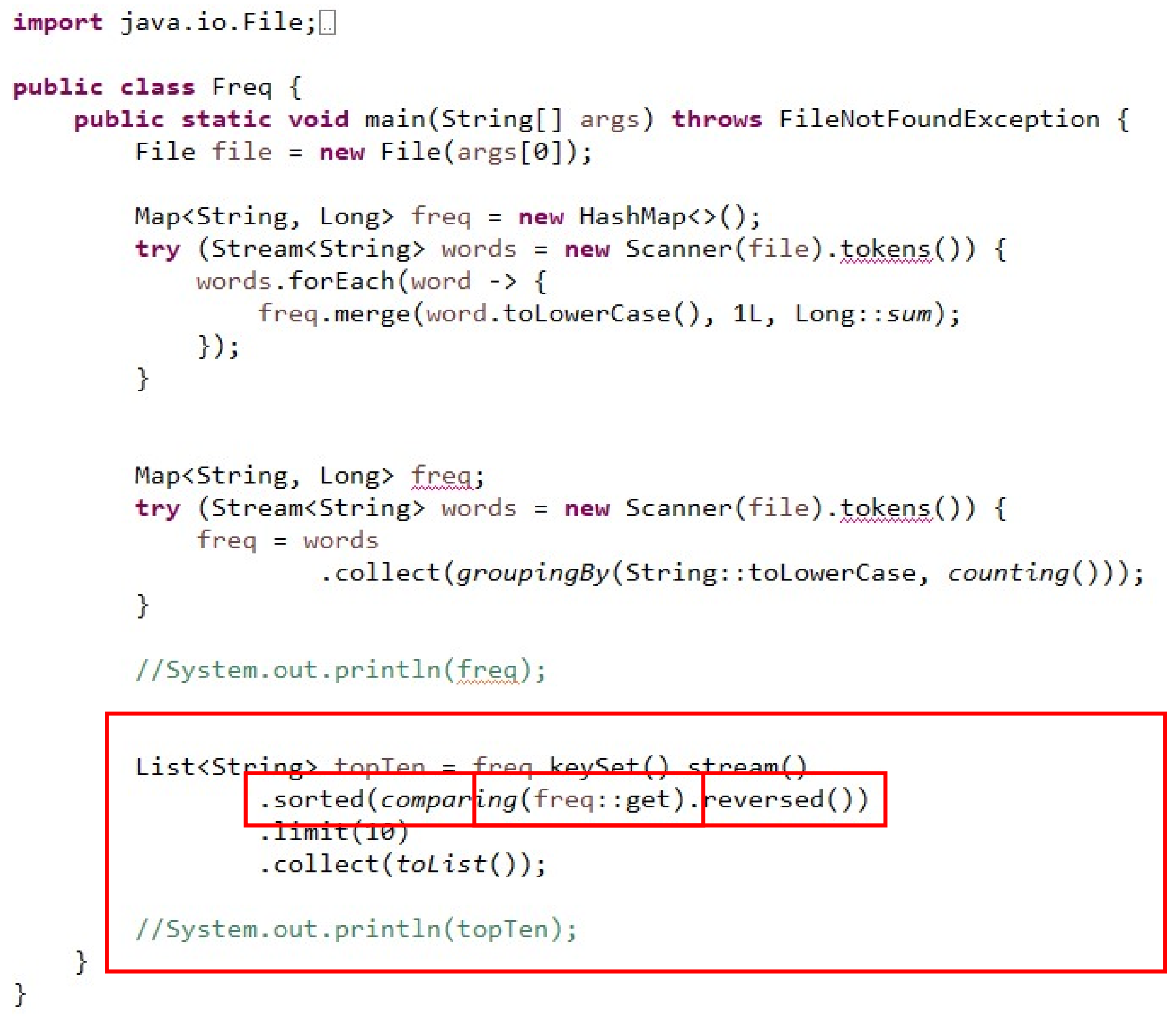
Code 5~7



```
import java.io.File;

public class Freq {
    public static void main(String[] args) throws FileNotFoundException {
        File file = new File(args[0]);

        Map<String, Long> freq = new HashMap<>();
        try (Stream<String> words = new Scanner(file).tokens()) {
            words.forEach(word -> {
                freq.merge(word.toLowerCase(), 1L, Long::sum);
            });
        }

        Map<String, Long> freq;
        try (Stream<String> words = new Scanner(file).tokens()) {
            freq = words
                    .collect(groupingBy(String::toLowerCase, counting()));
        }

        //System.out.println(freq);

        List<String> topTen = freq.keySet().stream()
                .sorted(comparing(freq::get).reversed())
                .limit(10)
                .collect(toList());

        //System.out.println(topTen);
    }
}
```

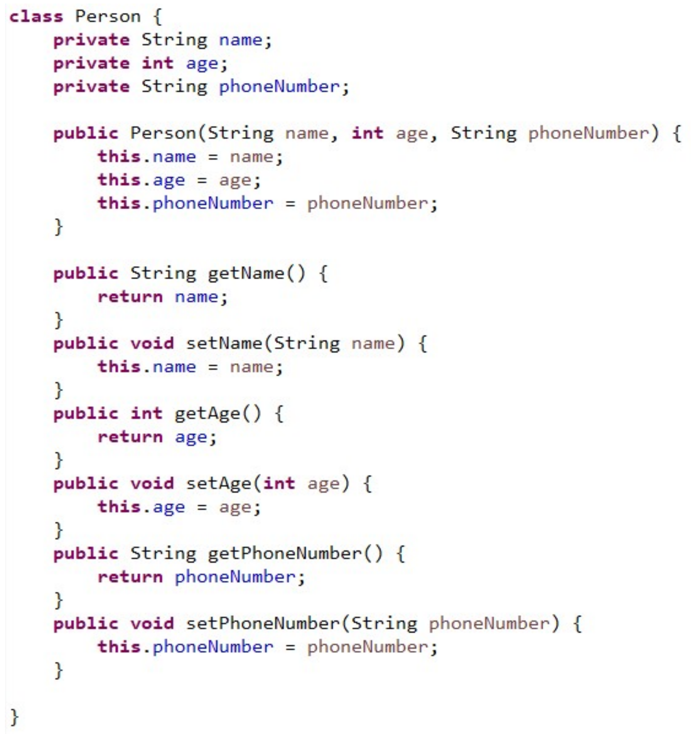
* tokens() : 지정한 delimiter로 tokenize 하는 메소드

Code 5~7

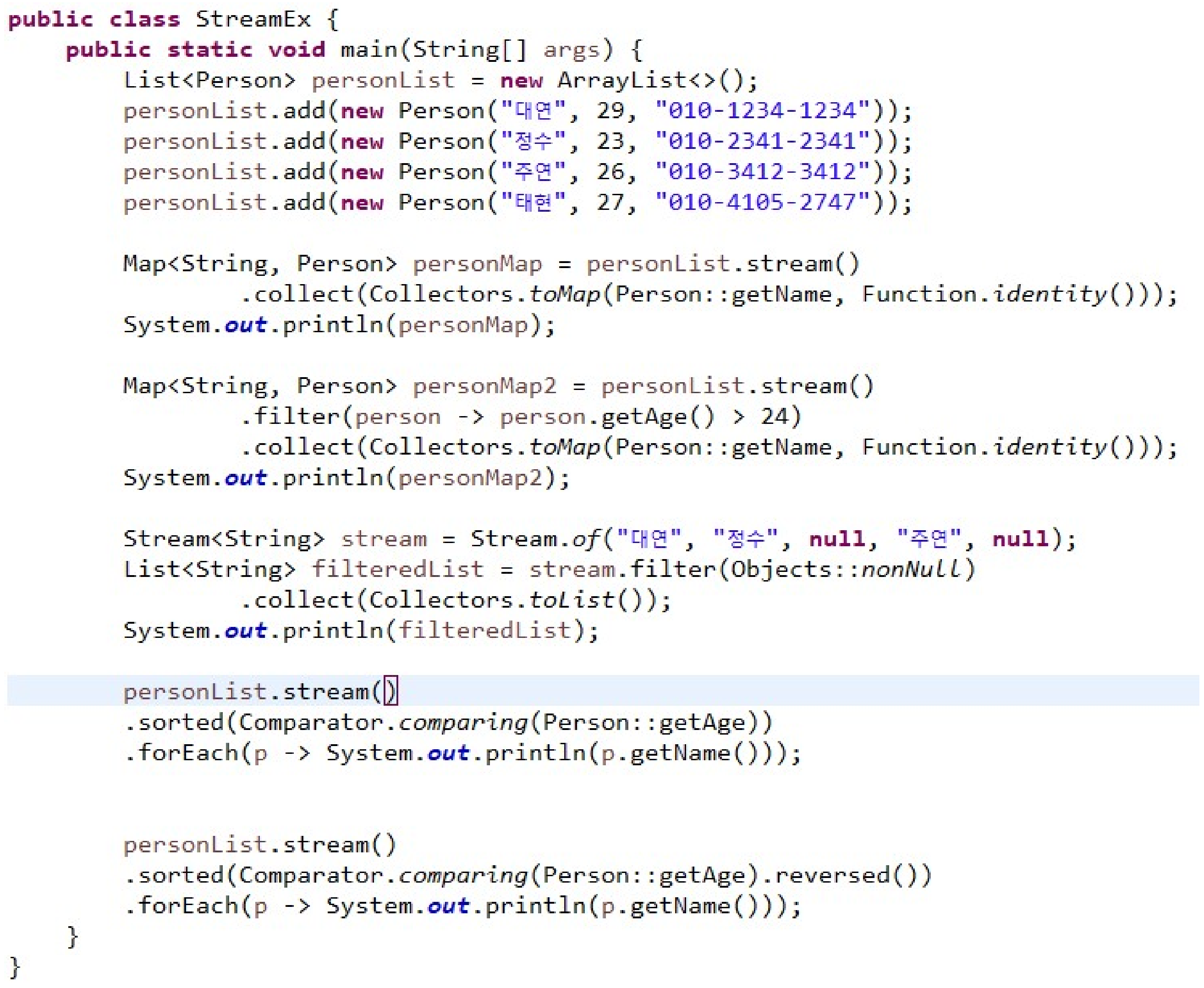


```java
import java.io.File;

public class Freq {
    public static void main(String[] args) throws FileNotFoundException {
        File file = new File(args[0]);

        Map<String, Long> freq = new HashMap<>();
        try (Stream<String> words = new Scanner(file).tokens()) {
            words.forEach(word -> {
                freq.merge(word.toLowerCase(), 1L, Long::sum);
            });
        }


        Map<String, Long> freq;
        try (Stream<String> words = new Scanner(file).tokens()) {
            freq = words
                    .collect(groupingBy(String::toLowerCase, counting()));
        }

        //System.out.println(freq);

        List<String> topTen = freq.keySet().stream()
                .sorted(comparing(freq::get).reversed())
                .limit(10)
                .collect(toList());

        //System.out.println(topTen);
    }
}
```

Code example



```java
class Person {
    private String name;
    private int age;
    private String phoneNumber;

    public Person(String name, int age, String phoneNumber) {
        this.name = name;
        this.age = age;
        this.phoneNumber = phoneNumber;
    }

    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
    public String getPhoneNumber() {
        return phoneNumber;
    }
    public void setPhoneNumber(String phoneNumber) {
        this.phoneNumber = phoneNumber;
    }

}
```



```java
public class StreamEx {
    public static void main(String[] args) {
        List<Person> personList = new ArrayList<>();
        personList.add(new Person("대연", 29, "010-1234-1234"));
        personList.add(new Person("정수", 23, "010-2341-2341"));
        personList.add(new Person("주연", 26, "010-3412-3412"));
        personList.add(new Person("태현", 27, "010-4105-2747"));

        Map<String, Person> personMap = personList.stream()
                .collect(Collectors.toMap(Person::getName, Function.identity()));
        System.out.println(personMap);

        Map<String, Person> personMap2 = personList.stream()
                .filter(person -> person.getAge() > 24)
                .collect(Collectors.toMap(Person::getName, Function.identity()));
        System.out.println(personMap2);

        Stream<String> stream = Stream.of("대연", "정수", null, "주연", null);
        List<String> filteredList = stream.filter(Objects::nonNull)
                .collect(Collectors.toList());
        System.out.println(filteredList);

        personList.stream()
        .sorted(Comparator.comparing(Person::getAge))
        .forEach(p -> System.out.println(p.getName()));


        personList.stream()
        .sorted(Comparator.comparing(Person::getAge).reversed())
        .forEach(p -> System.out.println(p.getName()));
    }
}
```