

Quick Tour

Java Language Features 8 - 17



A large orange circle is positioned on the left side of the slide, covering approximately one-third of the vertical space.

JAVA 8

Stream API

Functional Interface

Lambda Expressions

Interface Default & Static Method

Optional Class

Method Reference

Stream API

```
public List<Account> getAccountDetailsFromAccountNumber(final long accountNumber) throws FileNotFoundException {  
    final List<Account> accountData = new ArrayList<>();  
  
    for (Account accountRecord : readData.read()) {  
        if (accountNumber == accountRecord.getAccountNumber()) {  
            accountData.add(accountRecord);  
        }  
    }  
  
    return accountData;  
}
```

```
public List<Account> getAccountDetailsFromAccountNumber(final long accountNumber) throws FileNotFoundException {  
  
    return readData.read() List<Account>  
        .stream() Stream<Account>  
        .filter(a -> a.getAccountNumber() == accountNumber)  
        .collect(Collectors.toList());  
}
```

Functional Interface & Lambda Expression ->

```
@FunctionalInterface  
public interface Square {  
    int calculate(int n);  
}  
  
@Test  
void calculateSquareOfGivenNumber() {  
  
    final Square square = new Square() {  
        @Override  
        public int calculate(int n) {  
            return n * n;  
        }  
    };  
  
    System.out.println(square.calculate(5));  
}
```

```
@FunctionalInterface  
public interface Square {  
    int calculate(int n);  
}  
  
@Test  
void calculateSquareOfGivenNumber() {  
    final Square square = (int n) -> n*n;  
  
    System.out.println(square.calculate(5));  
}
```

Interface Default & Static Method

Default

- It is a method with default keyword
- We can provide the implementation in interface or let class provide it
- Class can override this method

Static

- It is a static method which belongs to the interface only.
- We have to provide implementation of this method in interface itself
- Class can not override this method

Interface Static & Default Method

30/11/2021

Sunil Ghargaonkar

The screenshot shows an IDE interface with a Java code editor and a terminal window below it.

```
6
7     @Test
8     void defaultInterfaceMethodTest() {
9         final Welcome welcome = new InviteGuest();
10
11         Welcome.invite();
12         welcome.greet();
13         welcome.serve();
14     }
15
16     interface Welcome {
17         static void invite() {
18             System.out.println("Let's invite guests for the barbecue party");
19         }
20
21         default void greet() {
22             System.out.println("Hello Everyone!");
23         }
24
25         void serve();
26     }
27
28     static class InviteGuest implements Welcome {
29
30         @Override
31         public void serve() { System.out.println("Would you like to have some drinks?"); }
32
33         @Override
34         public void greet() {
35             System.out.println("Hello Java how is world?");
36         }
37     }
38 }
39
40 }
```

The code defines an interface `Welcome` with a static method `invite()`, a default method `greet()`, and a regular method `serve()`. It also contains a static class `InviteGuest` that implements the `Welcome` interface, providing concrete implementations for the `serve()` and `greet()` methods.

In the terminal window, the output of the test execution is shown:

```
est.defaultInterfaceMethodTest x
Tests passed: 1 of 1 test - 19 ms
/Library/Java/JavaVirtualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java ...
Let's invite guests for the barbecue party
Hello Java how is world?
Would you like to have some drinks
Process finished with exit code 0
```



```
public class EmailStoreHeaderName implements DisplayNameProvider {
    private final EmailStore emailStore;
    private DisplayUserName displayUserName;

    public EmailStoreHeaderName(final EmailStore emailStore) {
        this.emailStore = emailStore;
    }

    @Override
    public DisplayName getDisplayName(final long type) {
        if(emailStore != null){
            final String headerName = emailStore.getFromHdrName();

            if(headerName != null && !headerName.isEmpty()){
                displayUserName = new DisplayName(headerName, DisplayName.HeaderFromType.HEADER, isSuspicious: false, type);
            }
        }
        return displayUserName;
    }
}
```

Optional Class

```
public class EmailStoreHeaderName implements DisplayNameProvider {
    private final EmailStore emailStore;

    public EmailStoreHeaderName(final EmailStore emailStore) {
        this.emailStore = emailStore;
    }

    @Override
    public Optional<DisplayName> getDisplayName(final long type) {
        return Optional.ofNullable(emailStore).map(EmailStore::getFromHdrName)
            .filter(headerName -> !headerName.isEmpty())
            .map(displayName -> new DisplayName(displayName, DisplayName.HeaderFromType.HEADER, isSuspicious: false, type));
    }
}
```

Method Reference

```
@Override
public Optional<DisplayUserName> getDisplayName(final long type) {
    return Optional.ofNullable(emailStore).map(name -> name.getFromHdrName())
        .filter(headerName -> !headerName.isEmpty())
        .map(displayName -> new DisplayUserName(displayName, DisplayUserName.HeaderFromType.HEADER, isSuspicious: false, type));
}
```

```
@Override
public Optional<DisplayUserName> getDisplayName(final long type) {
    return Optional.ofNullable(emailStore).map(EmailStore::getFromHdrName)
        .filter(headerName -> !headerName.isEmpty())
        .map(displayName -> new DisplayUserName(displayName, DisplayUserName.HeaderFromType.HEADER, isSuspicious: false, type));
}
```

A large orange circle is positioned on the left side of the slide, covering approximately one-third of the vertical space.

JAVA 9

Module System

Interface Private Methods

Immutable Collection Methods

Stream API Improvement

Jshell

Immutable Collection Methods

```
@Test
void listSetAndMapOf() {
    List<String> names = new ArrayList<>();
    Set<String> state = new HashSet<>();
    Map<String, String> personDetails = new HashMap<>();

    names.add("Sunil");
    names.add("Simran");

    state.add("Maharashtra");
    state.add("Punjab");

    personDetails.put("Sunil", "Maharashtra");
    personDetails.put("Simran", "Punjab");

    names.remove( index: 0);

    System.out.println(names);
}
```

```
@Test
void listSetAndMapOf2() {
    List<String> names2 = List.of("Sunil", "Simran");
    Set<String> state2 = Set.of("Maharashtra", "Punjab");
    Map<String, String> personDetails2 = Map.of(k1: "Sunil", v1: "Maharashtra", k2: "Simran", v2: "Punjab");

    names2.remove( index: 0);
    System.out.println(names2);
}

tAndMapOf2
Tests failed: 1 of 1 test - 16 ms

java.lang.UnsupportedOperationException Create breakpoint
    at java.base/java.util.ImmutableCollections$oe(ImmutableCollections.java:142)
    at java.base/java.util.ImmutableCollections$AbstractImmutableList.remove(ImmutableCollections.java:258)
    at com.java.test.programs.CollectionFactoryMethodTest.listSetAndMapOf2(CollectionFactoryMethodTest.java:35) <31 int
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <21 internal lines>
```

Stream API Improvement

- takeWhile
- dropWhile
- ofNullable
- Iterate

takeWhile

```
@Test
void takeWhileVsFilter() {
    System.out.println("----takeWhile----");
    IntStream.of(1, 10, 100, 1000, 10000, 1000, 100, 10, 1, 0, 10000)
        .takeWhile(i -> i < 5000)
        .forEach(System.out::println);

    System.out.println("----filter----");
    IntStream.of(1, 10, 100, 1000, 10000, 1000, 100, 10, 1, 0, 10000)
        .filter(i -> i < 5000)
        .forEach(System.out::println);
}
```

Tests passed: 1 of 1 test - 19 ms

```
----takeWhile----
1
10
100
1000
----filter----
1
10
100
1000
1000
100
10
1
0
```

dropWhile

```
@Test
void dropWhileVsFilter() {
    System.out.println("----takeWhile----");
    IntStream.of(2, 4, 6, 8, 9, 10, 12)
        .dropWhile(n -> n % 2 == 0)
        .forEach(System.out::println);

    System.out.println("----filter----");
    IntStream.of(2, 4, 6, 8, 9, 10, 12)
        .filter(n -> n % 2 == 0)
        .forEach(System.out::println);
}
```

✓ Tests passed: 1 of 1 test - 25 ms

```
/Library/Java/JavaVirtualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java ...
```

```
----takeWhile----
```

```
9
```

```
10
```

```
12
```

```
----filter----
```

```
2
```

```
4
```

```
6
```

```
8
```

```
10
```

```
12
```



jshell

```
sghargaonkar@mc-lon-nb15243 bin % jshell
| Welcome to JShell -- Version 17.0.1
| For an introduction type: /help intro
```

```
jshell> String program = "--Hello World--";
program ==> "--Hello World--"
```

```
jshell> program.replace('-', '*');
$2 ==> "**Hello World**"
```

```
jshell> █
```

JAVA 10

Unmodifiable Collection Copy

Optional orElseThrow()

Var for local variables var length = str.length();

Time-Based Versioning

Unmodifiable Collection Copy

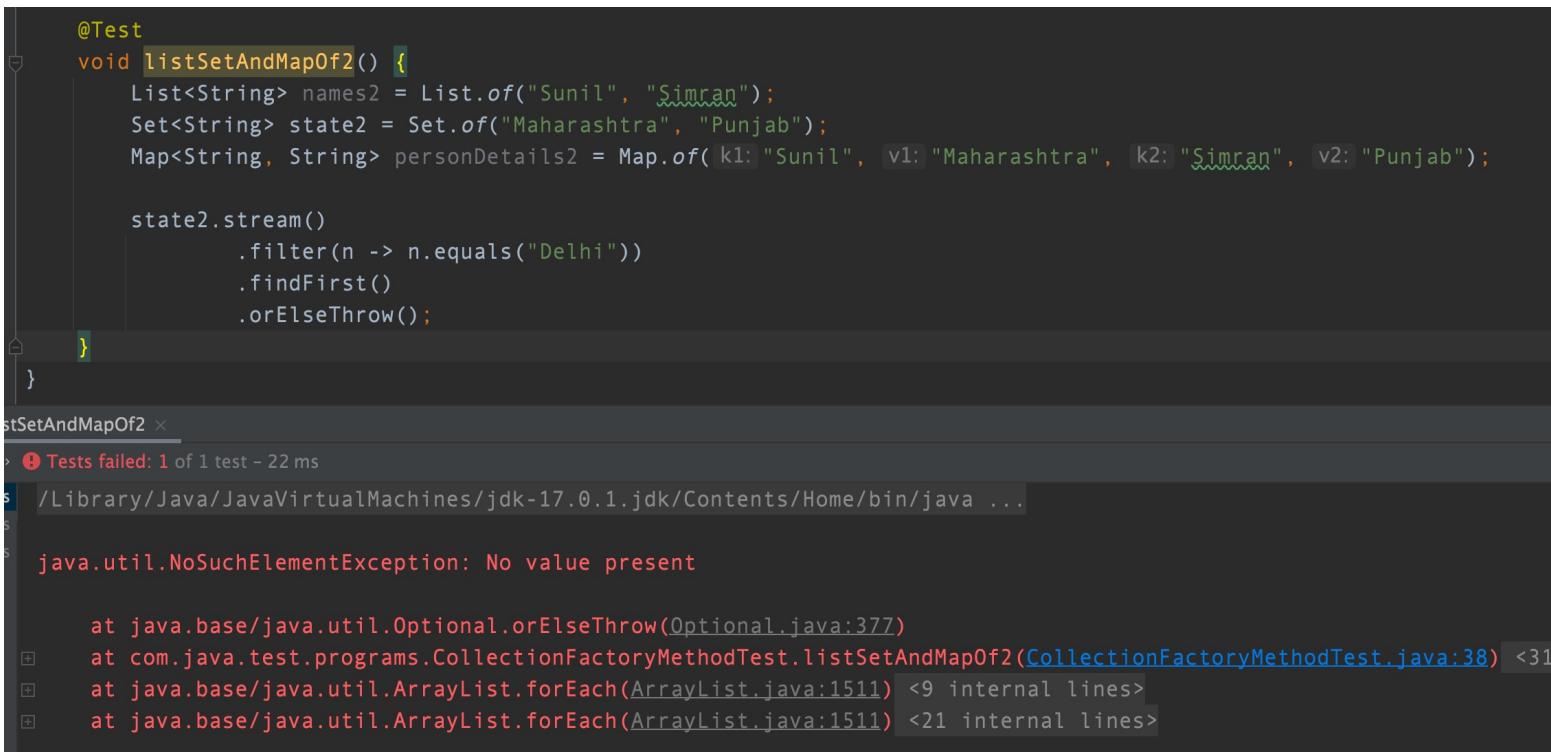


```
@Test
void listSetAndMapOf2() {
    List<String> names = List.of("Sunil", "Simran");
    Set<String> states = Set.of("Maharashtra", "Punjab");
    Map<String, String> personDetails2 = Map.of( k1: "Sunil", v1: "Maharashtra", k2: "Simran", v2: "Punjab");

    Set<String> newStates = Set.copyOf(states);

    List<String> state = newStates.stream()
        .filter(s -> s.equals("Punjab"))
        .collect(Collectors.toUnmodifiableList());
}
```

orElseThrow()



The screenshot shows a Java test method named `listSetAndMapOf2`. The code creates three collections: a list of strings, a set of strings, and a map of strings. It then filters the set to find the first element that equals "Delhi", and if none is found, it calls `.orElseThrow()`. The terminal output shows a failure due to a `NoSuchElementException`, which is caught by the `orElseThrow` block. The stack trace indicates the exception was thrown at the `Optional.orElseThrow` line.

```
@Test
void listSetAndMapOf2() {
    List<String> names2 = List.of("Sunil", "Simran");
    Set<String> state2 = Set.of("Maharashtra", "Punjab");
    Map<String, String> personDetails2 = Map.of( k1: "Sunil", v1: "Maharashtra", k2: "Simran", v2: "Punjab");

    state2.stream()
        .filter(n -> n.equals("Delhi"))
        .findFirst()
        .orElseThrow();
}

```

```
Tests failed: 1 of 1 test - 22 ms
java.util.NoSuchElementException: No value present
    at java.base/java.util.Optional.orElseThrow(Optional.java:377)
    at com.java.test.programs.CollectionFactoryMethodTest.listSetAndMapOf2(CollectionFactoryMethodTest.java:38) <31
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <21 internal lines>
```

Java -version



```
sghargaonkar@mc-lon-nb15243 ~ % java -version
openjdk version "17.0.1" 2021-10-19
OpenJDK Runtime Environment (build 17.0.1+12-39)
OpenJDK 64-Bit Server VM (build 17.0.1+12-39, mixed mode, sharing)
sghargaonkar@mc-lon-nb15243 ~ %
```

JAVA 11

TLS V1.3

New HTTP Client API

New String methods (repeat, isBlank , strip , lines)

New File Methods

Not Predicate

Collection to Array

```
var length = str.length();
```

Not Predicate



```
List<String> names = List.of("Sunil", "Simran");
Set<String> states = Set.of("Maharashtra", "Punjab");
Map<String, String> personDetails2 = Map.of(k1: "Sunil", v1: "Maharashtra", k2: "Simran", v2: "Punjab");

Set<String> newStates = Set.copyOf(states);

List<String> state = newStates.stream()
    .filter(Predicate.not(s -> s.equals("Punjab")))
    .collect(Collectors.toUnmodifiableList());
```

Collection to Array



```
@Test
void listSetAndMapOf2() {
    List<String> names = List.of("Sunil", "Simran");
    Set<String> states = Set.of("Maharashtra", "Punjab");
    Map<String, String> personDetails2 = Map.of(k1: "Sunil", v1: "Maharashtra", k2: "Simran", v2: "Punjab");

    Set<String> newStates = Set.copyOf(states);

    String [] state = newStates.stream()
        .filter(Predicate.not(s -> s.equals("Punjab")))
        .toArray(String[]::new);
}
```



JAVA 12

File::mismatch method

Switch Expressions (Preview)

File:: mismatch



```
@Test
void tempFileCreationAndMatchTest() throws IOException {
    final Path filePath1 = Files.writeString(Files.createTempFile( prefix: "demo1", suffix: ".txt"), csq: "Test JAVA 12 feature");

    final Path filePath2 = Files.writeString(Files.createTempFile( prefix: "demo2", suffix: ".txt"), csq: "Test JAVA 12 feature");

    long mismatch = Files.mismatch(filePath1, filePath2);
    assertEquals( expected: -1, mismatch);
}
```

A large, solid orange circle is positioned on the left side of the slide, partially overlapping the white background.

JAVA 13

Switch Expressions (preview)

Text Blocks (preview)

A large, solid orange circle is positioned on the left side of the slide, covering approximately one-third of the vertical space.

JAVA 14

NullPointerExceptions with StackTrace

Switch Expressions

Text Blocks (preview)



NPE StackTrace

The screenshot shows an IDE interface with a failed test result. The test method is annotated with @Test and @DisplayName("Throws exception when the passed configuration is null"). It creates a new DomainCategory object with config set to null and domainScanClient set to a DomainScanClientV2 instance. The test fails with a NullPointerException. The stack trace shows the exception was thrown at DomainCategory.java:44, which calls DomainCategory.java:52, and the test method at DomainCategoryTest.java:72. The stack trace also includes internal lines from java.util.ArrayList.forEach.

```
@Test
@DisplayName("Throws exception when the passed configuration is null")
void throwsExceptionWhenConfigIsNull() {
    new DomainCategory(config: null, domainScanClient);
}

exceptionWhenConfigIsNull ×
Tests failed: 1 of 1 test - 3 sec 971 ms

java.lang.NullPointerException Create breakpoint
    at com.mimecast.mta.scanner.domain.DomainCategory.<init>(DomainCategory.java:44)
    at com.mimecast.mta.scanner.domain.DomainCategory.<init>(DomainCategory.java:52)
    at com.mimecast.mta.scanner.domain.DomainCategoryTest.throwsExceptionWhenConfigIsNull(DomainCategoryTest.java:72)
    <31 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <9 internal lines>
    at java.util.ArrayList.forEach(ArrayList.java:1257) <21 internal lines>
```

The screenshot shows another IDE interface with a failed test result. The test class is NPEStrackTraceTest. It contains a static class DomainCategory and a static class DomainScanClientV2. The test method is annotated with @Test and named testNPEStrackTrace. It creates a new DomainCategory object with config set to null and domainScanClient set to a DomainScanClientV2 instance. The test fails with a NullPointerException. The stack trace shows the exception was thrown at NPEStrackTraceTest.java:10, which calls NPEStrackTraceTest.java:19, and the test method at NPEStrackTraceTest.java:19. The stack trace also includes internal lines from java.util.ArrayList.forEach.

```
class NPEStrackTraceTest {
    DomainScanClientV2 domainScanClient = new DomainScanClientV2();

    static class DomainCategory {
        public DomainCategory(final String config, DomainScanClientV2 client) {
            config.equals("Test");
        }
    }

    static class DomainScanClientV2 {
    }

    @Test
    void testNPEStrackTrace() {
        new DomainCategory(config: null, domainScanClient);
    }
}

Trace ×
Tests failed: 1 of 1 test - 24 ms
/Library/Java/JavaVirtualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java ...

java.lang.NullPointerException: Cannot invoke "String.equals(Object)" because "config" is null

    at com.java.test.programs.NPEStrackTraceTest$DomainCategory.<init>(NPEStrackTraceTest.java:10)
    at com.java.test.programs.NPEStrackTraceTest.testNPEStrackTrace(NPEStrackTraceTest.java:19) <31
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <9 internal lines>
    at java.base/java.util.ArrayList.forEach(ArrayList.java:1511) <21 internal lines>
```

Switch Expression



```
switch (choice) {  
    case 1:  
        final AccountBalance accountBalance = new AccountBalance(accountNumber);  
        System.out.println(accountBalance.formatAccountBalance());  
        break;  
    case 2:  
        final Withdraw withdraw = new Withdraw(accountNumber);  
        withdraw.withdrawAmount(amount);  
        break;  
    case 3:  
        final Deposit deposit = new Deposit(accountNumber);  
        deposit.depositMoney(amount);  
        break;  
    default:  
        System.out.println("Unable to find the choice");  
}
```

```
switch (choice) {  
    case 1 -> {  
        final AccountBalance accountBalance = new AccountBalance(accountNumber);  
        System.out.println(accountBalance.formatAccountBalance());  
    }  
    case 2 -> {  
        final Withdraw withdraw = new Withdraw(accountNumber);  
        withdraw.withdrawAmount(amount);  
    }  
    case 3 -> {  
        final Deposit deposit = new Deposit(accountNumber);  
        deposit.depositMoney(amount);  
    }  
    default -> System.out.println("Unable to find the choice");  
}
```

A large, solid orange circle is positioned on the left side of the slide, covering approximately one-third of the vertical space.

JAVA 15

Record Class

Sealed Classes

Text Blocks

Text Blocks



```
@Test
void name() {
    String JSON_RESPONSE = "{\n        \"results\": [\n            {\n                \"hit\": true,\n                \"value\": \"\",\n                \"alias\": \"freemail_domains.map\",\n                \"key\": \"gmail.com\",\n                \"map\": \"/etc/rspamd/maps.d/freemail_domains.map\"\n            }\n        ],\n        \"success\": true\n    }";
    String JSON_RESPONSE2 = """
        {
            \"results\": [
                {
                    \"hit\": true,
                    \"value\": "",
                    \"alias\": \"freemail_domains.map\",
                    \"key\": \"gmail.com\",
                    \"map\": \"/etc/rspamd/maps.d/freemail_domains.map\"
                }
            ],
            \"success\": true
        }""";
    System.out.println(JSON_RESPONSE);
    System.out.println("-----");
    System.out.println(JSON_RESPONSE2);
}
```

```
e.name x
✓ Tests passed: 1 of 1 test ~ 22 ms
/Library/Java/JavaVirtualMachines/jdk-17.0.1.jdk/Contents/Home/bin/java ...
{
    "results": [
        {
            "hit": true,
            "value": "",
            "alias": "freemail_domains.map",
            "key": "gmail.com",
            "map": "/etc/rspamd/maps.d/freemail_domains.map"
        }
    ],
    "success": true
}
-----
{
    "results": [
        {
            "hit": true,
            "value": "",
            "alias": "freemail_domains.map",
            "key": "gmail.com",
            "map": "/etc/rspamd/maps.d/freemail_domains.map"
        }
    ],
    "success": true
}
```



JAVA 16

Record Class

Collection convertor method

Sealed Class support for Inner Class

Record Class

```
@Test
void recordClassTest() {
    BooksWithRecord booksWithRecord = new BooksWithRecord( title: "Can't hurt me", author: "David Goggins");
    System.out.printf(" Title: %s%n Author: %s%n", booksWithRecord.title(), booksWithRecord.author());
    BooksWithoutRecord booksWithoutRecord = new BooksWithoutRecord( title: "Can't hurt me", author: "David Goggins");
    System.out.printf(" Title: %s%n Author: %s", booksWithoutRecord.getTitle(), booksWithoutRecord.getAuthor());
}

public record BooksWithRecord(String title, String author) {
}

public class BooksWithoutRecord {
    private final String title;
    private final String author;

    public BooksWithoutRecord(String title, String author) {
        this.title = title;
        this.author = author;
    }

    public String getTitle() {
        return title;
    }

    public String getAuthor() {
        return author;
    }
}
```

Collection convertor methods

```
@Test
void listSetAndMapOf2() {
    List<String> names = List.of("Sunil", "Simran");
    Set<String> states = Set.of("Maharashtra", "Punjab");
    Map<String, String> personDetails2 = Map.of( k1: "Sunil", v1: "Maharashtra", k2: "Simran", v2: "Punjab");

    states.stream()
        .filter(Predicate.not(s -> s.equals("Punjab")))
        .collect(Collectors.toList());

    states.stream()
        .filter(Predicate.not(s -> s.equals("Punjab")))
        .toList();

}
```

A large, solid orange circle is positioned on the left side of the slide, overlapping the white background.

JAVA 17

Sealed Classes with strict extend rules

Enhanced pseudo-random number generator

Sealed Classes

```
@Test
void defaultInterfaceMethodTest() {
    final Welcome welcome = new InviteGuest();

    Welcome.invite();
    welcome.greet();
    welcome.serve();
}

sealed interface Welcome permits InviteGuest{
    static void invite() { System.out.println("Let's invite guests for the barbecue party"); }

    default void greet() { System.out.println("Hello Everyone!"); }

    void serve();
}

static final class InviteGuest implements Welcome {

    @Override
    public void serve() { System.out.println("Would you like to have some drinks"); }

    @Override
    public void greet() { System.out.println("Hello Java how is world?"); }
}
```

Questions?
