Exam 1Z0-817: Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer Quiz

Mikalai Zaikin

Belarus
Minsk
<NZaikin[at]iba.by>

October 2020

Revision History		
Revision commit fdc5bf26aadbd081f741fe08fb7eecffed07a3e3	Date: Mon Oct 19 01:12:13 2020 +0300	Author: mzaikin <mzaikin@172.16.114.135>
.		
Revision commit 9598fef486886d5b9ff5158335a6bf22fd9c9d3a	Date: Mon Oct 19 01:00:43 2020 +0300	Author: mzaikin <mzaikin@172.16.114.135>
Improved 030103 question wording		
Revision commit 79ae69504168398270b09b56dc0d70e994c126f0	Date: Thu Feb 6 21:15:38 2020 +0300	Author: Mikalai Zaikin <mzaikin@us.ibm.com>
Fixed typo (number of correct options)		

Abstract

The purpose of this document is to help in preparation for Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer Test (Exam 1Z0-817).

This document should not be used as the only study material for Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer Test (Exam 1Z0-817). It might cover not all objective topics, and it might be not enough. I tried to make this document as much accurate as possible, but if you find any error, please let me know.

Preface

If you believe you have found an error in the "Upgrade OCP Java 6, 7 & 8 to Java SE 11 Developer Quiz" or have a suggestion to improve it, please send an e-mail to me. Please, indicate the topic and page URL.

Part I. Exam Objectives

Chapter 1. Understanding Modules

1.1. Describe the Modular JDK

Question 010101

Given the code fragment:

```
try (
    Connection conn = DriverManager.getConnection(...);
    Statement stmt = conn.createStatement();) {
    // ...
}
```

Which two actions when done simultaneously will allow the code compile successfully?

Options (choose 2):

1. Add to the class definition: `import java.sql.*;`

2. Add to the class definition: `requires java.sql.*;`
3. Add to the module definition: `requires java.base;`
4. Add to the module definition: `requires java.se;`
5. Add to the module definition: `open java.sql;`

Question 010102

You are writing a modular application for Java 11 platform which uses SAX XML parser.

From JavaDoc you know that the `javax.xml.parsers.SAXParser` parser class belongs to `java.xml` module.

Which step you must do in order to use this class in your application?

Options (choose 1):

1. Add the option to `java` and `javac`: `-p java.xml`
2. Add the option to `java` and `javac`: `-cp java.xml.mod`
3. Add directive to the module definition: `requires java.xml;`
4. Add directive to the module definition: `requires java.base;`

Question 010103

While compiling your modular application you got the error from `javac`:

```
import java.sql.Connection;
           ^
  (package java.sql is declared in module java.sql, but module modS does not read it)
1 error
```

Which two changes when done independently will resolve this error?

Options (choose 2):

1. Run `javac` with option: `--module-path java.sql`
2. Run `javac` with option: `-modulepath java.sql`
3. Run `javac` with option: `-mp java.sql`
4. Add directive to the module definition: `requires java.sql;`
5. Add directive to the module definition: `requires transitive java.sql;`

Question 010104

Which two statements are correct about Java modules?

Options (choose 2):

1. Automatic module name is defined in `module-info.java` which is compiled later to `module-info.class`
2. The `java.base` standard module exports Java core packages
3. A `module-info.class` automatically found by JVM inside any sub-folder on module path
4. One module automatically can use another module as long as both are listed on module path
5. A `module-info.java` with zero directives is a valid module definition

1.2. Declare modules and enable access between modules

Question 010201

Given the application consisting of 2 modules:

- `modA` module contains `pkgA` package
- `modB` module contains `pkgB` package
- classes in `pkgB` package depend on classes from `pkgA` package

Which two actions when done simultaneously will allow the code to compile successfully?

Options (choose 2):

1. `modA` module must open `pkgA` package

2. `modA` module must export `pkgA` package
3. `modB` module must require `pkgA` package
4. `modB` module must require `modA` module

Question 010202

Given the application consisting of 2 modules:

- `modA` module contains `pkgA` package
- `modB` module contains `pkgB` package
- classes in `pkgB` package depend on classes from `pkgA` package

Which action among other must a developer take in order to run the application successfully?

Options (choose 1):

1. `modA` module must open `pkgA` package
2. `modA` module must require `modB` module
3. `modB` module must import `modA` module
4. `modB` module must open `modA` module

Question 010203

Which statement is true about Java modules?

Options (choose 1):

1. Java module may contain at most 1 package.
2. Java package may be splitted between several modules.
3. Java module must be named as the top level package which the module contains.
4. Java module must contain at least one package.
5. In Java module definition all directives are optional.

1.3. Describe how a modular project is compiled and run

Question 010301

Given:

- An application module `greeter` packaged in the `greeter.jar` file.
- The `p2.Client` class depends on some classes from the `greeter` module.
- Layout of the modular application is as follows:

```
•
••••mod
•       greeter.jar
•
••••src
    •   module-info.java
    •
    ••••p2
            Client.java
```

Which two commands when run independently will compile the `.java` source code?

Options (choose 2):

1. `javac --module-path mod src/module-info.java src/p2/Client.java`
2. `javac --module-path mod/greeter.jar src/module-info.java src/p2/Client.java`
3. `javac -classpath mod src/module-info.java src/p2/Client.java`
4. `javac -cp mod/greeter.jar src/module-info.java src/p2/Client.java`
5. `javac -mp mod/greeter.jar src/module-info.java src/p2/Client.java`

Question 010302

Given:

- An application module `greeter` is packaged in the `greeter.jar` file.
- An application module `client` is packaged in the `client.jar` file.
- The `client` module depends on classes from the `greeter` module.
- The `client` JAR was created by this command:

```
jar --create --file=client.jar --main-class=pkg.Main -C . .
```

- Both JARs are located in the current directory.

Which three commands when run independently will run the `pkg.Main` class?

Options (choose 3):

1. `java -p . -m client`
2. `java --module-path . --module client/pkg.Main`
3. `java -cp greeter.jar;client.jar -m client/pkg.Main`
4. `java -cp greeter.jar;client.jar pkg.Main`
5. `java -cp greeter.jar;client.jar -jar client.jar`

Question 010303

Given:

- An application module `modA` is packaged in the `modA.jar` file.
- An application module `modB` is packaged in the `modB.jar` file.
- An application module `modC` is packaged in the `modC.jar` file.
- All the JAR files are located in the same directory.
- The `modC` module depends on classes from the `modB` module.
- The `modB` module depends on classes from the `modA` module.

Which option can be used to run a class from `modC` module?

Options (choose 1):

1. `--module-path modA.jar;modB.jar;modC.jar` on Windows platform
2. `--module-path modA.jar,modB.jar,modC.jar` on all platforms
3. `--module-path modA:modB:modC` on Linux platform
4. `--module-path modA,modB,modC` on all platforms

Chapter 2. Services in a Modular Application

2.1. Describe the components of Services including directives

Question 020101

Given the modular Service application consisting of three modules:

- The `modI` module contains Service Interface
- The `modP` module contains Service Provider
- The `modC` module contains Service Consumer

What is true about the modular application?

Options (choose 1):

1. The `modI` module definition must have `requires` directive with the service provider module name
2. The `modC` module definition must have `requires` directive with the service provider module name
3. The service implementation class from `modP` module must implement service interface from `modI` module
4. The `modP` module definition must have `exports` directive with the package name of the service implementation class
5. The `modC` module definition must have `requires` directive with the service interface module name

Question 020102

Which two statements are true about modular Service application?

Options (choose 2):

1. Service Provider class must have `public` no-args constructor.
2. Service Provider must be a `public` type.
3. Service Provider must have a `public static provider()` method which returns an instance of the type which is assignable to the Service's type.
4. At runtime Service Locator returns the first provider found on the module path.
5. Services mechanism can be used to break cyclic dependencies.

Question 020103

Which two statements are true about modular Service type?

Options (choose 2):

1. Service type can be an interface.
2. Service type may NOT be an abstract class.
3. Service type can be a concrete class.
4. Service type may NOT be a final class.
5. Service type can be an enum.

2.2. Design a service type, load the services using `ServiceLoader`, check for dependencies of the services including consumer module and provider module

Question 020201

You are developing a modular application which uses Services. There could be hundreds of service providers from different vendors. For performance improving and memory saving reasons you need to analyze vendor's Service Provider class name before creating an instance of the provider. Which service loader method allows you to implement this requirement?

Options (choose 1):

1. `ServiceLoader.findFirst()`
2. `ServiceLoader.iterator()`
3. `ServiceLoader.stream()`
4. `ServiceLoader.load•(...)`

Question 020202

You have developed a `mod.service` Services module with a `pkg.service.MonitoringService` interface.

Now you started working on `mod.provider` Service Provider module with a `pkg.provider.MonitoringServiceImpl` class. Current module definition looks as follows:

```
module mod.provider {
    requires mod.service;
    provides pkg.service.MonitoringService with pkg.provider.MonitoringServiceImpl;
}
```

Which statement is true about `mod.provider` module definition?

Options (choose 1):

1. You need to add `uses pkg.service.MonitoringService;` directive.
2. You need to remove `requires mod.service;` directive.
3. You need to add `exports pkg.provider;` directive.
4. The current module definition is correct and does not require any directive.

Question 020203

Assuming Service, Provider, and Consumer modules are separate. Which two directives are mandatory in the Consumer module definition?

Options (choose 2):

1. `uses <provider module name>;`
2. `uses <service module name>;`
3. `uses <service type name>;`
4. `requires <provider module name>;`

5. `requires <service module name>;`

Question 020204

You are migrating Service application created for Java 8 version to Java 11. You need to integrate old Java 8 based Consumer and Provier JARs in the new modular application. Which two statements are correct about the migration to new Java version?

Options (choose 2):

1. Place the unchanged Consumer JAR file on the module path.
2. You have to add to the Consumer JAR file new module definition with the `uses` directive and put the JAR on the module path.
3. Place the unchanged Provider JAR file on the module path.
4. You have to add to the Provider JAR file new module definition with the `provides` directive and put the JAR on the module path.

Chapter 3. Java Interfaces

3.1. Create and use methods in interfaces

Question 030101

Given the `C1` class:

```
class C1 implements I1, I2 { }
```

Which two `I1` and `I2` pair will compile successfully along with `C1` class?

Options (choose 2):

1.

```
interface I1 {
    public void doIt() { }
}
interface I2 {
    private void doIt() { }
}
```

2.

```
interface I1 {
    default public void doIt() { }
}
interface I2 {
    public static void doIt() { }
}
```

3.

```
interface I1 {
    public static void doIt() { }
}
interface I2 {
    public static void doIt() { }
}
```

4.

```
interface I1 {
    default public void doIt() { }
}
interface I2 {
    default public void doIt() { }
}
```

5.

```
interface I1 {
    private static void doIt() { }
}
interface I2 {
    private abstract void doIt() { }
}
```

Question 030102

Given the `I1` interface:

```
interface I1 {
    public static void doIt() {
        // ...
    }
}
```

Which statement is correct about `doIt()` method?

Options (choose 1):

1. The method can be overriden in the class which implements `I1` interface.
2. The method can be shadowed in the class which implements `I1` interface.
3. The method can only be invoked from `public static` or `private static` methods of `I1` interface.
4. The method can invoke `public static` or `private static` methods of `I1` interface.

Question 030103

Which four method declarations are allowed in Java 11 interfaces?

Options (choose 4):

1. `public abstract void doIt();`
2. `private static void doIt() {}`
3. `private abstract void doIt();`
4. `public void doIt();`
5. `protected void doIt() {}`
6. `private void doIt() {}`

Question 030104

Given the code:

```
class Calc implements Adding {
    public static void main(String[] args) {
        System.out.print(new Calc().sumOfTwo(1, 1));
    }
}
interface Adding {
    public Integer sumOfTwo(Integer a, Integer b) {
        return sum(a, b);
    }

    private Integer sum(Integer... ints) {
```

```
        return Arrays.asList(ints).stream().mapToInt(a->a).sum();
    }
}
```

What is the result?

Options (choose 1):

1. `2`
2. Compilation fails due to `Calc.main(...)` method
3. Compilation fails due to `Adding.sumOfTwo(...)` method
4. Compilation fails due to `Adding.sum(...)` method

Question 030105

Given the code:

```
interface Greeter {
    default public String say() {
        return "Hello";
    }
}
interface Welcomer {
    default public String say() {
        return "Welcome";
    }
}
class Speaker implements Greeter, Welcomer {
    public static void main(String[] args) {
        System.out.println(new Speaker().say());
    }
}
```

What is the result?

Options (choose 1):

1. Code successfully compiles and prints `Hello`.
2. Compilation fails, in order to print `Hello` new method must be added to the `Speaker` class:

    ```
    public String say() {
        return Greeter.say();
    }
    ```

3. Compilation fails, in order to print `Hello` new method must be added to the `Speaker` class:

    ```
    public String say() {
        return Greeter.this.say();
    }
    ```

4. Compilation fails, in order to print `Hello` new method must be added to the `Speaker` class:

    ```
    public String say() {
        return Greeter.super.say();
    }
    ```

3.2. Define and write functional interfaces

Question 030201

Given the two interfaces, which statement is correct:

```
@FunctionalInterface
public interface ParentIntf {
    void doIt();
}
```

```
@FunctionalInterface
public interface ChildIntf extends ParentIntf {

}
```

Options (select 1):

1. Compilation of `ChildIntf` fails. To make the code compile sucessfully, remove `@FunctionalInterface` from `ChildIntf` declaration.
2. Compilation of `ChildIntf` fails. To make the code compile sucessfully, add in `ChildIntf` interface:

```
@FunctionalInterface
public interface ChildIntf extends ParentIntf {
    void doIt();
}
```

3. Compilation of `ChildIntf` fails. To make the code compile sucessfully, add in `ChildIntf` interface:

```
@FunctionalInterface
public interface ChildIntf extends ParentIntf {
    void doItNow();
}
```

4. The code compiles successfully.

Question 030202

Given the two interfaces, which three statements are correct:

```
@FunctionalInterface
public interface ParentIntf {
        void doIt();
}
```

```
@FunctionalInterface
public interface ChildIntf extends ParentIntf {
        void doItNow();
}
```

Options (select 3):

1. Compilation fails. To make the code compile sucessfully remove `@FunctionalInterface` annotation from the `ChildIntf` interface:

```
public interface ChildIntf extends ParentIntf {
    void doItNow();
}
```

2. Compilation fails. To make the code compile sucessfully, modify the `ChildIntf` interface method signature:

```
@FunctionalInterface
public interface ChildIntf extends ParentIntf {
    void doIt();
}
```

3. Compilation fails. To make the code compile sucessfully remove `@FunctionalInterface` annotation from the `ParentIntf` interface:

```
public interface ParentIntf {
    void doIt();
}
```

4. Compilation fails. To make the code compile sucessfully, modify the `ChildIntf` declaration:

```
@FunctionalInterface
public interface ChildIntf {
    void doItNow();
}
```

Question 030203

Which two statements are correct about Java functional interfaces?

Options (select 2):

1. A functional interface must be annotated with `@FunctionalInterface`.
2. A functional interface may contain one or more `abstract` methods.
3. A functional interface may contain one or more `default` methods.
4. A functional interface may contain one or more `static` methods.

Question 030204

Given the code:

```
interface CPU1 {
    int doAdd(); // line 1
    default int doSub() { return 1 - 1; }
}

interface CPU2 {
    default int doAdd() {return 1 + 1;} // line 2
    String toString();
}

public class Calculator {

    CPU1 c1 = () -> { return 1 + 2; }; // line 3
    CPU2 c2 = () -> { return 1 + 3; }; // line 4

    void calculate() {
        System.out.print(c1.doAdd());
        System.out.print(c2.doAdd());
    }

    public static void main(String[] args) {
        new Calculator().calculate();
    }
}
```

What is the result?

Options (select 1):

1. Compilation fails at `line 1`.
2. Compilation fails at `line 2`.
3. Compilation fails at `line 3`.
4. Compilation fails at `line 4`.
5. The code prints `32`.
6. The code prints `34`.

Question 030205

Given the `Event` interface:

```
interface Event {
    String toString();
}
```

and the code:

```
Event e = () -> { return "Mouse"; };
System.out.println(e);
System.out.println(e.toString());
```

What is the result?

Options (select 1):

1. A hashcode of `e` object, followed by `Mouse`.
2. A blank line, followed by `Mouse`.
3.

   ```
   Mouse
   Mouse
   ```

4. Compilation fails.

Chapter 4. Lambda Operations on Streams

4.1. Extract stream data using `map`, `peek`, and `flatMap` methods

Question 040101

What two statements are true about stream pipelines in Java?

Options (choose 2):

1. A pipeline may have one or more sources.
2. A pipeline may consist of a source followed by a terminal operation.
3. A pipeline must have one or more intermediate operations.
4. A pipeline must have a terminal operation.

Question 040102

Given the code:

```
List<String> list = List.of("one", "two", "three");
long l = list.stream().forEach(s -> System.out.println(s)).filter(s ->
s.startsWith("t")).count();
System.out.println(l);
```

```

```

What is the result?

Options (choose 1):

1.

```
two
three
2
```

2.

```
one
two
three
2
```

3.

```
one
two
three
```

   followed by Runtime Exception.
4. Compilation fails.

Question 040103

Given the code:

```
List<Integer> nums = List.of(1, 2, 3);
Stream<Integer> stream = nums.stream();
stream.map(i -> i*2).forEach(i -> System.out.printf("%d ", i));
stream.forEach(i -> System.out.printf("%d ", i));
```

What is the result?

Options (choose 1):

1.

```
2 4 6 2 4 6
```

2.

```
2 4 6 1 2 3
```

3.

```
2 4 6
```

followed by Runtime Exception.
4. Compilation fails.

Question 040104

Given the code fragment:

```
List<String> words = List.of("One", "Two", "Three");
Stream<String> s = words.stream().map(w -> w.toLowerCase());
List<String> list = s.collect(Collectors.toList());
System.out.print(list);
```

What is the output?

Options (choose 1):

1. `[one]`
2. `[One]`
3. `[one, two, three]`
4. `[One, Two, Three]`
5. Blank output.

Question 040105

Given the `Employee` class:

```
public class Employee {
    private String name;
    private int age;
    public Employee(String n, int i) {
        name = n;
        age = i;
    }
    public String getName() {
        return name;
    }
    public int getAge() {
        return age;
    }
}
```

and the code fragment:

```
Stream<Employee> emps = Stream.of(new Employee("John", 53), new Employee("Jane", 46), new
Employee("Deb", 45));
... // line n1
System.out.print(averageAge);
```

Which line inserted at `line n1` position will print average age of all employees?

Options (choose 1):

1. `double averageAge = emps.map(e -> e.getAge()).average().getAsDouble();`
2. `double averageAge = emps.mapToInt(e -> e.getAge()).average().getAsDouble();`
3. `double averageAge = emps.map(e -> e).average().getAsDouble();`
4. `double averageAge = emps.mapToInt(e -> e).average().getAsDouble();`

Question 040106

Given the code fragment and assuming that `salaries` is a properly populated list of `Double` values:

```
List<Double> salaries = ... // properly populated list
... // line n1
System.out.print(minSalary);
```

Which two code fragments when inserted independently at `line n1` position will print the lowest salary?

Options (choose 2):

1. `double minSalary = salaries.stream().mapToDouble(Double::doubleValue).min().getAsDouble();`
2. `double minSalary = salaries.stream().mapToDouble((double d) -> d).min().getAsDouble();`
3. `double minSalary = salaries.stream().mapToDouble(d -> d).min().getAsDouble();`
4. `double minSalary = salaries.stream().map(d -> (double)d).min().getAsDouble();`
5. `double minSalary = salaries.stream().map(d -> d).min().getAsDouble();`

Question 040107

Given the `Employee` class:

```
public class Employee {
    public static final int MANAGER=100;
    public int type;
    public String name;
    public double salary;

    public Employee(int t, String n, double s) {
        type = t;
        name = n;
        salary = s;
    }
}
```

Assuming that `emps` variable refers to properly declared and initialized `List` of `Employee`s, which line of code will help to find manager's highest salary?

Options (choose 1):

1. `double maxSalary = emps.filter(a -> a.type == Employee.MANAGER).max().getAsDouble();`
2. `double maxSalary = emps.filter(a -> a.type == Employee.MANAGER).map(a -> a.salary).max().getAsDouble();`
3. `double maxSalary = emps.filter(a -> a.type == Employee.MANAGER).mapToDouble(a -> a.salary).max().getAsDouble();`
4. `double maxSalary = emps.filter(a -> a.type == Employee.MANAGER).mapToDouble(a -> a).max().getAsDouble();`

Question 040108

Given the code fragment:

```
List<String> names1 = List.of("Dzmitry", "John");
List<String> names2 = List.of("Mikalai");
List<String> names3 = List.of("David", "Laura");
Stream<List<String>> s = Stream.of(names1, names2, names3);
s.flatMap(names -> names.stream()).forEach(System.out::println);
```

What is the output?

Options (choose 1):

```
Dzmitry
John
```

2.

```
David
Laura
```

3.

```
Dzmitry
John
Mikalai
David
Laura
```

4.

```
David
Dzmitry
John
Laura
Mikalai
```

4.2. Search stream data using search `findFirst`, `findAny`, `anyMatch`, `allMatch` and `noneMatch` methods

Question 040201

Given the code fragment:

```
Stream<String> txt = Stream.of("Once", "One", "Other");
System.out.print(txt.filter(w -> w.startsWith("O")).findAny().get());
```

What is the output?

Options (choose 1):

1. `Once`
2. `OnceOneOther`
3. Either `Once`, or `One`, or `Other`, chosen randomly.
4. All three: `Once`, and `One`, and `Other` in no particular order.

Question 040202

Given the code fragment:

```
Stream<String> txt = Stream.of("Code", "And", "Code");
txt.allMatch(w -> w.equals("Code")).forEach(System.out::print); // line n1
```

What is the result?

Options (choose 1):

1. `CodeAndCode`
2. `CodeCode`
3. `Code`
4. Compilation fails at `line n1`.

Question 040203

Given the `Employee` class:

```
public class Employee {
    public static final int MANAGER=100;
    public int type;
    public String name;

    public Employee(int t, String n) {
        type = t;
        name = n;
    }
}
```

and the following code fragment:

```
List<Employee> emps = List.of(new Employee(100, "Mike"), new Employee(99, "John"), new
Employee(100, "Deb"));
boolean b1 = emps.stream().map(e -> e.type).allMatch(t -> t == Employee.MANAGER);
boolean b2 = emps.stream().filter(e -> !(e.type == Employee.MANAGER)).anyMatch(m -> m.type
== Employee.MANAGER); // line n1
System.out.print(b1 + " " + b2);
```

What is the result?

Options (choose 1):

1. `true true`
2. `true false`
3. `false true`
4. `false false`
5. Compilation fails at `line n1`.

Question 040204

Given the following code fragment:

```
List<String> list = List.of("A", "B", "C", "A", "A", "B");
... // line n1
System.out.print(v);
```

What line inserted at `line n1` position will print `true`?

Options (choose 1):

1. `boolean v = list.stream().allMatch(a -> "A".equals(a));`
2. `boolean v = list.stream().noneMatch(a -> "A".equals(a));`
3. `boolean v = list.stream().map(a -> "A").noneMatch(a -> "A".equals(a));`
4. `boolean v = list.stream().filter(a -> "A".equals(a)).allMatch(a -> "A".equals(a));`
5. `boolean v = list.stream().filter(a -> "A".equals(a)).noneMatch(a -> "A".equals(a));`

Question 040205

Given the following code fragment:

```
List<Integer> ints = List.of(10, 20, 30);
Optional<Integer> o = ints.stream().anyMatch(i -> i > 15); // line n1
System.out.print(o.get());
```

What is the result?

Options (choose 1):

1. `20`
2. `30`
3. Either `20`, or `30`, randomly chosen.
4. Compilation fails at `line n1`.

Question 040206

Given the code fragment:

```
Integer i1 = Stream.of(1,2).findAny().get();
Integer i2 = Stream.of(1,2).parallel().findAny().get();
```

What statement is correct about the assignments?

Options (choose 1):

1. `i1` is always `1`, and `i2` is always `1`
2. `i1` is always `1`, and `i2` can be `1` or `2`
3. `i1` is always `1`, and `i2` is always `2`
4. Both `i1` and `i2` can be `1` or `2`

4.3. Use the `Optional` class

Question 040301

Given the following code fragment:

```
 1 Integer i = null;
 2 Optional<Integer> o = Optional.of(i);
 3 System.out.println(o.isPresent());
 4 Integer ii = o.get();
 5 System.out.println(ii.intValue());
```

What is the result?

Options (choose 1):

1. Runtime exception at line 2.
2. Runtime exception at line 4.
3. Runtime exception at line 5.
4. Compilation fails.

Question 040302

Given the following code fragment:

```
String s1 = "ABC";
String s2 = null;
Optional<String> o1 = Optional.ofNullable(s1);
```

```
Optional<String> o2 = Optional.ofNullable(s2);
System.out.println(o1.isPresent());
System.out.println(o2.isPresent());
System.out.println(o1.get());
System.out.println(o2.get());
```

What is the result?

Options (choose 1):

1.

```
true
true
ABC
null
```

2.

```
true
false
ABC
null
```

3.

```
true
true
ABC
```

Followed by runtime exception.

4.

```
true
false
ABC
```

Followed by runtime exception.

5.

```
true
```

Followed by runtime exception.

Question 040303

Given the code:

```
var o = Optional.ofNullable(1);
o = o.or(() -> Optional.ofNullable(2));
var i = o.orElseGet(() -> 3);
System.out.print(i);
```

What is the output?

Options (choose 1):

1. `1`
2. `2`
3. `3`
4. Blank output.

4.4. Perform calculations using `count`, `max`, `min`, `average` and `sum` stream operations

Question 040401

Given the following code fragment:

```
List<Integer> ints = List.of(2, 4, 6, 5, 8);
Stream<Integer> stream = ints.stream();
... // line n1
```

Which code inserted at `line n1` position will calculate average value of numbers from the list?

Options (choose 1):

1. `Optional<Double> average = stream.average().get();`
2. `Double average = stream.average().get();`
3. `double average = stream.map(d -> d).average().getAsDouble();`
4. `double average = stream.mapToInt(d -> d).average().getAsDouble();`

Question 040402

Given the following code fragment:

```
List<Integer> ints = List.of(2, 4, 6, -1, 5, 8);
Stream<Integer> stream = ints.stream();
... // line n1
```

Which two code fragments inserted independently at `line n1` position will print the minimum value contained by the list?

Options (choose 2):

1. `System.out.print(stream.min(Integer::min).get());`
2. `System.out.print(stream.min(Integer::max).get());`
3. `System.out.print(stream.min(Integer::compare).get());`
4. `System.out.print(stream.map(i -> i).min().get());`
5. `System.out.print(stream.mapToInt(i -> i).min().getAsInt());`

Question 040403

Given the `Employee` class:

```
public class Employee {
    public String name;
    public int salary;
    public Employee(String n, int s) {
        name = n;
        salary = s;
    }
}
```

and following code fragment:

```
Stream<Employee> stream = ... // assume stream is properly initialized by valid Employee
objects
... // line n1
```

Which code inserted at `line n1` position will print the minimum salary of the employees?

Options (select 1):

1. `System.out.print(stream.min((a, b) -> a.salary - b.salary).get().salary);`
2. `System.out.print(stream.min((a, b) -> a.salary - b.salary).get().getAsInt());`
3. `System.out.print(stream.min(a -> a.salary).get().getAsInt());`
4. `System.out.print(stream.min(Integer::compare).get());`

4.5. Sort a collection using lambda expressions

Question 040501

Managers of your company want to make a bonus payment to an employee who (1) has the longest record of service and (2) has the lowest base salary.

Given the `Employee` class:

```
public class Employee {
    public String name;
    public int baseSalary;      // in USD
    public int recordOfService; // in years
    public Employee(String n, int s, int r) {
        name = n;
        baseSalary = s;
        recordOfService = r;
    }
}
```

and the code fragment:

```
Stream<Employee> emps = // assume stream is properly initialized by valid Employee objects

Comparator<Employee> byRoS = (e1, e2) -> e1.recordOfService - e2.recordOfService;
Comparator<Employee> bySal = (e1, e2) -> e1.baseSalary - e2.baseSalary;

... // line n1
```

Which code inserted at `line n1` position will help you to find the required employee?

Options (choose 1):

1. `Employee e = emps.sorted(byRoS).reversed().sorted(bySal).findFirst().get();`
2. `Employee e = emps.sorted(byRoS).reversed().thenComparing(bySal).findFirst().get();`
3. `Employee e = emps.sorted(byRoS.reversed().thenComparing(bySal)).findFirst().get();`
4. `Employee e = emps.sorted(byRoS.reversed()).thenComparing(bySal).findFirst().get();`

Question 040502

Given the `Employee` class:

```
public class Employee {
    public String name;
    public Employee(String n) {
        name = n;
```

```
    }
    @Override
    public boolean equals(Object o) {
        return name.equals(o);
    }
    @Override
    public int hashCode() {
        return name.hashCode();
    }
}
```

and the code fragment:

```
Stream<Employee> emps = Stream.of(new Employee("Steve"), new Employee("Nick"), new
Employee("Jack"));
emps.sorted().forEach(a -> System.out.print(a.name + " ")); // line n1
```

What is the result?

Options (choose 1):

1. `Steve Nick Jack`
2. `Jack Nick Steve`
3. Compilation fails at `line n1`.
4. Runtime exception.

Question 040503

Given the code:

```
List<String> list = Arrays.asList("C101", "B12", "A1", "C102");
list.stream()
    .sorted(Comparator.reverseOrder())
    .sorted((a, b) -> a.length() - b.length())
    .collect(Collectors.toList());
list.forEach(System.out::print);
```

What is the output?

Options (choose 1):

1. `C101B12A1C102`
2. `A1B12C102C101`
3. `C102C101B12A1`
4. `A1B12C101C102`

4.6. Use `Collectors` with streams, including the `groupingBy` and `partitioningBy` operation

Question 040601

Given the `Employee` class:

```
public class Employee {
    private String name;
    private String department;

    public Employee(String n, String d) {
        name = n;
        department = d;
    }
```

```
    public String getDepartment() {
        return department;
    }

    @Override
    public String toString() {
        return name;
    }
}
```

and the following code fragment:

```
Employee e1 = new Employee("Mikalai", "Development");
Employee e2 = new Employee("Volha", "HR");
Employee e3 = new Employee("Anastasia", "Management");
Employee e4 = new Employee("Daria", "Management");
Employee e5 = new Employee("Ivan", "Management");
Stream<Employee> str = Stream.of(e1, e2, e3, e4, e5);
str.collect(Collectors.groupingBy(Employee::getDepartment)).forEach((a, b) ->
System.out.println(b)); // line n1
```

What is the result?

Options (choose 1):

1.
```
[Mikalai, Volha, Anastasia, Daria, Ivan]
```

2.
```
[Mikalai]
[Volha]
[Anastasia, Daria, Ivan]
```

3.
```
[Mikalai]
[Volha]
[Anastasia]
[Daria]
[Ivan]
```

4. Compilation fails at `line n1`.

Question 040602

Given the `Employee` class:

```
public class Employee {
    private String name;
    private int salary;

    public Employee(String n, int s) {
        name = n;
        salary = s;
    }
```

```
    public int getSalary() {
        return salary;
    }

    @Override
    public String toString() {
        return name;
    }
}
```

and the code fragment:

```
Employee e1 = new Employee("John", 98);
Employee e2 = new Employee("Rosie", 102);
Employee e3 = new Employee("Steven", 70);
Stream<Employee> str = Stream.of(e1, e2, e3);
System.out.print(str.collect(Collectors.partitioningBy(e -> e.getSalary() <
100)).get(e3.getSalary() > 100));
```

What is the result?

Options (choose 1):

1.
```
[John]
```

2.
```
[Rosie]
```

3.
```
[Steven]
```

4.
```
[John, Steven]
```

Question 040603

Given the `Employee` class:

```
public class Employee {
    private String name;
    private String department;

    public Employee(String n, String d) {
        name = n;
        department = d;
    }

    public String getDepartment() {
        return department;
    }
}
```

and the code fragment:

```
Employee e1 = new Employee("Mikalai", "Development");
Employee e2 = new Employee("Volha", "HR");
Employee e3 = new Employee("Anastasia", "Management");
Employee e4 = new Employee("Daria", "Management");
Employee e5 = new Employee("Ivan", "Management");
Stream<Employee> str = Stream.of(e1, e2, e3, e4, e5);
... // line n1
```

Which two codes when inserted independently at `line n1` position will extract all distinct departments?

Options (choose 2):

1. `List l = str.distinct().collect(Collectors.toList());`
2. `Collection l = str.map(Employee::getDepartment).collect(Collectors.toSet());`
3. `List l = str.map(e -> e.getDepartment()).distinct().collect(Collectors.toList());`
4. `String l = str.distinct().map(e -> e.getDepartment()).collect(Collectors.joining(", "));`

Question 040604

Given the code fragment:

```
Stream<String> str = Stream.of("Hi", "No", "Hey", "Yes", "Wow");
System.out.print(str.collect(Collectors.groupingBy(a -> a.length(),
Collectors.counting())).get(2)); // line n1
```

What is the result?

Options (choose 1):

1. `Hey`
2. `2`
3. `null`
4. Compilation error at `line n1`.

Chapter 5. Java File I/O (NIO.2)

5.1. Use `Path` interface to operate on file and directory paths

Question 050101

Given the code fragment:

```
Path p1 = Paths.get("app");
Path p2 = p1.getRoot();      // line n1
Path p3 = p2.resolve("lib"); // line n2
System.out.print(p3);
```

and assuming the following directories exist on filesystem:

```
C:\
••••app
••••lib
```

What is the result if the code run from working directory `C:\`?

Options (choose 1):

1. `lib`
2. `C:\lib`
3. Runtime exception at `line n1`.
4. Runtime exception at `line n2`.

Question 050102

Given the class running on Unix-like platform:

```
import java.nio.file.Path;

public class PathTest {
    public static void main(String[] args) {
        Path pm = Path.of("/home/mikalai");
        Path pv = Path.of("/home/volha");
        Path res = pm.resolve(pv);
        System.out.print(res);
    }
}
```

What is the output?

Options (choose 1):

1. `/home/mikalai`
2. `/home/volha`
3. `/home/volha/home/mikalai`
4. `/home/mikalai/home/volha`

Question 050103

Given the code:

```
... // line n1
... // line n2
System.out.print(p);
```

Which two changes when done simultaneously will make the code to print `home`?

Options (choose 2):

1. `Path pm = Path.of("/home/mikalai"); // line n1`
2. `Path pm = Path.of("home/mikalai"); // line n1`
3. `Path p = pm.getParent(); // line n2`
4. `Path p = pm.getRoot(); // line n2`

5.2. Use `Files` class to check, delete, copy or move a file or directory

Question 050201

Given the filesystem structure:

```
/tmp
••••a
•   •••• file.txt
•
••••b
    •••• log.txt
```

and the code fragment:

```
Path f1 = Path.of("/tmp/a/file.txt");
Path f2 = Path.of("/tmp/b");
Files.move(f1, f2, StandardCopyOption.ATOMIC_MOVE, StandardCopyOption.REPLACE_EXISTING);
```

Which are results?

Options (choose 2):

1. The file `/tmp/a/file.txt` is renamed to `/tmp/b/file.txt`
2. The file `/tmp/a/file.txt` is deleted.
3. The file `/tmp/a/file.txt` is not deleted.
4. Exception is thrown at runtime.
5. State of the files is undefined after code run.

Question 050202

Given the filesystem structure:

```
/tmp
••••a
     •••• file.txt
```

and the code fragment:

```
Path f1 = Path.of("/tmp/a/file.txt");
Path f2 = Path.of("/tmp/b");
Files.move(f1, f2);
Files.delete(f1);
```

Which are results?

Options (choose 2):

1. The file `/tmp/a/file.txt` is deleted.
2. The file `/tmp/a/file.txt` is not deleted.
3. Exception is thrown at runtime.
4. Code runs silently without exception.

Question 050203

Given the filesystem structure:

```
/tmp
••••a
•    •••• file.txt
•
••••b
    •••• log.txt
```

and the code fragment:

```
Path f1 = Path.of("/tmp/a/file.txt");
Path f2 = Path.of("/tmp/b/log.txt");
... // line n1
```

Which code added at `line n1` position will replace `log.txt` with `file.txt`?

Options (choose 1):

1. `Files.move(f1, f2);`
2. `Files.move(f1, f2, StandardCopyOption.REPLACE_EXISTING, StandardCopyOption.COPY_ATTRIBUTES);`
3. `Files.move(f1, f2, StandardCopyOption.REPLACE_EXISTING, StandardCopyOption.ATOMIC_MOVE);`
4. `Files.move(f1, f2, StandardOption.REPLACE_EXISTING);`

Question 050204

Given the filesystem structure:

```
/tmp
••••a
    •••• file.txt
```

and the code fragment:

```
Path f1 = Path.of("/tmp/a/file.txt");
... // line n1
```

Which code fragment added at `line n1` position will copy `file.txt` file to the `/tmp` directory?

Options (choose 1):

1.
```
Path f2 = Path.of("/tmp");
Files.copy(f1, f2, StandardCopyOption.REPLACE_EXISTING);
```

2.
```
Path f2 = Paths.get("/tmp", "file.txt");
Files.copy(f1, f2, StandardCopyOption.ATOMIC_MOVE);
```

3.
```
Path f2 = Path.of("/tmp", f1.getFileName().toString());
Files.copy(f1, f2);
```

4.
```
Path f2 = Path.of(f1.getRoot().toString(), "file.txt");
Files.copy(f1, f2);
```

Question 050205

Given the filesystem structure:

```
/tmp
••••a
    •••• file.txt
```

and the code fragment:

```
Stream<Path> files = Files.walk(Paths.get("/tmp"), 2);
files.forEach (n -> {
    System.out.println(n);
});
```

Wha is the output?

Options (choose 1):

1.

```
/tmp
/tmp/a
```

2.

```
/tmp/a
/tmp/a/file.txt
```

3.

```
/tmp
/tmp/a
/tmp/a/file.txt
```

4.

```
a
a/file.txt
```

5.3. Use Stream API with `Files`

Question 050301

Given the following folders structure:

```
C:\
 |
 tmp
    |-README.TXT
    |-code
        |-A.java
```

Which code fragment will print:

```
A.java
README.txt
```

Options (choose 1):

1.

```
Path f = Paths.get("C:\\tmp");
Stream<Path> stream = Files.find(f, 3, (a, b) -> b.isRegularFile());
stream.sorted().forEach(System.out::println);
```

2.

```
Path f = Paths.get("C:\\tmp");
Stream<Path> stream = Files.find(f, 3, (a, b) -> b.isDirectory());
stream.sorted().forEach(s -> System.out.println(s.getFileName()));
```

3.

```
Path f = Paths.get("C:\\tmp");
Stream<Path> stream = Files.find(f, 3, (a, b) -> b.isRegularFile());
stream.sorted().forEach(s -> System.out.println(s.getFileName()));
```

4.

```
Path f = Paths.get("C:\\tmp");
Stream<Path> stream = Files.find(f, 3, (a, b) -> b.isRegularFile());
stream.sorted().forEach(s -> System.out.println(s));
```

Question 050302

Given the following folders structure:

```
C:\
 |
 tmp
    |-README.TXT
    |-code
        |-A.java
```

and the following code fragment:

```
Path f = Paths.get("C:\\tmp");
Stream<Path> stream = Files.walk(f);
stream.forEach(s -> System.out.println(s));
```

What is the output?

Options (choose 1):

1.

```
C:\
C:\tmp
C:\tmp\code
```

```
C:\tmp\code\A.java
C:\tmp\README.txt
```

2.

```
C:\tmp
C:\tmp\code
C:\tmp\code\A.java
C:\tmp\README.txt
```

3.

```
C:\tmp
C:\tmp\code
```

4.

```
C:\tmp\code\A.java
C:\tmp\README.txt
```

Question 050303

Assuming that `f` variable is properly intialized and points to some Java class source code.

```
Path f = Paths.get(...);
```

Which two code fragments will print all classes/packages imported by that Java class?

Options (choose 2):

1.

```
Stream<String> stream = Files.lines(f);
stream.filter(s -> s.startsWith("import")).forEach(s ->
System.out.println(s.substring("import".length() + 1, s.length() - 1)));
```

2.

```
List<String> list = Files.lines(f);
list.stream().filter(s -> s.startsWith("import")).forEach(s ->
System.out.println(s.substring("import".length() + 1, s.length() - 1)));
```

3.

```
Stream<String> stream = Files.readAllLines(f);
stream.filter(s -> s.startsWith("import")).forEach(s ->
System.out.println(s.substring("import".length() + 1, s.length() - 1)));
```

4.

```
List<String> list = Files.readAllLines(f);
list.stream().filter(s -> s.startsWith("import")).forEach(s ->
System.out.println(s.substring("import".length() + 1, s.length() - 1)));
```

Chapter 6. Migration to a Modular Application

6.1. Migrate the application developed using a Java version prior to SE 9 to SE 11 including top-down and bottom-up migration, splitting a Java SE 8 application into modules for migration

Question 060101

You need to split monolithic Java 8 application into modules to create Java 11 modular application. The original JAR file contains the following types:

```
application.jar
••••coffee
•       Arabica.java
•       Coffee.java
•       CoffeeMachine.java
•       Robusta.java
•       SteamCoffeeMachine.java
•
••••drink
        Drink.java
        Machine.java
```

Which will be the best design to split classes?

Options (choose 1):

1.

```
coffee.jar
••••coffee
        Arabica.java
        Coffee.java
        CoffeeMachine.java
        Robusta.java
        SteamCoffeeMachine.java
```

```
drink.jar
••••drink
        Drink.java
        Machine.java
```

2.

```
coffee-base.jar
••••coffee
        Coffee.java
        CoffeeMachine.java
        SteamCoffeeMachine.java
```

```
coffee-types.jar
••••coffee
        Arabica.java
```

```
        Robusta.java
```

```
drink.jar
••••drink
        Drink.java
        Machine.java
```

3.

```
coffee-drink.jar
••••coffee
        Coffee.java
        Arabica.java
        Robusta.java
```

```
coffee-machine.jar
••••coffee
        CoffeeMachine.java
        SteamCoffeeMachine.java
```

```
drink.jar
••••drink
        Drink.java
        Machine.java
```

4.

```
drink.jar
••••coffee
•       Arabica.java
•       Coffee.java
•       Robusta.java
•
••••drink
        Drink.java
```

```
machine.jar
••••coffee
•       CoffeeMachine.java
•       SteamCoffeeMachine.java
•
••••drink
        Machine.java
```

Question 060102

You need to migrate a Java 8 application to Java 11 modular application using "top down" method. The application consists of three JARs: `app.jar`, `lib1.jar`, and `lib2.jar`. Classes in the `app.jar` depend on classes in `lib1.jar` and `lib2.jar`.

Which command will be used to run the application after migration?

Options (choose 1):

1.

```
java --module-path ./app.jar;./lib1.jar;./lib2.jar -m app/by.boot.java.App
```

2.

```
java -cp ./app.jar;./lib1.jar;./lib2.jar by.boot.java.App
```

3.

```
java --module-path ./app.jar -cp ./lib1.jar;./lib2.jar -m app/by.boot.java.App
```

4.

```
java -cp ./app.jar --module-path ./lib1.jar;./lib2.jar by.boot.java.App
```

Question 060103

You need to migrate a Java 8 application to Java 11 modular application using "bottom up" method. The application consists of three JARs: `app.jar`, `lib1.jar`, and `lib2.jar`. Classes in the `app.jar` depend on classes in `lib1.jar` and `lib2.jar`.

Which two commands will be used to run the application after migration?

Options (choose 2):

1.

```
java --module-path ./app.jar;./lib1.jar;./lib2.jar -module app/by.boot.java.App
```

2.

```
java -cp ./app.jar;./lib1.jar;./lib2.jar by.boot.java.App
```

3.

```
java --module-path ./app.jar -cp ./lib1.jar;./lib2.jar -add-modules app -module
app/by.boot.java.App
```

4.

```
java -cp ./app.jar --module-path ./lib1.jar;./lib2.jar --add-modules lib1,lib2
by.boot.java.App
```

6.2. Use `jdeps` to determine dependencies and identify way to address the cyclic dependencies

Question 060201

As a first step to migrating application to Java 11, you need to analyze Java 8 JAR file and determine which class (if any) misuses Java API.

Which command will you use?

Options (choose 1):

1.

```
jdeps -s non-modular.jar
```

2.

```
jdeps --list-deps non-modular.jar
```

3.

```
jdeps -jdkinternals non-modular.jar
```

4.

```
jdeps --generate-module-info . non-modular.jar
```

Question 060202

You want to automate Java 8 JAR migration to a Java 11 module and generate module definition by using `jdeps` utility. The JAR file classes have dependency injection annotations which are processed by application framework and values are injected into classes with help of Reflection API.

Which command will you use?

Options (choose 1):

1.

```
jdeps -dotoutput . non-modular.jar
```

2.

```
jdeps --generate-open-module . non-modular.jar
```

3.

```
jdeps --check module-name=non.modular --module-path non-modular.jar
```

4.

```
jdeps --generate-module-info . non-modular.jar
```

Question 060203

A modular Java 11 application has `modA` with module definition:

```
module modA {
    requires modB;
    exports pkgA;
}
```

amd `modB`:

```
module modB {
    requires modA;
    exports pkgB;
}
```

What will help to compile the application?

Options (choose 1):

1. Use `-Xlint:circular` command line option for `javac`
2. Use `-Xlint:module` command line option for `javac`
3. Use `-Xlint:none` command line option for `javac`
4. Create a new module `modC` that both `modA` and `modB` are dependent, remove `requires modB;` from `modA` definition.

Chapter 7. Local-Variable Type Inference

7.1. Use local-variable type inference

Question 070101

Which statement is true about `var` reserved type name?

Options (select 1):

1. `var` varialbes are effectively final
2. `var` variables are dynamically typed
3. Public `var` varialbes are inherited by subclasses
4. You can use `var` variables with diamond operator on the right-hand side

Question 070102

Given:

```
class A {}
class B extends A {}
```

Which code fragment will FAIL to compile?

Options (select 1):

1.
```
{
    var a = new A();
    a = new B();
}
```

2.
```
{
    var b = new B();
    b = new B();
}
```

3.
```
{
    var c = new B() {};
    c = new B() {};
}
```

4.

```
{
    var d = (A) null;
    d = (B) null;
}
```

Question 070103

Which two classes will compile successfully?

Options (select 2):

1.

```
class A {
    var a = 1;
}
```

2.

```
class B {
    static {
        var b = 1;
    }
}
```

3.

```
class C {
    {
        var c = 1;
    }
}
```

4.

```
class D {
    {
        var d = {1,2};
    }
}
```

5.

```
class E {
    {
        var[] e = new int[] {1,2};
    }
}
```

Question 070104

Given the classes:

```
1 class User {
2     String name;
3 }
4 public class UserDB {
```

```
 5     var list = new ArrayList<User>();
 6     public var displayUsers() {
 7         var list = new ArrayList<>();
 8         var user = new User();
 9         user.name = "Mikalai";
10         list.add(user);
11         for (var u : list) {
12             System.out.println(u.name);
13         }
14         return 0;
15     }
16 }
```

Which three lines FAIL to compile?

Options (select 3):

1. 5
2. 6
3. 7
4. 8
5. 11
6. 12

7.2. Create and use lambda expressions with local-variable type inferred parameters

Question 070201

You are writing a pipeline which processes stream of strings. Application uses a framework with constraint validator. Which code properly assures that stream has no `null` values?

Options (select 1):

1.
```
list.stream().map(@NonNull s -> s.toUpperCase()).collect(Collectors.toList());
```

2.
```
list.stream().map((@NonNull s) -> s.toUpperCase()).collect(Collectors.toList());
```

3.
```
list.stream().map((@NonNull var s) -> s.toUpperCase()).collect(Collectors.toList());
```

4.
```
list.stream().map((@NotNull final v) -> v.toUpperCase()).collect(Collectors.toList());
```

Question 070202

Given the code:

```
List<String> list = List.of("B", "C", "A");
BiFunction<String, String, String> bf = (var a, b) -> a + b;
Function<String, String> f = c -> bf.apply(c, c);
list.stream().map(f).sorted().forEach(System.out::print);
```

What is the result?

Options (select 1):

1.

```
AABBCC
```

2.

```
BBCCAA
```

3.

```
CCBBAA
```

4. Compilation fails

Question 070203

You need to write a `BiFunction` which converts first string parameter to uppercase, second string parameter to lowercase and return concatenated string.

To ensure that `null` values handled properly, you created new annotation `@StringParam(default="default value")` which annotates lambda parameter and provides default value if any passed in parameter is `null`.

Which two are the correct syntax to declare the `BiFunction`?

Options (select 2):

1.

```
BiFunction<String, String, String> bf =
        (@StringParam(default="A") a, @StringParam(default="B") b) ->
        a.toUpperCase() + b.toLowerCase();
```

2.

```
BiFunction<String, String, String> bf =
        (@StringParam(default="A") var a, @StringParam(default="B") var b) ->
        a.toUpperCase() + b.toLowerCase();
```

3.

```
BiFunction<String, String, String> bf =
        (@StringParam(default="A") String a, @StringParam(default="B") String b) ->
        a.toUpperCase() + b.toLowerCase();
```

4.

```
BiFunction<String, String, String> bf =
        (@StringParam(default="A") var a, @StringParam(default="B") String b) ->
        a.toUpperCase() + b.toLowerCase();
```

Chapter 8. Lambda Expressions

8.1. Create and use lambda expressions

Question 080101

Given the code:

```
 1 interface Validator {
 2     boolean validate();
 3 }
 4
 5 public class Vehicle {
 6
 7     boolean startEngine(Driver driver) {
 8         Validator v = new Validator() {
 9             @Override
10             public boolean validate() {
11                 return !driver.isDrunk();
12             }
13         };
14         return v.validate();
15     }
16 }
```

Which code correctly replaces inner local anonymous class defined on lines 8-13 with lambda expression?

Options (select 1):

1.
```
Validator v = (Driver driver) -> { return !driver.isDrunk(); };
```

2.
```
Validator v = new Driver() -> { return !driver.isDrunk(); };
```

3.
```
Validator v = (driver) -> { return !driver.isDrunk(); };
```

4.
```
Validator v = () -> { return !driver.isDrunk(); };
```

Question 080102

Oracle University wants you to sort all its students based on their height. Current code is shown below

```
public class Student {
    public int getHeight() {
        ...
```

```
    }
}
```

```
ArrayList<Student> students = ...

students.sort(new Comparator<Student>() {
    @Override
    public int compare(Student s1, Student s2) {
        return s1.getHeight() - s2.getHeight();
    }
});
```

Which three lambda expressions demonstrate equivalent sorting logic?

Options (select 3):

1.
```
students.sort((s1, s2) -> s1.getHeight() - s2.getHeight());
```

2.
```
students.sort((s1, s2) -> { return s1.getHeight() - s2.getHeight(); });
```

3.
```
students.sort((s1, s2) -> { s1.getHeight() - s2.getHeight(); });
```

4.
```
students.sort((Student s1, Student s2) -> { return s1.getHeight() - s2.getHeight(); });
```

5.
```
students.sort((Student s1, Student s2) ->  return s1.getHeight() - s2.getHeight(); );
```

Question 080103

Given the class:

```
public class StreamTest {
    @FunctionalInterface
    interface NameFilter extends Predicate<String> {
        public default boolean test(String str) {
            return str.contains("Ja");
        }
        public boolean doTest();
    }
    public static void main(String[] args) {
        List<String> strs = List.of("Jada", "Jana", "Jaelynn", "Jaylee", "Jaycee");
        Predicate<String> p1 = s -> s.length() > 4;
        NameFilter p2 = new NameFilter() {
            public boolean test(String s) {
```

```
                return s.startsWith("Jay");
            }
            public boolean doTest() { return false; }
        };
        long count = strs.stream().filter(p1).filter(p2).count();
        System.out.println(count);
    }
}
```

What is the result?

Options (select 1):

1. Compilation fails due to invalid syntax of `NameFilter` interface.
2. Compilation fails due to invalid syntax of `p2` definition.
3. `0`
4. `2`
5. `3`

8.2. Use lambda expressions and method references

Question 080201

Given the `Student` class:

```
class Student {
    String name;
    int height; // in centimeters
    public Student(String n, int h) {
        name = n;
        height = h;
    }
    static int orderByHeight(Student s1, Student s2) {
        return s1.height - s2.height;
    }
    public String toString() { return name + " " + height;  }
}
```

and the following code fragment:

```
Student[] arr = new Student[4];
arr[0] = new Student("George", 167);
arr[1] = new Student("Harry", 159);
arr[2] = new Student("Fred", 167);
arr[3] = new Student("Ron", 161);
... // line n1
System.out.println(Arrays.asList(arr));
```

Which code inserted at `line n1` position will print `[Harry 159, Ron 161, George 167, Fred 167]`?

Options (choose 1):

1.
```
Arrays.sort(arr, Student::orderByHeight(s1, s2));
```

2.
```
Arrays.sort(arr, Student::orderByHeight());
```

3.

```
Arrays.sort(arr, Student::orderByHeight);
```

4.

```
Arrays.sort(arr, (s1, s2) -> orderByHeight(s1, s2));
```

Question 080202

Given the code:

```
public class PrimeNumbersPrinter {

    public static void main(String[] args) {
        List<Integer> primes = Arrays.asList(new Integer[]{2, 3, 5, 7, 11, 13, 17, 19});
        PrimeNumbersPrinter p = new PrimeNumbersPrinter();
        ... // line n1
    }

    ... // line n2
}
```

Which two code changes when done at the same time will allow to print prime numbers to console?

Options (choose 2):

1. Replace `line n1` with the line:

```
primes.forEach(p::print);
```

2. Replace `line n1` with the line:

```
primes.forEach(p::print(i));
```

3. Replace `line n1` with the line:

```
primes.forEach(PrimeNumbersPrinter::print);
```

4. Replace `line n2` with the method:

```
public void print(Supplier<Integer> s) {
    System.out.println(s.get());
}
```

5. Replace `line n2` with the method:

```
public void print(Integer i) {
    System.out.println(i);
}
```

6. Replace `line n2` with the method:

```
public void print(List<Integer> l) {
    for (Integer i : l) {
        System.out.println(i);
    }
}
```

Question 080203

Given the `Student` class:

```
class Student {
    public String name;

    public Student(String s) {
        name = s;
    }
}
```

and the following code:

```
public class PopulateStudentsProcessor {

    public static List<String> names = Arrays.asList(new String[]{"Tabitha", "Sadie",
"Tristram"});
    public static List<Student> students = new ArrayList<>();

    public static void main(String[] args) {
        for (String n : names) {
            // line n1
        }
    }

    public static void registerStudent(Function<String, Student> f, String n) {
        Student s = f.apply(n);
        students.add(s);
    }
}
```

Which line inserted at `line n1` position will allow to polulate list of students?

Options (choose 1):

1.

```
registerStudent((n) -> new Student(n), n);
```

2.

```
registerStudent(new Function(new Student(n)), n);
```

3.

```
registerStudent(Student::new(n), n);
```

4.

```
registerStudent(Student::new, n);
```

Question 080204

Given the `MegaLogger` class:

```
import java.util.Date;
import java.util.function.Supplier;

public class MegaLogger {
    public static void main(String[] args) {
        // line n1
    }

    public static void logEvent(Supplier<Date> tms, String event) {
        System.out.println(tms.get() + " : " + event);
    }
}
```

Which line inserted at `line n1` position will log the event at current timestamp?

Options (choose 1):

1.

```
logEvent(Date()::new, "Application started");
```

2.

```
logEvent(Date::new, "Application started");
```

3.

```
logEvent(Date->new(), "Application started");
```

4.

```
logEvent(new Date(), "Application started");
```

8.3. Use built-in functional interfaces including `Predicate`, `Consumer`, `Function`, and `Supplier`

Question 080301

Which of the following Java functional interfaces from `java.util.function` package has `void` return type?

Options (choose 1):

1. `Predicate`
2. `Consumer`
3. `Function`
4. `Supplier`
5. `UnaryOperator`

Question 080302

Which of the following Java functional interfaces from `java.util.function` package always produces a result of the same type as its operand?

Options (choose 1):

1. `Predicate`
2. `Consumer`
3. `Function`
4. `Supplier`
5. `UnaryOperator`

Question 080303

Which of the following Java functional interfaces from `java.util.function` package may produce a `Boolean` object?

Options (choose 3):

1. `Predicate`
2. `Consumer`
3. `Function`
4. `Supplier`
5. `UnaryOperator`

Question 080304

What statement about `java.util.function.BiPredicate` is true?

Options (choose 1):

1. Its functional method returns `boolean` primitive value.
2. Its functional method accepts two boolean primitive arguments.
3. It improves performance by avoiding unnecessary auto-boxing operation.
4. It inherits from `Predicate` functional interface.

Question 080305

Which two Java functional interfaces from `java.util.function` package produce primitives?

Options (choose 2):

1. `Predicate`
2. `Consumer`
3. `Function`
4. `DoubleFunction`
5. `ToDoubleFunction`

Question 080306

Given the code:

```
class Student {
    public String name;
    public Student(String n) {
        name = n;
    }
    public String toString() {
        return name;
    }
}

public class Faculty {
    public static void main(String[] args) {
        Student p = new Student("Harry");
        Student w = new Student("Ron");
        Student g = new Student("Hermiona");
```

```
        List<Student> list = Arrays.asList(new Student[]{p, w, g});

        Function<List<Student>, Student> f1 = arg -> {
            for (Student s : arg) {
                if (s.name.startsWith("H")) {
                    s.name.toUpperCase();
                    return s;
                }
            }
            return null;
        };
        f1.apply(list); // line 1
        System.out.print(list);
    }
}
```

Assuming that all imports are correct, what is the result?

Options (choose 1):

1. `[Harry, Hermiona]`
2. `[HARRY, HERMIONA]`
3. `[Harry, Ron, Hermiona]`
4. `[HARRY, Ron, Hermiona]`
5. Compilation fails at `line 1`.

Question 080307

Given the `Advertizer` class:

```
import java.util.function.Function;
public class Advertizer {
    public static void main(String[] args) {
        String j8 = " Java 8";
        String j9 = " Java 9";
        Function<String, String> greatifier = s -> s + " is great!";
        System.out.print(addMarketing(j8, greatifier));
        addMarketing(j9, greatifier);
        System.out.print(j9);
    }
    public static String addMarketing(String s, Function<String, String> f) {
        return f.apply(s);
    }
}
```

What is the output?

Options (choose 1):

1. `Java 8 Java 9`
2. `Java 8 is great! Java 9`
3. `Java 8 Java 9 is great!`
4. `Java 8 is great! Java 9 is great!`

Question 080308

Given the `City` class:

```
public class City {
    public String name;
    public int population;
    public City(String n, int p) {
        name = n;
```

```
        population = p;
    }
}
```

and the following code:

```
City city1 = new City("Minsk", 2002600);
// line 1
```

which code fragment inserted at `line 1` will print city information?

Options (choose 1):

1.

```
Consumer<City> cons = city2 -> city2.name + " - " + city2.population;
cons.accept(city1);
System.out.print(city1);
```

2.

```
Consumer<City> cons = city2 -> System.out.print(city2.name + " - " + city2.population);
cons.accept(city1);
```

3.

```
Consumer<City> cons = city2 -> city2.name + " - " + city2.population;
System.out.print(cons.accept(city1));
```

4.

```
Consumer<City> cons = city2 -> city2.name + " - " + city2.population;
System.out.print(cons);
```

Question 080309

Given the `Programmer` class:

```
public class Programmer {
    public String name;
    public double salary;
    public Programmer(String n, double s) {
        name = n;
        salary = s;
    }
}
```

and the following code:

```
public static void main(String[] args) {
    Programmer p1 = new Programmer("Volha Zaikina", 2100);
    Programmer p2 = new Programmer("Mikalai Zaikin", 2000);
    List<Programmer> list = new ArrayList();
    list.add(p1);
    list.add(p2);
    // line 1
}
```

which two code fragments independently inserted at `line 1` will increase salary by 10%?

Options (choose 2):

1.

```
Consumer cons = (Programmer p) -> p.salary = p.salary * 1.1;
list.forEach(cons);
```

2.

```
list.forEach(p -> { p.salary = p.salary * 1.1; } );
```

3.

```
Consumer<Programmer> cons = p -> p.salary = p.salary * 1.1;
list.forEach(cons);
```

4.

```
list.forEach(Programmer p -> { p.salary = p.salary * 1.1; return p; } );
```

Question 080310

Given the `Book` class:

```
public class Book {
    String title;
    int isbn;

    public Book(String t, int i) {
        title = t;
        isbn = i;
    }

    public String toString() {
        return title + ", ISBN " + isbn;
    }
}
```

and the following code:

```
Supplier<Book> s = ...; // code here
Book b = s.get();
System.out.println(b);
```

which code fragment inserted instead of `...; // code here` placeholder will print `Exam 1Z0-817: Upgrade to Java SE 11 Developer Study Guide, ISBN 1234567890`?

Options (choose 1):

1. `Book("Exam 1Z0-817: Upgrade to Java SE 11 Developer Study Guide", 1234567890)::new`
2. `Book::new("Exam 1Z0-817: Upgrade to Java SE 11 Developer Study Guide", 1234567890)`
3. `(Book) -> new Book("Exam 1Z0-817: Upgrade to Java SE 11 Developer Study Guide", 1234567890)`
4. `() -> new Book("Exam 1Z0-817: Upgrade to Java SE 11 Developer Study Guide", 1234567890)`

Question 080311

Given the code:

```
class Phone {
    public Phone() {
        System.out.print("I can dial.");
    }
}

class Smartphone extends Phone {
    public Smartphone() {
        System.out.print("I can run Java.");
    }
}

public class PhoneTest {
    public static void main(String[] args) {
        Supplier<Phone> sp = () -> new Phone(); // line 1
        Supplier<Smartphone> ss = Smartphone::new; // line 2
        Phone phone = ss.get();
    }
}
```

What is the result?

Options (choose 1):

1. `I can dial.`
2. `I can dial.I can run Java.`
3. Compilation fails at `line 1`.
4. Compilation fails at `line 2`.

Question 080312

Given the code:

```
UnaryOperator<Integer> uo1 = i -> i + 2;
UnaryOperator<Integer> uo2 = i -> i * 2;
Integer i = 5;
System.out.println(uo1.apply(uo2.apply(i)));
```

What is the result?

Options (choose 1):

1. `5`
2. `7`
3. `10`
4. `12`
5. `14`

Question 080313

Given the code:

```
UnaryOperator<Integer> uo = i -> "Java " + i; // line 1
uo.apply(7); // line 2
System.out.print(uo.apply(8));
```

What is the result?

Options (choose 1):

1. Compilation fails at `line 1`
2. Compilation fails at `line 2`
3. Runtime exception thrown at `line 2`
4. Code prints `Java 8`

Question 080314

Given the code fragment:

```
// line 1
List<Double> list = new ArrayList<>();
list.add(12.34);
list.add(56.78);
list.replaceAll(f);
System.out.print(list);
```

Which code inserted at `line 1` position will print `[123.4, 567.8]`?

Options (choose 1):

1.
```
UnaryOperator<Double> f = d -> d * 10;
```

2.
```
Function<Double> f = d -> d * 10;
```

3.
```
Supplier<Double> f = d -> d * 10;
```

4.
```
```

```
Consumer<Double> f = d -> d * 10;
```

5.

```
ToDoubleFunction f = d -> d * 10;
```

Question 080315

Given the code:

```
class Photo {

    private String name;
    private int size;

    public Photo(String n, int s) {
        name = n;
        size = s;
    }

    // getters and setters go here

    public String toString() {
        return name + " " + size;
    }
}

public class FileManager {

    public static void main(String[] args) {
        List<Photo> photos = new ArrayList<>();
        photos.add(new Photo("Sea view", 123));
        photos.add(new Photo("Mountain view", 456));

        // line 1

        System.out.print(photos.stream().filter(c).collect(Collectors.toList()));
    }
}
```

Which line, inserted at `line 1` position, will enable the code to print `[Mountain view 456]`?

Options (choose 1):

1.

```
Photo<Predicate> c = (Photo p) -> p.getSize() > 300;
```

2.

```
Predicate<Photo> c = p -> p.getSize() > 300;
```

3.

```
Predicate c = p -> p.getSize() > 300;
```

4.

```
Photo c = p -> p.getSize() > 300;
```

Question 080316

Given the code:

```
public class Teacher {
    public Teacher(String lName){
        lastName = lName;
    }
    public String lastName;
    public String toString() { return lastName; }
}
```

```
interface MagicPredicate extends Predicate<Teacher> {
    static boolean validate(Teacher t) {
        return "Snape".equals(t.lastName);
    }
    default boolean check(Teacher t) { // line 1
        return "Hagrid".equals(t.lastName);
    }
}
```

```
public class University {
    public static void main(String[] args){
        Teacher t1 = new Teacher("Snape");
        Teacher t2 = new Teacher("Lupin");
        Teacher t3 = new Teacher("Hagrid");

        List<Teacher> teachers = List.of(t1,t2,t3);

        Predicate<Teacher> f = new MagicPredicate() {
            public boolean test(Teacher t) { // line 2
                return "Lupin".equals(t.lastName);
            }
        };

        System.out.print(teachers.stream().filter(f).findAny().get());
    }
}
```

Assuming all imports are correct, what is the output?

Options (choose 1):

1. `Snape`
2. `Lupin`
3. `Hagrid`
4. Compilation fails at `line 1`
5. Compilation fails at `line 2`

8.4. Use primitive and binary variations of base interfaces of `java.util.function` package

Question 080401

Given the code:

```
```

```
Map<Integer, String> months = new HashMap<>();
months.put(1, "January");
months.put(2, "February");
months.put(3, "March");
// ... more months here
months.put(12, "December");

// line 1

months.forEach(f);
```

Which line inserted at `line 1` position will print numbers of all months which name starts with "`J`"?

Options (choose 1):

1. `Consumer<Map.Entry> f = (m) -> { if (m.getValue().startsWith("J")) System.out.println(m.getKey()); };`
2. `BiConsumer<Integer, String> f = (i, s) -> { if (s.startsWith("J")) System.out.println(i); };`
3. `BiConsumer<int, String> f = (i, s) -> { if (s.startsWith("J")) System.out.println(i); };`
4. `Consumer<String> f = (i, s) -> { if (s.startsWith("J")) System.out.println(i); };`

Question 080402

You are writing new reporting component for the Java application for mobile phone operator. The component should accept as input a mobile phone number and generate monthly bill for the customer. Mobile phone numbers are currently stored as `long` primitives in the existing part of the application. It is expected a high volume of bulk operations every month for your component and you should avoid high memory consumption by your code.

Which Java Functional Interface from `java.util.function` package will help you to implement the new requirement?

Options (choose 1):

1. `LongBinaryOperator`
2. `LongUnaryOperator`
3. `ToLongFunction`
4. `LongFunction`
5. `Function`

Question 080403

You are hired to add new functionality to existing Java application for mobile phone operator. The new function should accept as input BASE64-encoded string with client's information (phone number and authorization code) and return number of free minutes left on the balance as `int` primitive.

Which Java Functional Interface from `java.util.function` package will help you to implement the new requirement?

Options (choose 1):

1. `IntFunction`
2. `ToIntBiFunction`
3. `ToIntFunction`
4. `IntSupplier`

Chapter 9. Parallel Streams

9.1. Develop the code that use parallel streams

Question 090101

Given the code fragment:

```
Integer[] arr = {1, 2, 3, 4, 5};
List<Integer> lst = Arrays.asList(arr);
lst.stream().parallel().peek(System.out::print).count(); // line n1
```

What is the result?

Options (choose 1):

1. `12345`
2. Numbers in unpredictable order.
3. Runtime exception at `line n1`.
4. Compilation fails `line n1`.

Question 090102

Given the code fragment:

```
Stream<Integer> s = Stream.of(new Integer[]{1, 2, 3, 4, 5});
List lst = s.parallel().collect(Collectors.toList()); // line n1
System.out.print(lst);
```

What is the result?

Options (choose 1):

1. `[1, 2, 3, 4, 5]`
2. Numbers in unpredictable order.
3. Runtime exception at `line n1`.
4. Compilation fails at `line n1`.

Question 090103

Which statement is correct about Java Streams?

Options (choose 1):

1. Type of the stream (sequential or parallel) can not be changed once the stream is created from the source.
2. Parallel streams are faster than sequential.
3. Parallel streams are thread safe.
4. Parallel stream pipelines are lazy.

9.2. Implement decomposition and reduction with streams

Question 090201

Given the code fragment:

```
List<Integer> list = List.of(3, 7, 5, 2);
... // line n1
System.out.print(sum);
```

Which line inserted at `line n1` position will print `17`?

Options (choose 1):

1. `int sum = list.stream().sum();`
2. `int sum = list.stream().reduce((a, b) -> a + b);`
3. `int sum = list.stream().reduce((a, b) -> a + b).sum();`
4. `int sum = list.stream().reduce(0, (a, b) -> a + b);`

Question 090202

Given the code fragment:

```
List<String> list = List.of("H", "e", "l", "l", "o");
... // line n1
System.out.print(s);
```

Which two lines inserted independently at `line n1` position will print `Hello`?

Options (choose 2):

1. `String s = list.stream().reduce("H", (a, b) -> a + b);`
2. `String s = list.stream().reduce("", a.concat(b));`
3. `String s = list.stream().reduce("", String::concat);`
4. `String s = list.stream().reduce((a, b) -> a.concat(b)).get();`

Question 090203

Given the code fragment:

```
Map.Entry<Integer, String> entry = Map.entry(0, "A");
HashMap<Integer, String> map = new HashMap<>();
map.put(1, "B");
map.put(2, "C");
Map.Entry<Integer, String> result = map.entrySet().stream().reduce(entry,
        (a, b) -> Map.entry(a.getKey()*b.getKey(), a.getValue()+b.getValue()),
        (a, b) -> Map.entry(a.getKey()*b.getKey(), a.getValue()+b.getValue()));
System.out.println(result.getKey() + result.getValue());
```

What is the output?

Options (choose 1):

1. `0ABC`
2. `3ABC`
3. `3A`
4. `123A`

Chapter 10. Language Enhancements

10.1. Use try-with-resources construct

Question 100101

Given the two resource classes:

```
class Reader implements AutoCloseable {
    public void close() throws Exception {
        System.out.print("Reader closed.");
    }
    public String read() {
        throw new NullPointerException("Read failed.");
    }
}
```

```
class Writer implements AutoCloseable {
    public void close() throws Exception {
        System.out.print("Writer closed.");
    }
    public void write() {
        throw new NullPointerException("Write failed.");
```

```
    }
}
```

and the code fragment:

```
try (Reader r = new Reader(); Writer w = new Writer()) {
    r.read();
    w.write();
} catch (Exception e) {
    System.out.print(e.getMessage());
}
```

What is the output?

Options (choose 1):

1. `Reader closed.Writer closed.`
2. `Writer closed.Reader closed.Read failed.`
3. `Read failed.`
4. `Reader closed.Read failed.`

Question 100102

Given the two resource classes:

```
class Reader implements AutoCloseable {
    public void close()  { throw new NullPointerException("Close failed."); }
    public String read() { throw new NullPointerException("Read failed."); }
}
```

```
class Writer implements AutoCloseable {
    public void close() { System.out.print("Writer closed."); }
    public void write() { throw new NullPointerException("Write failed."); }
}
```

and the code fragment:

```
try (Reader r = new Reader(); Writer w = new Writer()) {
    r.read();
    w.write();
} catch (Exception e) {
    System.out.print(e.getMessage());
}
```

What is the output?

Options (choose 1):

1. `Writer closed.Read failed.`
2. `Writer closed.Read failed.Close failed.`
3. `Writer closed.Close failed.Read failed.`
4. `Close failed.Read failed.`

Question 100103

Given the resources:

```
class Reader implements Closeable { ... }
```

```
class Writer implements Closeable { ... }
```

and the code fragment:

```
Reader r = new Reader(); Writer w = new Writer();
try (r;w) {
    r.read();
    w.write();
} catch (Exception e) {
    ...
} finally {
    ...
}
```

Which statement is true?

Options (choose 1):

1. `Reader` and `Writer` must implement `AutoCloseable` interface.
2. The `Reader r = new Reader(); Writer w = new Writer();` line must be placed inside `try(...)` block.
3. A programmer can close reources inside the `catch` block if needed.
4. The try-with-resourecs may not have `finally` block.

10.2. Develop code that handles multiple Exception types in a single catch block

Question 100201

Given the exception classes:

```
class A extends Exception {}
```

```
class B extends A {}
```

```
class C extends RuntimeException {}
```

```
class D extends C {}
```

And assuming the `try` block is fully valid. Which `catch` block will compile successfully?

Options (choose 1):

1. `catch (A | D | NullPointerException ex) { }`
2. `catch (A | B ex) { }`
3. `catch (A | C | Exception ex) { }`
4. `catch (D | Throwable ex) { }`

Question 100202

Given the exception classes:

```
class BusinessException extends Exception {}
```

```
class A extends BusinessException {}
```

```
class B extends BusinessException {}
```

and the code fragment:

```
try {
    ...
} catch (A | B ex) { // line n1
    ex = new BusinessException(); // line n2
    throw ex;
}
```

Assuming the `try` block and surrounding method code is valid. What is the result?

Options (choose 1):

1. Compilation fails due to line n1
2. Compilation fails due to line n2
3. Compilation succeeds, and if the `try` block fails the `catch` block throws `A` or `B` exception
4. Compilation succeeds, and if the `try` block fails the `catch` block throws `BusinessException`

Question 100203

Given the exception classes:

```
class BusinessException extends Exception {}
```

```
class A extends BusinessException {}
```

```
class B extends BusinessException {}
```

and the method:

```
public void doIt() throws A, B { // line n1
    try {
        boolean someCondition = ... // calculate the flag
        if (someCondition) throw new A(); else throw new B();
    } catch (Exception ex) { // line n2
        logger.log(ex);
        throw ex;
    }
}
```

Assuming the surrounding code is valid. What is the result?

Options (choose 1):

1. Compilation fails at line n1. To make it compilable change the line n1 as follows:

   ```
   public void doIt() throws Exception { // line n1
   ```

2. Compilation fails at line n1. To make it compilable change the line n1 as follows:

   ```
   public void doIt() throws BusinessException { // line n1
   ```

3. Compilation fails at line n2. To make it compilable change the line n2 as follows:

```
} catch (A | B ex) { // line n2
```

4. Compilation succeeds without changes.