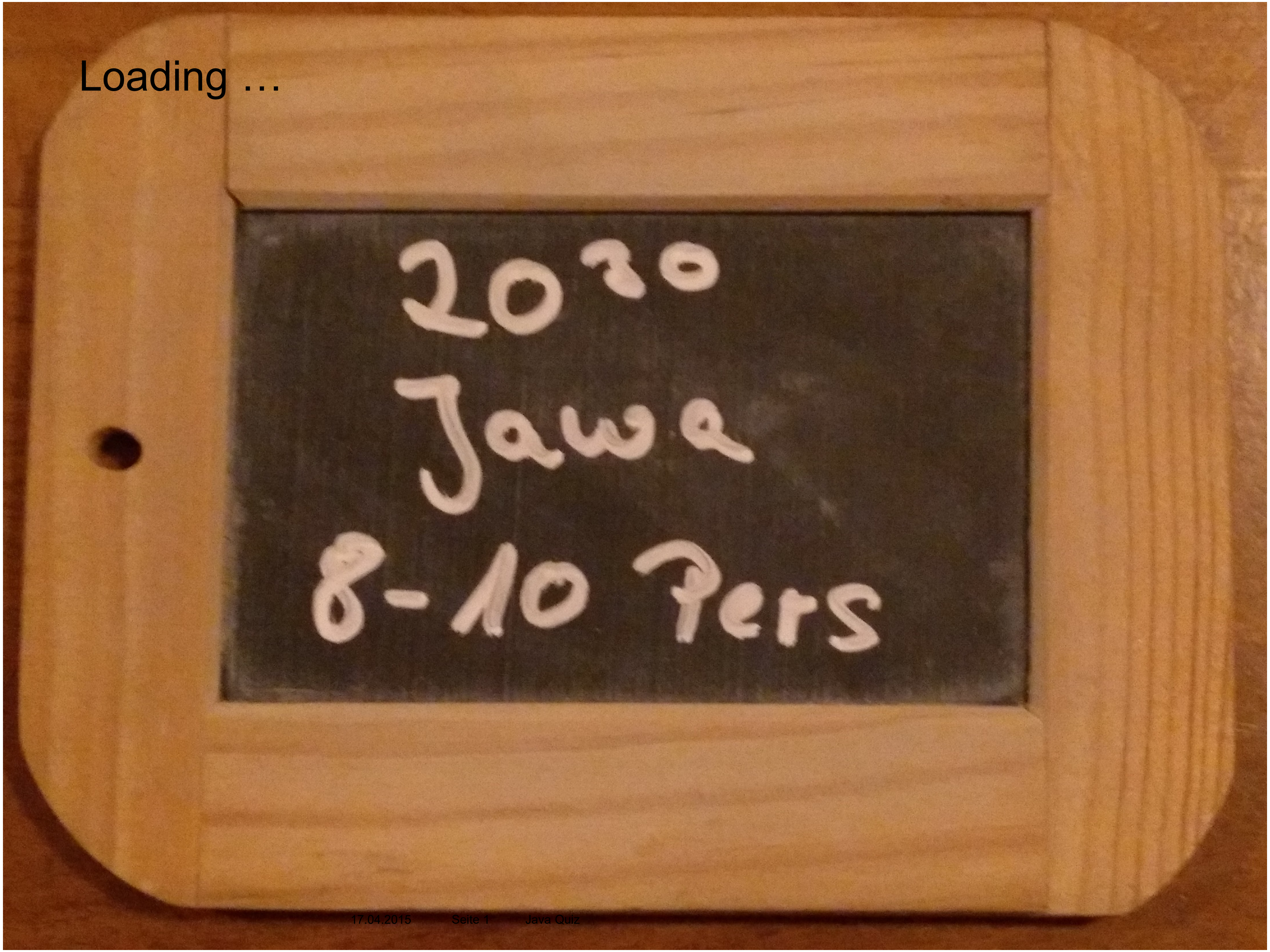
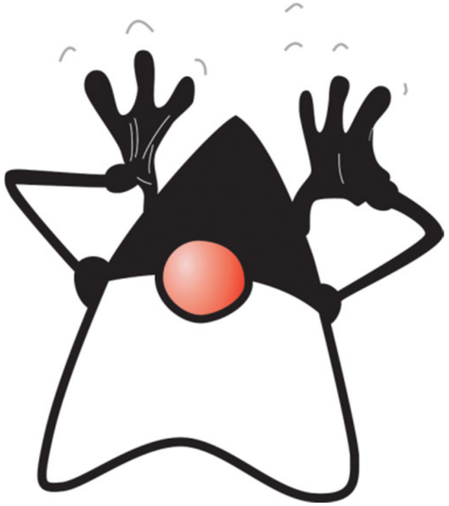


Loading ...

finanz informatik
solutions plus



Java (11) Quiz
What you always knew about Java but no-one came to ask

Frankfurt am Main, 27. Januar 2021

Jens-Hagen Syrbe
Finanz Informatik Solutions Plus GmbH

Modus Operandi

-
  ```
  IntStream.range(0, 0x17).forEach(i -> {
      askQuestion(i);
      discussQuestion(i);
      storeAnswers(i);
      discussAnswers(i);
      showSolution(i);
  });
  getFancyPrizes();
  ```

- Interactive
  - Etherpad https://pad.riseup.net/p/JugfJavaQuiz

- Questions state clearly if one or more answers are correct.

0 Test

- Do you understand how to store your answers in Etherpad? https://pad.riseup.net/p/JugfJavaQuiz

a) Yess!

b) Nope!

0 Test

- Do you understand how to store your answers in Etherpad? https://pad.riseup.net/p/JugfJavaQuiz

✓ a) Yess!

✗ b) Nope!

1 Primitive data types

- Which one of the following code snippets does compile?

```
a) float number = 3.14;

b) int number = 10L + 20L;

c) int number = 10L + 2;

d) long number = 10L + 2;
```

1 Primitive data types

- Which one of the following code snippets does compile?

✗ `a) float number = 3.14;`

✗ `b) int number = 10L + 20L;`

✗ `c) int number = 10L + 2;`

✓ `d) long number = 10L + 2;`

- Bonus question: Why does this compile? (JLS* 15.26.2)

```
int i = 2;
i += Long.MAX_VALUE;
```

- * Java Language Specification
  https://docs.oracle.com/javase/specs/jls/se11/html/index.html

2 IEEE 754 floating point arithmetic

- What will be the output of:

```
System.out.println(Math.floor( (1 * 0 + 0.1 + 0.7) * 10));
```

a) 0.0

b) 7.0

c) 8.0

d) 18.0

2 IEEE 754 floating point arithmetic

- What will be the output of:

```
System.out.println(Math.floor( (1 * 0 + 0.1 + 0.7) * 10));
```

✗ a) 0.0

✓ b) 7.0

✗ c) 8.0

✗ d) 18.0

3 Equality of objects

- Assuming the JVM runs with default settings and given the variables

```
Integer a = 42;
Integer b = 42;
Integer c = 420;
Integer d = 420;
```

what is the value of

```
(a == b) ^ (c == d)
```

a) true

b) false

c) depends on the JVM implementation

3 Equality of objects

- Assuming the JVM runs with default settings and given the variables

```
Integer a = 42;
Integer b = 42;
Integer c = 420;
Integer d = 420;
```

what is the value of

```
(a == b) ^ (c == d)
```

✓ a) true

✗ b) false

✗ c) depends on the JVM implementation

4 Loops

- Which of the following for loops is an infinite loop? (choose up to 3)

```
a) for(; ;)

b) for(int i=0; ; i++)

c) for(int i=0; i<1; i--)
```

d) none of the above

4 Loops

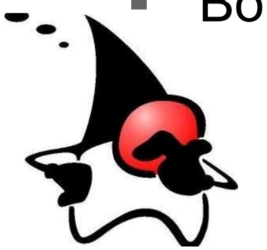
- Which of the following for loops is an infinite loop? (choose up to 3)

✓ a) `for(; ;)`

✓ b) `for(int i=0; ; i++)`

✗ c) `for(int i=0; i<1; i--)`

✗ d) none of the above



- Bonus: Exactly one of the following is an infinite loop (JLS 5.6.1):

```
for (short i = -1; i != 0; i >>>=1)
for (int   i = -1; i != 0; i >>>=1)
```

5 Language concepts: finally

- What will be the output of the main method?

```java
public class Question {
  public static String compute(final int val) {
    try {
      if (val == 0) throw new Exception();
      return "0";
    } catch (final Exception exception) {
      return "1";
    } finally {
      return "2";
  } }
  public static void main(final String[] args) {
    System.out.println(compute(0) + compute(1) + compute(2));
} }
```

a) 012

b) 100

c) 202

d) 222

5 Language concepts: finally

- What will be the output of the main method?

```java
public class Question {
  public static String compute(final int val) {
    try {
      if (val == 0) throw new Exception();
      return "0";
    } catch (final Exception exception) {
      return "1";
    } finally {
      return "2";
  } }
  public static void main(final String[] args) {
    System.out.println(compute(0) + compute(1) + compute(2));
} }
```

✗ a) 012

✗ b) 100

✗ c) 202

✓ d) 222

Rule of thumb: never return from `finally`

6 Classes: inheritance

- What is the behaviour of this code? (choose 1)

```
class Parent {
    public void crashes() {
        throw new RuntimeException("Eh!");
}   }
class SonA extends Parent {
    public void crashes() throws IOException {
        throw new IOException("Eh!");
}   }
class Test extends Parent {
    public static void main(String[] args) {
        ((Parent) new SonA()).crashes();
}   }
```

a) Throws a RuntimeException.

b) Throws an IOException.

c) Does not compile.

6 Classes: inheritance

- What is the behaviour of this code? (choose 1)

```java
class Parent {
    public void crashes() {
        throw new RuntimeException("Eh!");
}   }
class SonA extends Parent {
    public void crashes() throws IOException {
        throw new IOException("Eh!");
}   }
class Test extends Parent {
    public static void main(String[] args) {
        ((Parent) new SonA()).crashes();
}   }
```

✗ a) Throws a RuntimeException.

✗ b) Throws an IOException.

✓ c) Does not compile.

7 Classes: prevent inheritance

- How can you prevent that a class is being inherited?
(choose all that apply)

a) Make your class final.

b) Even if your class is not final, just make all the constructors of your class private.

c) Even if your class is not final, just make all the constructors of your class final.

d) Even if your class is not final, just make all the constructors of your class default.

7 Classes: prevent inheritance

- How can you prevent that a class is being inherited? (choose all that apply)

✓ a) Make your class final.

✓ b) Even if your class is not final, just make all the constructors of your class private.

✗ c) Even if your class is not final, just make all the constructors of your class final.

✗ d) Even if your class is not final, just make all the constructors of your class default.

8 Classes: visibility

- Given these two classes, what can be done to make the code compile? (choose 1)

```java
package p1; // line n1
import java.io.PrintStream;
public class Logger {
  PrintStream ps;
  public void log(String s) {
    ps.println(s);
  }
  Logger() {
    ps = System.out;
} }
```

```java
package p2;
import java.util.Arrays;
// line n2
public class LoggingEngine {
  public static void main(String[] args) {
    Logger logger = new Logger();
    Arrays.asList(args).stream().forEach(
        s -> logger.log(s));
} }
```

a) At line n2, add `import p1.Logger;` .

b) At line n2, add `import static p1.Logger;` .

c) Change line n1 to `package p2;` and move the source file appropriately.

d) Make the class LoggingEngine extend Logger and add the protected modifier to the Logger constructor.

8 Classes: visibility

- Given these two classes, what can be done to make the code compile? (choose 1)

```
package p1; // line n1
import java.io.PrintStream;
public class Logger {
  PrintStream ps;
  public void log(String s) {
    ps.println(s);
  }
  Logger() {
    ps = System.out;
} }
```

```
package p2;
import java.util.Arrays;
// line n2
public class LoggingEngine {
  public static void main(String[] args) {
    Logger logger = new Logger();
    Arrays.asList(args).stream().forEach(
        s -> logger.log(s));
} }
```

✗ a) At line n2, add `import p1.Logger;` .

✗ b) At line n2, add `import static p1.Logger;` .

✓ c) Change line n1 to `package p2;` and move the source file appropriately.

✗ d) Make the class LoggingEngine extend Logger and add the protected modifier to the Logger constructor.

9 Classes: encapsulation

- Which statement correctly characterizes class encapsulation?

```java
public class BookOrder {
  private int id;
  private List<String> items = new ArrayList<>();
  public BookOrder(int id)      {  this.id = id; }
  public int getId()             {  return id; }
  public List<String> getItems() {  return items; }
  public String toString()       {  return id + " " + items; }
}
```

a) The class is properly encapsulated because its instance variables are private.

b) The class is not properly encapsulated because client code can arbitrarily change object state.

c) The class is not properly encapsulated because the implementation of the id variable allows the creation of orders with the same id.

d) The class is not properly encapsulated because encapsulation implies getters and setters, and there are no setter methods.

9 Classes: encapsulation

- Which statement correctly characterizes class encapsulation?

```java
public class BookOrder {
  private int id;
  private List<String> items = new ArrayList<>();
  public BookOrder(int id)       {  this.id = id; }
  public int getId()             {  return id; }
  public List<String> getItems() {  return items; }
  public String toString()       {  return id + " " + items; }
}
```

✗ a) The class is properly encapsulated because its instance variables are private.

✓ b) The class is not properly encapsulated because client code can arbitrarily change object state.

✗ c) The class is not properly encapsulated because the implementation of the id variable allows the creation of orders with the same id.

✗ d) The class is not properly encapsulated because encapsulation implies getters and setters, and there are no setter methods.

10 Classes: generics

- Given the following classes, which of the code snippets do compile? (choose 2)

```
public class AnimalHouse<E> {
  private E animal;
  public E getAnimal() { return animal; }
  public void setAnimal(E x) { animal = x; }
}
class Animal {}
class Cat extends Animal {}
class Dog extends Animal {}
```

```
a) AnimalHouse<Animal> house = new AnimalHouse<Cat>();
   house.setAnimal(new Cat());
b) AnimalHouse<?> house = new AnimalHouse<Cat>();
   house.setAnimal(new Cat());
c) AnimalHouse house = new AnimalHouse();
   house.setAnimal(new Dog());
d) AnimalHouse<Animal> house = new AnimalHouse<Animal>();
   house.setAnimal(new Cat());
e) AnimalHouse<? extends Animal> house = new AnimalHouse<Cat>();
   house.setAnimal(new Cat());
```

10 Classes: generics

- Given the following classes, which of the code snippets do compile? (choose 2)

```
public class AnimalHouse<E> {
  private E animal;
  public E getAnimal() { return animal; }
  public void setAnimal(E x) { animal = x; }
}
class Animal {}
class Cat extends Animal {}
class Dog extends Animal {}
```

✗ a)

```
AnimalHouse<Animal> house = new AnimalHouse<Cat>();
house.setAnimal(new Cat());
```

✗ b)

```
AnimalHouse<?> house = new AnimalHouse<Cat>();
house.setAnimal(new Cat());
```

✓ c)

```
AnimalHouse house = new AnimalHouse();
house.setAnimal(new Dog());
```

Warning: Raw use of …

Warning: Unchecked call …

✓ d)

```
AnimalHouse<Animal> house = new AnimalHouse<Animal>();
house.setAnimal(new Cat());
```

but getAnimal returns Animal, not Cat

✗ e)

```
AnimalHouse<? extends Animal> house = new AnimalHouse<Cat>();
house.setAnimal(new Cat());
```

11 Classes: type inference

- How can an additional method of an anonymous class be invoked? (choose up to 4)

```
a) Object extendedObject = new Object() {
     public void additionalMethod() { ... } };
   extendedObject.additionalMethod();
```

```
b) new Object() {
     public void additionalMethod() { ... }
   }.additionalMethod();
```

```
c) var extendedObject = new Object() {
     public void additionalMethod() { ... } };
   extendedObject.additionalMethod();
```

```
d) Callable<? extends Object> extendedObject =
     () -> new Object() {
       public void additionalMethod() { ... } };
   extendedObject.additionalMethod();
```

e) None of the above will work.

11 Classes: type inference

- How can an additional method of an anonymous class be invoked? (choose up to 4)

```
✗ a) Object extendedObject = new Object() {
       public void additionalMethod() { ... } };
     extendedObject.additionalMethod();
✓ b) new Object() {
       public void additionalMethod() { ... }
     }.additionalMethod();
✓ c) var extendedObject = new Object() {
       public void additionalMethod() { ... } };
     extendedObject.additionalMethod();
✗ d) Callable<? extends Object> extendedObject =
       () -> new Object() {
         public void additionalMethod() { ... } };
     extendedObject.additionalMethod();
```

✗ e) None of the above will work.

12 Reflection

- Trying to break the concept of a singleton. What will be printed?

```java
public class MySingleton {
  private static MySingleton ourInstance = new MySingleton();
  private MySingleton() {};
  public static MySingleton getInstance() { return ourInstance; }
}
public class Main {
  public static void main(String[] args) {
    MySingleton singleton = MySingleton.getInstance();
    try {
      Class clazz = Class.forName("MySingleton");
      Constructor[] constructors = clazz.getDeclaredConstructors();
      constructors[0].setAccessible(true);
      MySingleton other = (MySingleton) constructors[0].newInstance();
      if (other == singleton) System.out.println("They are the same");
      else System.out.println("They are not the same");
    } catch (ClassNotFoundException e) { System.out.println("This should not happen");
    } catch (InstantiationException e) { System.out.println("Instantiation exception");
    } catch (IllegalAccessException e) { System.out.println("Illegal access");
    } catch (InvocationTargetException e) { System.out.println("Invocation target exception");
} } }
```

a) They are the same

b) They are not the same

c) Instantiation exception

d) Illegal access

12 Reflection

- Trying to break the concept of a singleton. What will be printed?

```java
public class MySingleton {
  private static MySingleton ourInstance = new MySingleton();
  private MySingleton() {};
  public static MySingleton getInstance() { return ourInstance; }
}
public class Main {
  public static void main(String[] args) {
    MySingleton singleton = MySingleton.getInstance();
    try {
      Class clazz = Class.forName("MySingleton");
      Constructor[] constructors = clazz.getDeclaredConstructors();
      constructors[0].setAccessible(true);
      MySingleton other = (MySingleton) constructors[0].newInstance();
      if (other == singleton) System.out.println("They are the same");
      else System.out.println("They are not the same");
    } catch (ClassNotFoundException e) { System.out.println("This should not happen");
    } catch (InstantiationException e) { System.out.println("Instantiation exception");
    } catch (IllegalAccessException e) { System.out.println("Illegal access");
    } catch (InvocationTargetException e) { System.out.println("Invocation target exception");
} } }
```

✗ a) They are the same

✓ b) They are not the same

✗ c) Instantiation exception

✗ d) Illegal access

13 Collection and Stream API: List

- What will be printed by the program?

```java
System.out.print("Bonjour ");
List<String> list = Arrays.asList("Hello", "Hey", "Bye", "Ciao");
try {
  for (int i = 0; i < 6; i++) {
    if (i == 3) {
      list.add("Hi");
      System.out.print("Challenge ");
  } }
} catch (ConcurrentModificationException e) { System.out.print("Java ");
} catch (UnsupportedOperationException uoe) { System.out.print("Editx ");
} catch (ArrayIndexOutOfBoundsException ae) { System.out.print("Quiz ");
}
```

a) Bonjour Java

b) Bonjour Editx

c) Bonjour Challenge

d) Bonjour Challenge Java

e) Bonjour Challenge Editx

f) Bonjour Challenge Quiz

13 Collection and Stream API: **Unmodifiable**List

- What will be printed by the program?

```java
System.out.print("Bonjour ");
List<String> list = Arrays.asList("Hello", "Hey", "Bye", "Ciao");
try {
  for (int i = 0; i < 6; i++) {
    if (i == 3) {
      list.add("Hi");
      System.out.print("Challenge ");
  } }
} catch (ConcurrentModificationException e) { System.out.print("Java ");
} catch (UnsupportedOperationException uoe) { System.out.print("Editx ");
} catch (ArrayIndexOutOfBoundsException ae) { System.out.print("Quiz ");
}
```

✗ a) Bonjour Java

✓ b) Bonjour Editx

✗ c) Bonjour Challenge

✗ d) Bonjour Challenge Java

✗ e) Bonjour Challenge Editx

✗ f) Bonjour Challenge Quiz

Same result for

```java
List<String> list = List.of("Hello", "Hey", "Bye", "Ciao");
```

14 Collection and Stream API: Map

- Given an instance of Map, you would like to create a new instance of Map from the existing one that has the same iteration order. Which concrete implementation of the Map interface should be used for the new instance? (choose 1)

a) HashMap

b) LinkedHashMap

c) TreeMap

d) The answer depends on the implementation of the existing instance.

14 Collection and Stream API: Map

- Given an instance of Map, you would like to create a new instance of Map from the existing one that has the same iteration order. Which concrete implementation of the Map interface should be used for the new instance? (choose 1)

✗ a) HashMap

✓ b) LinkedHashMap

✗ c) TreeMap

✗ d) The answer depends on the implementation of the existing instance.

15 Collection and Stream API: ConcurrentMap

- Among the following ConcurrentMap methods, which one is atomic?

a) compute

b) computeIfAbsent

c) putIfAbsent

d) replaceAll

15 Collection and Stream API: ConcurrentMap

- Among the following ConcurrentMap methods, which one is atomic?

✗ a) compute

✗ b) computeIfAbsent

✓ c) putIfAbsent

✗ d) replaceAll

16 Collection and Stream API: Streams

- Given the code:

```
    List<String> src = List.of("Java 11", "Exam");
```

Which code fragment determines if the word “Java” is present in the `src` list in the most computationally efficient way? (choose 1)

```
a) List<String> res = List.of();
   src.stream()
      .peek(v -> { if (v.contains("Java")) res.add(v); })
      .count();
   var a = (res.size() > 0);


b) var a = src.stream()
      .filter(v -> v.contains("Java"))
      .findAny();


c) var a = src.stream().anyMatch(v -> v.contains("Java"));


d) var a = src.stream()
      .filter(v -> v.contains("Java"))
      .collect(Collectors.toList());
```

16 Collection and Stream API: Streams

- Given the code:

```
    List<String> src = List.of("Java 11", "Exam");
```

Which code fragment determines if the word “Java” is present in the `src` list in the most computationally efficient way? (choose 1)

✗

```
a) List<String> res = List.of();
   src.stream()
      .peek(v -> { if (v.contains("Java")) res.add(v); })
      .count();
   var a = (res.size() > 0);
```

✗

```
b) var a = src.stream()
      .filter(v -> v.contains("Java"))
      .findAny();
```

✓

```
c) var a = src.stream().anyMatch(v -> v.contains("Java"));
```

✗

```
d) var a = src.stream()
      .filter(v -> v.contains("Java"))
      .collect(Collectors.toList());
```

17 Collection and Stream API: Optional

- What is the output if you run the compiled class as `java OptionalArgs Java` ? (choose 1)

```
import java.util.Optional;

public class OptionalArgs {
  public static void main(String[] args) {
    var v = Optional.ofNullable(args[0])
        .or(() -> Optional.ofNullable(args[1]))
        .flatMap(o -> Optional.empty())
        .stream().findFirst().orElse("default");
    System.out.print(v);
} }
```

a) A runtime exception

b) Blank output

c) `Java`

d) `default`

17 Collection and Stream API: Optional

- What is the output if you run the compiled class as `java OptionalArgs Java` ? (choose 1)

```java
import java.util.Optional;

public class OptionalArgs {
  public static void main(String[] args) {
    var v = Optional.ofNullable(args[0])
        .or(() -> Optional.ofNullable(args[1]))
        .flatMap(o -> Optional.empty())
        .stream().findFirst().orElse("default");
    System.out.print(v);
} }
```

✗ a) A runtime exception

✗ b) Blank output

✗ c) `Java`

✓ d) `default`

18 Functional programming: lambda syntax

- Which of the following lines can be inserted at line n1 to print true? (choose all that apply)

```java
public static void main(String[] args) {
  // line n1
}
private static boolean challenge(Predicate<Integer> open) {
  return open.test(5);
}
```

```
a) System.out.println(challenge(i -> i==5));
b) System.out.println(challenge(i -> {i==5;}));
c) System.out.println(challenge((i) -> i==5));
d) System.out.println(challenge((i) -> {return i==5;}));
e) System.out.println(challenge((int i) -> i==5));
f) System.out.println(challenge((int i) -> {return i==5;}));
```

18 Functional programming: lambda syntax

- Which of the following lines can be inserted at line n1 to print true? (choose all that apply)

```
public static void main(String[] args) {
  // line n1
}
private static boolean challenge(Predicate<Integer> open) {
  return open.test(5);
}
```

✓ `a) System.out.println(challenge(i -> i==5));`

✗ `b) System.out.println(challenge(i -> {i==5;}));`

✓ `c) System.out.println(challenge((i) -> i==5));`

✓ `d) System.out.println(challenge((i) -> {return i==5;}));`

✗ `e) System.out.println(challenge((int i) -> i==5));`

✗ `f) System.out.println(challenge((int i) -> {return i==5;}));`

19 Maven: precendence of dependencies

- Given this POM file, which version of Guava will be selected? (choose 1)

```
<project …> …
  <dependencies>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>27.0-jre</version>
    </dependency>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>28.2-jre</version>
    </dependency>
  </dependencies>
</project>
```

a) 27.0-jre

b) 28.2-jre

c) Build error

19 Maven: precendence of dependencies

- Given this POM file, which version of Guava will be selected? (choose 1)

```
<project …> …
  <dependencies>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>27.0-jre</version>
    </dependency>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>28.2-jre</version>
    </dependency>
  </dependencies>
</project>
```

✗ a) 27.0-jre

✓ b) 28.2-jre

✗ c) Build error

20 Maven: dependencyManagement

- Given this POM file, which version of Guava will be selected? Guava 27.0.1-android is a transitive dependency of Truth 1.0. (choose 1)

```
<project …> …
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>com.google.guava</groupId>
        <artifactId>guava</artifactId>
        <version>27.0-jre</version>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.google.truth</groupId>
      <artifactId>truth</artifactId>
      <version>1.0</version>
    </dependency>
  </dependencies>
</project>
```

a) 27.0-jre

b) 27.0.1-android

c) Build error

20 Maven: dependencyManagement

- Given this POM file, which version of Guava will be selected? Guava 27.0.1-android is a transitive dependency of Truth 1.0. (choose 1)

```
<project …> …
  <dependencyManagement>
    <dependencies>
      <dependency>
        <groupId>com.google.guava</groupId>
        <artifactId>guava</artifactId>
        <version>27.0-jre</version>
      </dependency>
    </dependencies>
  </dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>com.google.truth</groupId>
      <artifactId>truth</artifactId>
      <version>1.0</version>
    </dependency>
  </dependencies>
</project>
```

✓ a) 27.0-jre

✗ b) 27.0.1-android

✗ c) Build error

21 Maven: dependencies in multi-module projects

- Given these POM files, which version of Guava will be selected? Guava 27.0.1-android is a transitive dependency of Truth 1.0. (choose 1)

```
<project …> … <artifactId>parent</artifactId> …
  <dependencies>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>28.2-jre</version>
    </dependency>
  </dependencies>
</project>
```

```
<project …> …
  <parent> …
   <artifactId>parent</artifactId> …
  </parent>
  <dependencies>
    <dependency>
      <groupId>com.google.truth</groupId>
      <artifactId>truth</artifactId>
      <version>1.0</version>
    </dependency>
  </dependencies>
</project>
```

a) 27.0.1-android

b) 28.2-jre

c) Build error

21 Maven: dependencies in multi-module projects

- Given these POM files, which version of Guava will be selected? Guava 27.0.1-android is a transitive dependency of Truth 1.0. (choose 1)

```
<project …> … <artifactId>parent</artifactId> …
  <dependencies>
    <dependency>
      <groupId>com.google.guava</groupId>
      <artifactId>guava</artifactId>
      <version>28.2-jre</version>
    </dependency>
  </dependencies>
</project>
```

```
<project …> …
  <parent> …
   <artifactId>parent</artifactId> …
  </parent>
  <dependencies>
    <dependency>
      <groupId>com.google.truth</groupId>
      <artifactId>truth</artifactId>
      <version>1.0</version>
    </dependency>
  </dependencies>
</project>
```

✗ a) 27.0.1-android

✓ b) 28.2-jre

✗ c) Build error

22 Last and prime

- Among the following, which one is a prime number?

```
a) Character.MAX_VALUE

b) Short.MAX_VALUE

c) Integer.MAX_VALUE

d) Long.MAX_VALUE
```

22 Last and prime

- Among the following, which one is a prime number?

✗ a) `Character.MAX_VALUE`

✗ b) `Short.MAX_VALUE`

✓ c) `Integer.MAX_VALUE`

✗ d) `Long.MAX_VALUE`

Sources and Thanks

- Andres Almiray
  http://andresalmiray.com/maven-dependencies-pop-quiz-results
- Heinz Kabutz
  https://www.javaspecialists.eu/archive/Issue245-Surprising--Cast.html
- José Paumard
- Joshua Bloch
  https://twitter.com/joshbloch/status/1330266449844310024
- Simon Roberts and Mikalai Zaikin (Java magazine)
  https://blogs.oracle.com/javamagazine/quiz-2
- EditX
  https://editx.eu/en/it-challenge/quiz-java-expert-0
- …