Actual exam

The real exam has 40 multiple choice (MC) questions, where each question has one correct answer out of five choices. You have 10 minutes reading time and 2 hours answer time. The exam is closed book, so bring only a type B pencil and an eraser. There is no booklet for rough work, as none is required.

The exam tests technical knowledge of the Java in the subject:

- words
- syntax
- behaviour
- use

There are a few "fill in the blanks" questions for standard plans
There are no coding questions; coding does not work with MC
There are no design questions; coding does not work with MC
There are no questions on FXML and CSS code.

Frequently asked questions

Can I have another 20 sample questions?

No. It was hard to find 140 Java questions to ask:

- 40 for the main exam
- 40 for the rescheduled exam
- 40 for the alternate exam
- 20 for the sample exam

Can I see a previous paper?

No. It was hard to find 100 Java questions to ask. The questions are re-used every semester, from the pool of 80 examined questions.

How do I study?

You need to know the detailed syntax, behaviour, and use of the new Java topics covered in the subject. You also need to know the patterns, processes and design rules covered in the subject. You should already know most of the material, because you used it to do the labs and assignments. If you skipped the non-assessed labs, then your task is a lot harder.

Is there a supplementary exam?

No.

Mini-exam

The sample exam has 20 questions. Here, I show four questions so you can test yourself on these four. The answers and reasoning for these four questions are shown on the next page, then the full sample exam of 20 questions is given, with answers at the end. You can look at all 20 questions now to see what they look like, or you can work through them after you have studied to check your knowledge.

The following question refers to this pattern:

```
<for each item>
     if (<item passes test>)
          return false;
return true;
```

4. What is the name of this pattern?

a) for-each loop
b) lookup
c) none
d) every
e) any

9. Which is ***not*** a benefit of object-oriented programming?

a) Objects hide implementation details
b) Dependencies are easier to manage
c) Objects better map onto the way the real world works
d) Classes are encapsulated inside of objects
e) Classes help to separate concerns.

10. Which of the following best describes polymorphism?

a) A variable can have multiple types.
b) An object can have multiple types.
c) A class can extend multiple superclasses.
d) A class can implement multiple interfaces.
e) Polymorphism literally means “many objects”.

The following question refers to the following line of code:

```
Customer customer = new Customer();
customer.addAccount(scanner.nextLine());
```

12. If a `NullPointerException` is thrown, which code is most likely to be `null`?

a) `customer`
b) `new Customer()`
c) `addAccount`
d) `scanner`
e) `nextLine`

Worked questions

The sample exam has 20 questions. Here, I show the reasoning behind four of them. You can look at all 20 questions now to see what they look like, or you can work through them after you have studied, to check your knowledge.

The following question refers to this pattern:

```
<for each item>
     if (<item passes test>)
          return false;
return true;
```

4. What is the name of this pattern?

a) for-each loop

No. This is more than just a for-each loop.

b) lookup

No. A lookup function returns a reference to the found object or null.

c) none

Yes. If even just one item passes the test, then it is false to say that none of them passed the test.

d) every

No. For "every" pattern, the second line changes to: if (!<item passes test>) because if even just one item fails the test, then it is false to say that every item passes the test.

e) any

No. For the "any" pattern, false and true should be swapped, because when asked the question "does any item pass the test?", it should return true if even just one item passes the test.

9. Which is ***not*** a benefit of object-oriented programming?

a) Objects hide implementation details

This *is* a benefit.

b) Dependencies are easier to manage

This *is* a benefit.

c) Objects better map onto the way the real world works

This *is* a benefit.

d) Classes are encapsulated inside of objects

While encapsulation is a benefit of OO, encapsulation refers to grouping related **methods** and **fields** together and hiding the inner workings of the object. Encapsulation does not refer to grouping classes into objects.

e) Classes help to separate concerns.

This *is* a benefit.

10. Which of the following best describes polymorphism?

a) A variable can have multiple types.

No. A variable `Rectangle r` has the type Rectangle and nothing else.

b) An object can have multiple types.

Yes. If `Rectangle` is a subclass of `Shape`, the object `new Rectangle(4, 5)` has type `Rectangle` and type `Shape`.

c) A class can extend multiple superclasses.

This is actually false.

d) A class can implement multiple interfaces.

While it is true that a class can implement multiple interfaces, polymorphism refers to an object having multiple types.

e) Polymorphism literally means "many objects".

No. However, polymorphism could be interpreted to mean "many shapes", suggesting that an object can pretend to be different shapes, or can pretend to be different types of thing (i.e. an object has many tipes).

The following question refers to the following line of code:

```
Customer customer = new Customer();
customer.addAccount(scanner.nextLine());
```

12. If a `NullPointerException` is thrown, which code is most likely to be `null`?

a) `customer`

No, this is definitely not null because it was assigned a definite value in line 1.

b) `new Customer()`

No, this can't be null because it is an instruction to create an object.

c) `addAccount`

If this method returns null, it doesn't matter because the result of this method is not being used, and therefore would not cause an exception.

d) `scanner`

Yes, we did not see in this small code snippet how the scanner was initialised, so it is possible the programmer forgot to initialise it. If the scanner is null and you try to invoke the nextLine method on an object that does not exist, that will throw a `NullPointerException`.

e) `nextLine`

This method never returns null. If there is no line, it throws a `NoSuchElementExecption`. And even if it were to return null, we are not trying to access one of its members. A `NullPointerException` occurs when we try to access a member of null. For example `foo.bar` or `foo.bar()` where foo is null.

20 sample exam questions

This sample exam has 20 questions. The correct answers are given at the end.

1. What does Java compare to find the right method header to match a method call?

a) method name
b) method name and number of parameters
c) method name, number and types of parameters
d) method name, number, type, and order of parameters
e) method name, number, type, order, and name of parameters

2. Which of the following statements is true about functions?

a) A function has side effects and can call a procedure.
b) A function has no side effects and can call a procedure.
c) A function has side effects and cannot call a procedure.
d) A function has no side effects and cannot call a procedure.
e) A function can have many parameters, but must have a return type of `void`.

3. What String function returns a sub-string of a string?

a) `toSubString`
b) `toSubstring`
c) `subString`
d) `substr`
e) `substring`

The following question refers to this pattern:

```
<for each item>
   if (<item passes test>)
      return false;
return true;
```

4. What is the name of this pattern?

a) for-each loop
b) lookup
c) none
d) every
e) any

5. When do you need braces around a chunk of code?

a) When there is more than one statement
b) When the code goes over several lines
c) When the code is complex
d) After an **`if`**
e) After an **`if`** or **`for`**

The following question refers to the method shown below.

```
private Worker worker(int id) {
    for (Worker worker : workers)
        if (worker.matches(id))
            return worker;
    <xxx1xxx>;
}
```

6. What line replaces <xxx1xxx>?

a) `break`
b) `return`
c) `return new Worker()`
d) `return null`
e) `return "Not found"`

The following question refers to this code:

```
private boolean any() {
    for (Element e : elements)
        if (<test>)
            return <xxx1xxx>;
    return <xxx2xxx>;
}
```

7. What line replaces <xxx2xxx>?

a) `e`
b) `element`
c) `null`
d) `true`
e) `false`

8. In terms of duration, which phase of software engineering is most affected by applying object-oriented design principles?

a) Analyse
b) Design
c) Code
d) Debug
e) Extend

9. Which is ***not*** a benefit of object-oriented programming?

a) Objects hide implementation details
b) Dependencies are easier to manage
c) Objects better map onto the way the real world works
d) Classes are encapsulated inside of objects
e) Classes help to separate concerns.

10. Which of the following best describes polymorphism?

a) A variable can have multiple types.
b) An object can have multiple types.
c) A class can extend multiple superclasses.
d) A class can implement multiple interfaces.
e) Polymorphism literally means “many objects”.

11. Which interface does an observer implement to respond to button clicks?

a) `ActionEvent`
b) `SelectionEvent`
c) `ActionListener`
d) `ChangeListener`
e) `EventHandler`

The following question refers to the following line of code:

```
Customer customer = new Customer();
customer.addAccount(scanner.nextLine());
```

12. If a `NullPointerException` is thrown, which code is most likely to be `null`?

a) `customer`
b) `new Customer()`
c) `addAccount`
d) `scanner`
e) `nextLine`

13. Which best defines the **`throws`** keyword in Java?

a) It defines an exception.
b) It throws an exception.
c) It indicates that an exception might be thrown.
d) It sets the exception message.
e) This keyword does not exist in Java.

14. Which of the following lines of code defines a new JavaFX application class?

a) `class MyApplication extends Application`
b) `class MyApplication inherits Application`
c) `class MyApplication implements Application`
d) `class MyApplication throws Application`
e) `class MyApplication uses Application`

15. Which of the following is a valid use of the `@Overrides` annotation?
a) On an abstract method.
b) To link a controller to an FXML file.
c) On an interface method.
d) On a method that must be overridden.
e) On an implementation of an interface method.

16. What code registers a listener `l` for when the selection of a toggle group `g` changes?
a) `g.getToggleGroup().selectedItemProperty().addListener(l);`
b) `g.selectedToggleProperty().addListener(l);`
c) `g.getSelectionModel().getSelectedItem().addListener(l);`
d) `g.getSelectionModel().selectedItemProperty().addListener(l);`
e) `g.selectionModel().addListener(l);`

17. Which of the following is true about `ArrayList`?
a) Inserting an item at the beginning of an `ArrayList` is fast.
b) Modifying an item in the middle of an `ArrayList` is fast.
c) `ArrayList` is like `LinkedList` except that an `ArrayList` has a fixed size.
d) Adding an item to the end of an `ArrayList` is normally very slow.
e) All of the above.

18. How many pointers need to be set to delete an element from a linked list?
a) 0
b) 1
c) 2
d) 3
e) 4

19. How does a TableView `tv` get linked to an observable list `l`?
a) `tv.setList(l)`
b) `l.addListener(tv)`
c) `tv.addModel(model)`
d) `tv.addTable(table)`
e) `l.attach(tv)`

20. Which is not a supported way to define an observer for a button click in Java 7?
a) As an inner class.
b) As an anonymous inner class.
c) As a lambda expression
d) As a method annotated with @FXML
e) As a top level class.

Answers

1d, 2d, 3e, 4c, 5a, 6d, 7e, 8e, 9d, 10b,
11e, 12d, 13c, 14a, 15e, 16b, 17b, 18b, 19a, 20c