



AARHUS UNIVERSITET

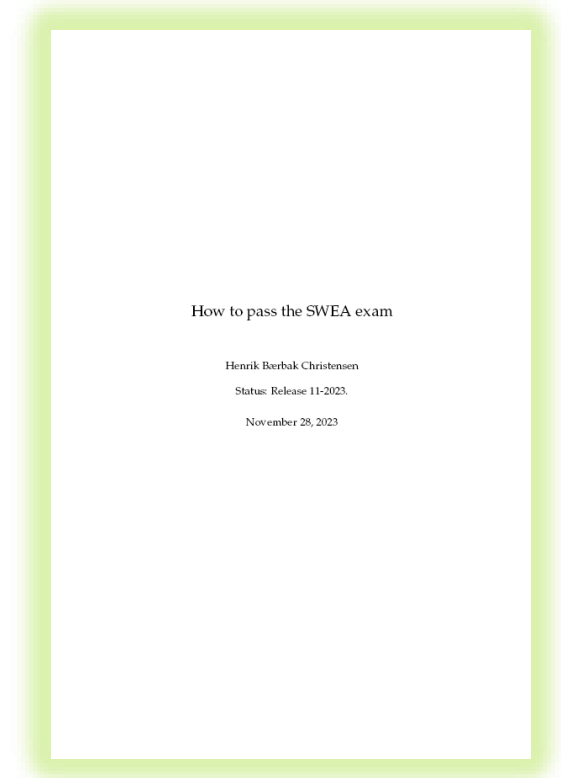
Software Engineering and Architecture

The Exam



- Release Nov 2023!
 - Feedback is very welcome
- On the last week plan
- Best: Rehearse during TA classes !
 - Many of you have already done so!

Exam guide





- A few before Christmas, the rest in January
 - Censors:
 - Magnus Madsen,
 - Clemens Klokmose,
 - Niels Olof Bouvin
- The examination plan is on BrightSpace
 - The faculty one is out-of-date!

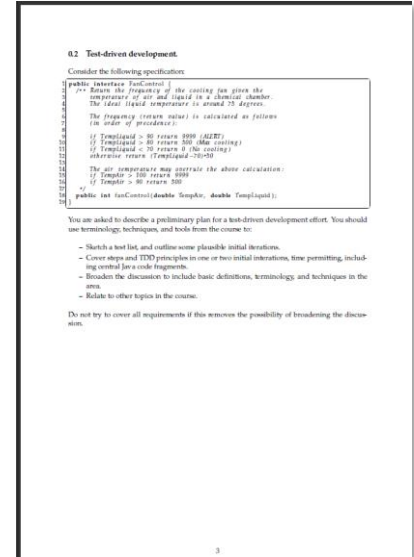
Dispensations

- **Important about those applying for dispensation**
 - (Requesting the preparation time to be extended)

• **Apply early!!! (= NOW!!!)**

- Reason:
 - Due to logistics, *all students with a dispensation are grouped into one or two examination days*
 - ***Thus it is vital that we know the number early to make usable planning of the examination days!!!***
- Days for ordinary exams *will* proceed with ~30-35 min slots!

- Examination format: Physical + Zoom based
- Physical
 - You get your exam exercise **on paper**.
 - The exam is physical in the sense
 - In the same room with me and censor
 - We talk 😊
- Zoom based
 - You present your solution produced during the preparation time using *Zoom shared screen*
 - *Code stuff in particular...*
 - And of course the whiteboard if you wish





- Be in the **waiting room of exam Zoom session**
 - Best before your exam actually starts 😊
- Have your ***study id card*** ready for verification
- Draw a ***random exercise***
 - *The exercise is a single A4 physical piece of paper*
- **Read, understand, and ‘start solution’**
- I do not expect a complete and polished solution!
- *Better to understand the exercise than show a solution to something else than the exercise*
 - Not a disaster, but it takes valuable time out of the examination



The Exam Exercise

- It should be obvious, but still...

Any form of copying and distribution of an exam exercise is exam cheating. Exam exercises are secrets.

Any instances of copies on any platform should be reported to me!

After exam, you must delete produced material.



How: The Preparation

- During the preparation period (30 – 35 min)
- **Use your own laptop to overall ‘solve’ (parts of) the exercise**
 - Prepare a UML diagram
 - Prepare a EC and test case table
 - Prepare code skeleton / partial solution
 - To be presented at the exam...
- **Resources available: “Everything” except any help from 3rd party**
 - No help from other persons or ChatGPT, or other AI tools



The Exam

- We sit together – we talk and discuss
- Censor, I and you are on Zoom and you **share your screen**
- Present your *developed code/material* and talk us through what you have made – and we ask questions etc.
- 11-12 minutes only
 - ‘but ... I only have started’



- After the exam we will
 - Of course give you a grade, explain the reasoning behind it, and provide advice wrt. future exams
 - **Require you to delete produced material**
 - As any distribution of your developed material and the exercise itself is considered *exam cheating*.



- Starting E2022 the mandatory points are *not counting towards the final grade...*
- **Mandatory point 60% of 430 = 258**
 - **Handin of all 10 exercises**
 - Is required to attend the exam
- Final grade is based upon oral exam *only*.



AARHUS UNIVERSITET

Some Stuff You May Meet



Realistic Exam Exercises

AARHUS UNIVERSITET

- All exam exercises are modelled after realistic systems!
 - No exercises ala
 - Use Broker to implement `objectFoo.doSomething(27,"Fisk");`
- Which means you *will run into Java classes like*
 - ZonedDateTime

```
ZonedDateTime time = ZonedDateTime.now(); // Get Current time and date
```
 - PrintStream

```
private PrintStream logfile;  
logfile.println("
```
- And code inspired by HotStone, MiniDraw, TeleMed, PayStation...



Realistic Exam Exercises

- All exam exercises are modelled after realistic systems!
 - ... which means it will contain realistic terms from IT and domain
- I expect you to know what terms like
 - GPS, Mobile phone, app, SMS/Text message, database, caching, server, blood pressure, inventory,
- ... means
 - Often terms are explained or written in Danish
 - Tenant (Da: “lejer af lejlighed”)
 - Caching (That is, store a local copy of data so a remote fetch is not required)



AARHUS UNIVERSITET

Hints and Advice

- Use the provided exam question set to *rehearse* the exam format
 - Pick an exercise from the example set
 - Spend 20-25 minutes
 - Finding solution techniques
 - Partially solving it **using your laptop with appropriate tools**
 - Spend 12 minutes doing a ‘dry-run’ exam with your group
 - Let your other groups members act examiner
 - Rotate and repeat 😊



Solve the Exercise

- SWEA is not just defining three interfaces with a method in each...
 - I see a lot of ‘solutions’ at the exam of this type
- **Where are they used? Where are those methods called? By Whom?**
 - Ala ‘theObject.doSomething(....)’



Appropriate Tools

- Experience from last year's Zoom based exams
- **Word is not a good editor for code !!! !!! !!!**
- **Free-hand drawing on a laptop does not work!**
 - A mouse is not a pen. Pen = fine muscles; mouse = coarse...
- Use IntelliJ?
 - Be sure you have tried it out before you do. Do not let it spoil the exam because it insists your code does not compile



Appropriate Tools

- Find a *good code editor that is not an IDE*
 - *Gedit, Emacs, SublimeText, EditPlus, Notepad++, etc.*
 - It does not choke on pseudo code but knows indentation, syntax highlight etc.
- Find a good spreadsheet type program
 - For making EC tables and test case tables
 - Filling out the Clean code analysis
- Find a decent way of drawing UML
 - An editor – or – draw it on the whiteboard at exam



Test the Technique

- Zoom Share Screen is essential and must be tested
 - Mac's require a setting to be changed and the machine rebooted
 - Ok, so the first 4 minutes of your exam is spent on that
 - **Bad grade due to not testing that aspect in advance?**
 - Screen sharing does not work on certain Ubuntu versions!
 - **Bad grade due to not testing that aspect in advance?**

Increase the
font size! 😊

- *As you can obviously see in line 17, I have coded the State pattern*
- *(No, I cannot)*

```

@overrid
public Status attackCard(Player playerAttacking, Card attackingCard, Card defendingCard) {
    Status status = null;
    // Handle attacks
    if (playerAttacking == Player.FINDUS) {
        if (attackingCard.getOwner() != Player.FINDUS) {
            status = Status.NOT_OWNED;
        } else {
            if (defendingCard.getOwner() == Player.FINDUS) {
                status = Status.ATTACK_NOT_ALLOWED_ON_OWN_POSITION;
            } else {
                if (player.FINDUS.isPlayerTurn()) {
                    status = Status.NOT_PLAYERS_TURN;
                } else {
                    if (attackingCard.isActive()) {
                        status = Status.ATTACK_NOT_ALLOWED_FOR_NON_ACTIVE_POSITION;
                    } else {
                        StandardCard mc = (StandardCard) attackingCard;
                        StandardCard dc = (StandardCard) defendingCard;
                        mc.lowerHealth(dc.getAttack());
                        mc.lowerHealth(dc.getAttack());
                    }
                }
            }
        }
        // remove defeated minions
        if (dc.getHealth() <= 0)
            dc.removeSelf();
        if (mc <= 0)
            mc.removeSelf();
        // toggle the action flag of attacker
        mc.setActive(false);
        status = Status.OK;
    }
    // Handle defense
    if (attackingCard.getOwner() != Player.FINDUSEN) {
        status = Status.NOT_OWNED;
        if (defendingCard.getOwner() == Player.FINDUSEN) {
            status = Status.ATTACK_NOT_ALLOWED_ON_OWN_POSITION;
        } else {
            if (player.FINDUSEN.isPlayerTurn()) {
                status = Status.NOT_PLAYERS_TURN;
            } else {
                if (attackingCard.isActive()) {
                    status = Status.ATTACK_NOT_ALLOWED_FOR_NON_ACTIVE_POSITION;
                } else {
                    StandardCard mc = (StandardCard) attackingCard;
                    StandardCard dc = (StandardCard) defendingCard;
                    // Player attacks the card
                    mc.lowerHealth(dc.getAttack());
                    defender.lowerHealth(mc.getAttack());
                }
            }
        }
        // remove defeated minions
        if (dc.getHealth() <= 0)
            dc.removeSelf();
        if (mc.getHealth() <= 0)
            mc.removeSelf();
        // toggle the action flag of attacker
        mc.setActive(false);
        status = Status.OK;
    }
    return status;
}
    
```


- ***At the exam you are only allowed to bring whatever you produce in the preparation.***
- The exception to this rule
 - The Invoker code base involves boilerplate code that is the same always (demarshalling 'request' into its request object)
 - It is OK to copy that from a template you prepare in advance...

```
@Override
public String handleRequest(String request) {
    // Do the demarshalling
    RequestObject requestObject = gson.fromJson(request, RequestObject.class);
    JSONArray array = JsonParser.parseString(requestObject.getPayload()).getAsJSONArray();

    ReplyObject reply;
    /* As there is only one TeleMed instance (a singleton)
       the objectId is not used for anything in our case.
    */
    try {
        // Dispatching on all known operations
    }
}
```

operation name, arguments

```
public class RequestObject {
    private final String operationName;
}
```





- Some students are not nervous at exams.
 - How do you do that???
- Some are...
 - Learn to **use** your nervousness, instead of trying to avoid it...
- *Perhaps it is not a problem, it is a condition of life...*
 - *Problems you solve, conditions you learn to live with...*



AARHUS UNIVERSITET

Paths to Failure

Common failing reasons



- Commonalities in failing the SWEA exam
- Level 1
 - Coding mastery
- Level 2
 - Shallow learning / "parrot knowledge"
 - Reading a specification causes problems
 - A few in the classic category: read the book also...

- Too many do not fail SWEA, but fail to have sufficient *coding practice* at my exam
 - Writing ‘nonsense’ and cannot see the problem

```
assertThat( pws.parse(String line), is(true) );
```

```
private ParseWorkSpecification pws;

@Before
public void setup() {
    ParseWorkSpecification pws = new ParseWorkSpecification();
}

@Test
public void shouldDoSomething() {
    assertThat(pws.parse("Wed 1"), is(true));
}
```

What is a constructor?



- "Parrot" knowledge
 - You can learn the *visual appearance* of Strategy pattern UML
 - You can learn the *visual appearance* of Strategy code template
 - You can learn the *common names* used in Strategy

 - Without any clue of what Strategy **is**
- Beware of it!

Pattern Matching

- Similar – ”pattern matching” shallow learning
 - Make ECs correctly for an interval in the exercise
 - But fail to mention what heuristics that has been used (range)
 - Fail to know how the range heuristics is formulated
 - Fail to know what an EC is at all
 - Make ‘ECs’ that are test cases
 - 7 [b1]
- Conclusion
 - Learned the ”template”, then put arbitrary stuff into it ☹️



Exercise Reading

- You will have to read the exercise and understand at least a bit of it

- **Read the code fragments / interfaces – they are the key question**



And of course the usual

AARHUS UNIVERSITET

- A few fail of the same reason as in other courses
 - Have we read the same book and coded the same exercises???



- You have to master *some* of this course to pass it
 - You need some level of mastery of Java or OO language
 - Which you learn by *doing it*
 - You need to understand what is *behind* the templates
 - Which you learn by *doing it*
 - You need to be able to read a small specification
 - Which you do by *reading it*



AARHUS UNIVERSITET

SWEA 2023 signing off...

It has been a pleasure flying with you