UNIVERSITY OF ICELAND

HBV401G SOFTWARE DEVELOPMENT

13. Final Exam Preparation

Matthias Book
Spring 2022

FACULTY OF INDUSTRIAL ENGINEERING, MECHANICAL ENGINEERING AND COMPUTER SCIENCE

Quiz #12 Solution: Equivalence Classes & Boundary Values

- We want to test the method `String read(String filename, int n)`, where
  - the filename should consist of 1 to 6 characters (numbers or letters)
  - the number of requested lines should be greater than 0 and less than 1000
- Indicate which statements are well-constructed test cases for the "valid input" or "invalid input" equivalence classes, and which are poorly constructed test cases:

a) `String s = read("file", -1);` "invalid input" equivalence class

b) `String s = read("file", 0);` "invalid input" equivalence class

c) `String s = read("file", 1);` "valid input" equivalence class

d) `String s = read("filename", 999);` "invalid input" equivalence class

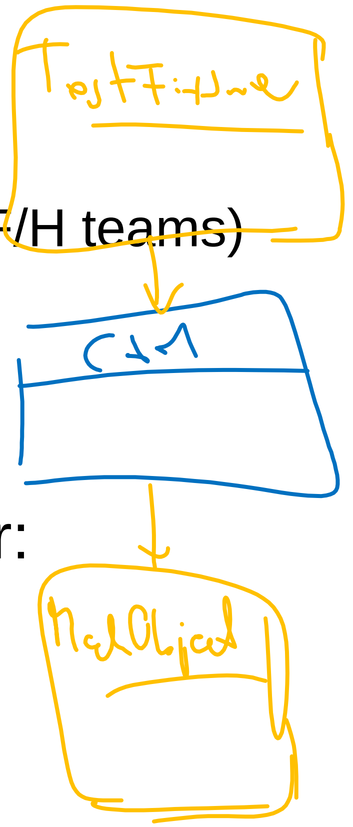
e) `String s = read("filename", 1000);` poorly constructed test case

f) `String s = read("filename", 1001);` poorly constructed test case

Recap: Team Assignment #4: Software Tests

- By **Sun 27 Mar**, submit in Canvas a **test document (PDF)** showing:
  - **A test fixture** you have implemented for one of your controller components
    - Provide source code of test fixture based on Junit
  - **A mock object** you have implemented to simulate a storage component (for D/F/H teams) or another team's component (for T teams)
    - Provide documented source code of mock object

- On **Wed 30 Mar**, present and **explain** your test document to your tutor:
  - Why did you choose the test cases (inputs and expected results) you did?
  - Why do you believe the behavior covered by your test cases is sufficient?
  - How does your mock object simulate the mocked object's behavior?

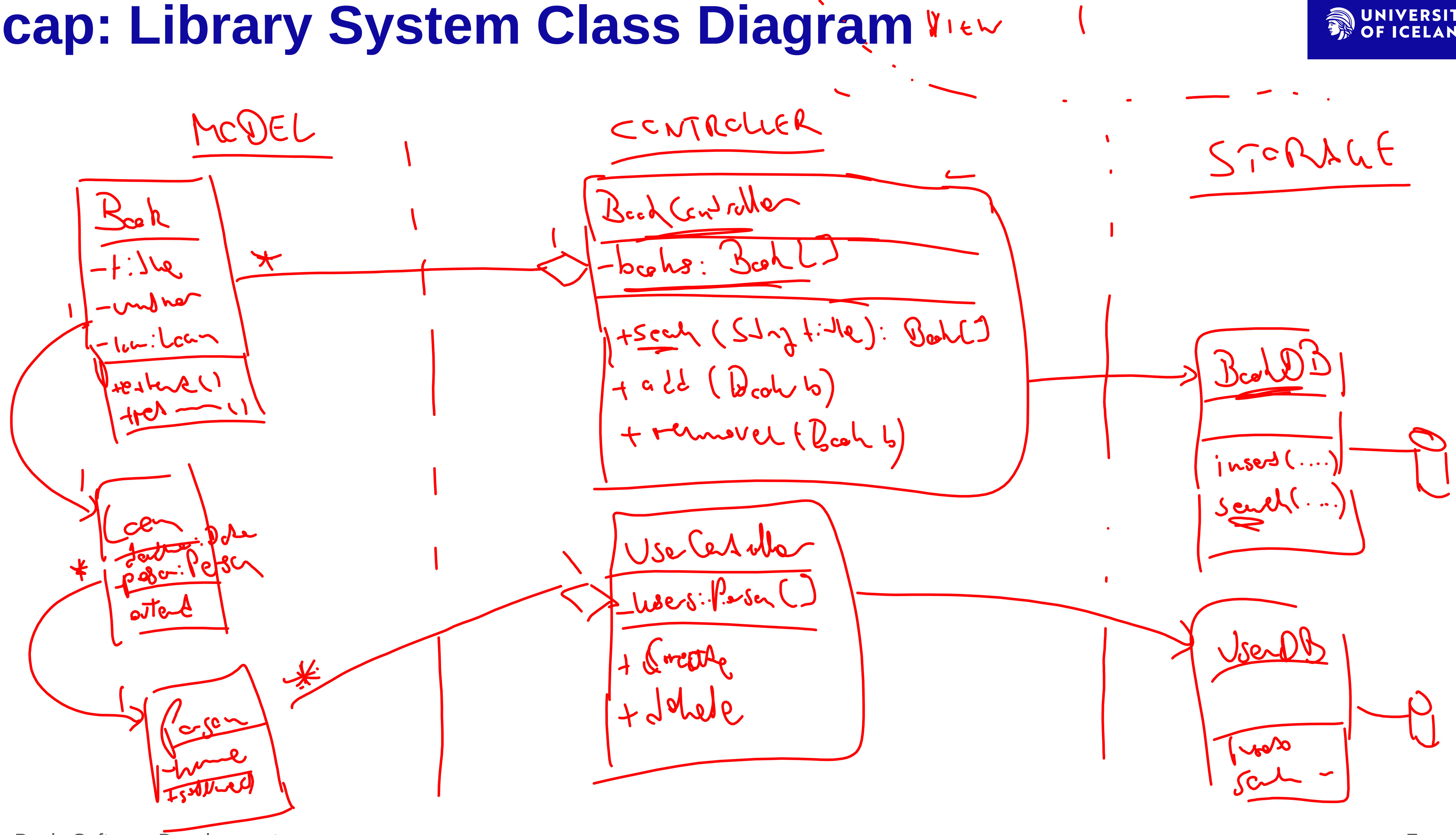


Recap: Team Assignment #4: Software Tests

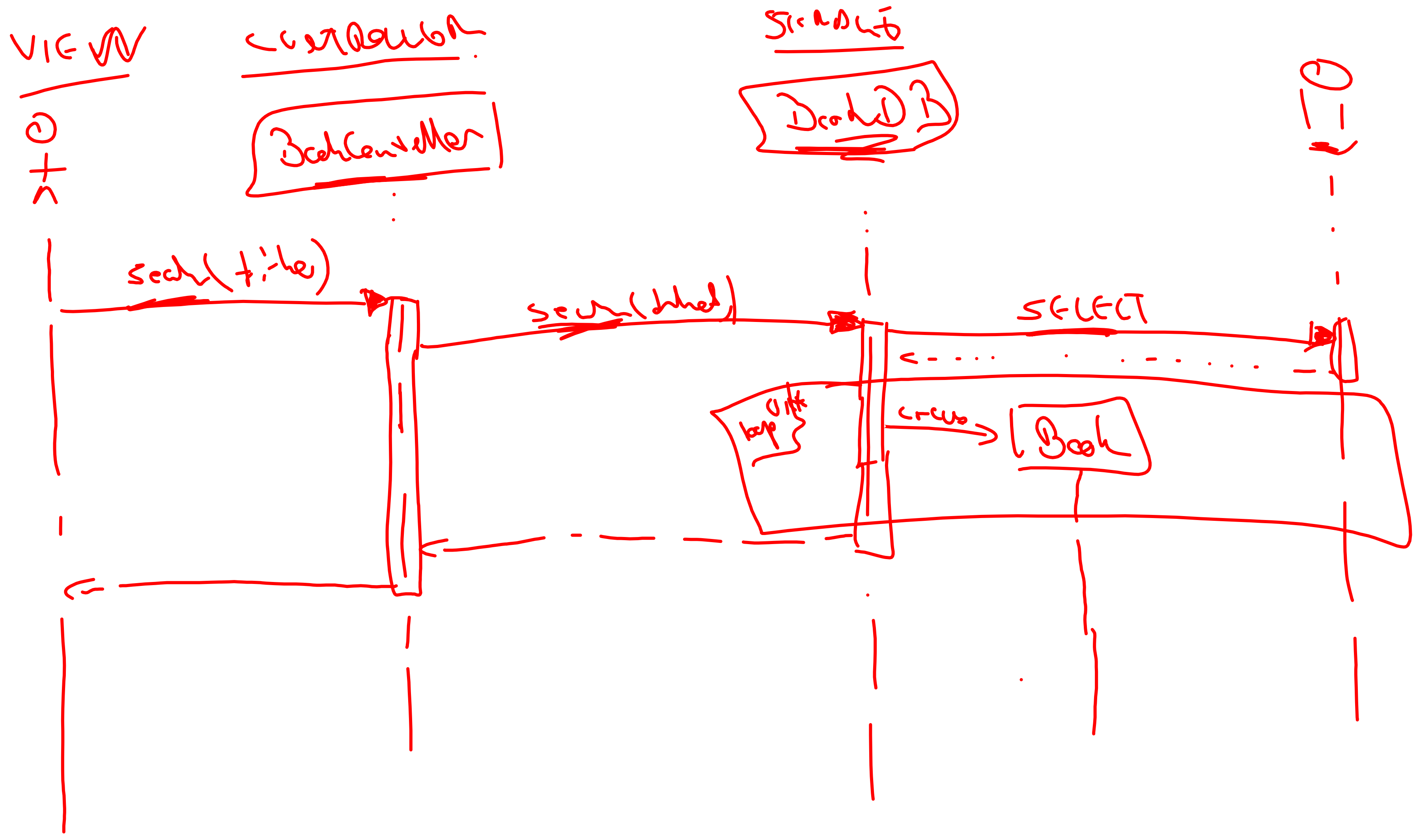
- **Grading criteria** (33.33% each)
  - The **test fixture** provides the proper **structural framework** for testing a controller’s method
    - It ensures a clean environment for each test using `@Before` and `@After` methods and passes suitable mock object instances to the controller under test.
    - The controller under test would not need to be changed between testing and productive use.
  - The **test fixture** exposes a controller's method to plausible **test cases**
    - The tests feed suitable input to the method for asserting properties of its output that express the method’s expected behavior in different scenarios (e.g. successful and unsuccessful search).
    - The tests do not test behavior of the mock objects.
  - **Mock objects** are used in a plausible way.
    - The mock objects simulate (rather than realize) complex behavior that the controller under test relies on.
    - Several mock objects are used to simulate different behaviors.

Sample Solutions

Recap: Library System Class Diagram

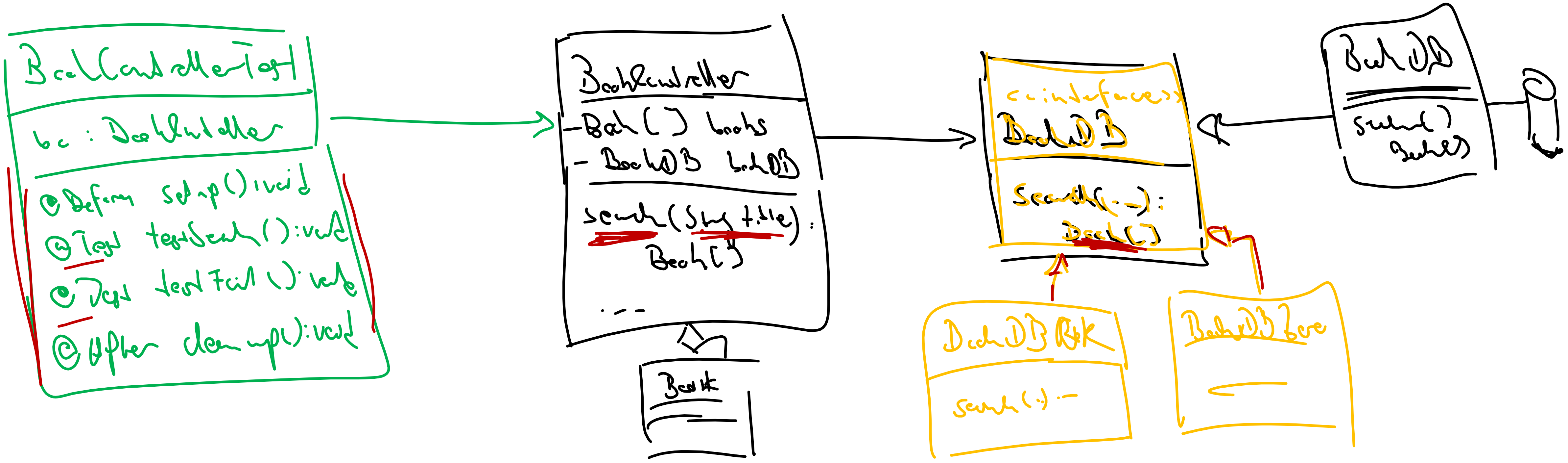


Recap: Library Search Sequence Diagram



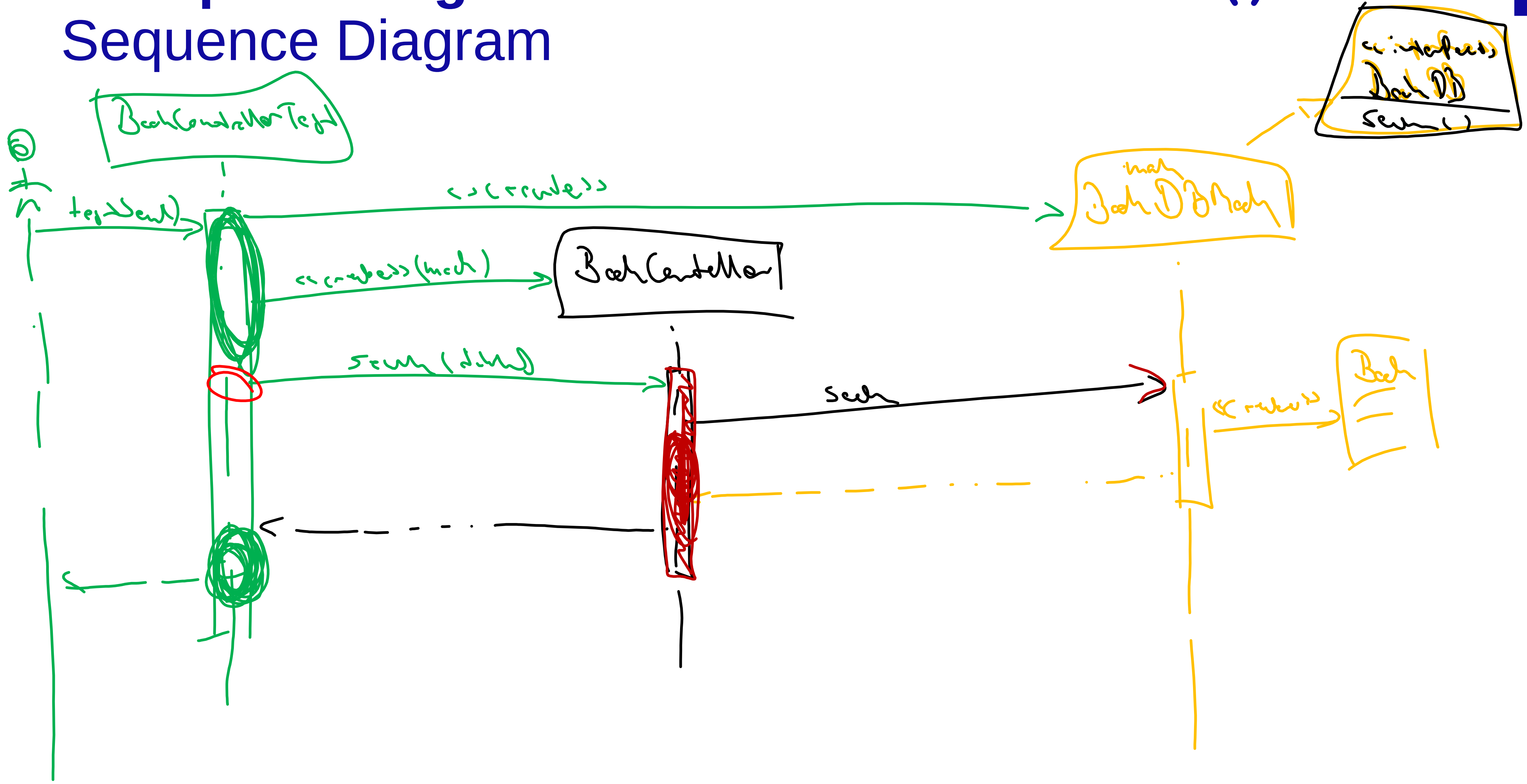
Recap: Testing `BookController.search()`

Class Diagram



Recap: Testing `BookController.search()`

Sequence Diagram



Break

Course Review

Week	Mon: Lectures	Wed: Team Consultations	Sun: Assignments due
1	Introduction, Software Processes		Team formation (16 Jan)
2	Requirements Elicitation	Project Topic Guidance	
3	Effort Estimation	#1a: User Stories Draft	#1a: User Stories (30 Jan)
4	Agile Project Planning	#1a: User Stories Presentation	
5	Modeling	#1b: Planning Poker	#1b: Product Backlog (13 Feb)
6	Object-Oriented Analysis	#1b: Backlog Presentation	
7	Object-Oriented Analysis/Design	#2: Domain Model Draft	#2: Domain Model (27 Feb)
8	Object-Oriented Design	#2: Domain Model Presentation	
9	Object-Oriented Design Principles	#3: Design Model Draft	#3: Design Model (13 Mar)
10	Software Testing: Automation	#3: Design Model Presentation	
11	Software Testing: Mock Objects	#4: Test Cases Draft	#4: Test Cases (27 Mar)
12	Software Testing: Coverage	#4: Test Cases Presentation	
13	Final Exam Preparation	#5: Final Product Draft	
14	*Spare lecture slot (if needed)*	EASTER BREAK	
15		#5: Final Product Presentation	#5: Final Product (24 Apr)

Conclusion: The Nature of Software Development

**“Because software is embodied knowledge,
and that knowledge is initially dispersed, tacit, latent, and incomplete,
software development is a social learning process.”**

Howard Baetjer, Jr.: Software as Capital. IEEE Computer Society Press, 1998

Further Reading
(if you’re interested – not required for exam preparation)

- **Agile software development**
  - ❖Dan Pilone, Russ Miles: Head First Software Development. O’Reilly, 2008
  - Mike Cohn: Agile Estimating and Planning. Prentice Hall, 2005
- **Requirements engineering**
  - Dean Leffingwell: Agile Software Requirements. Addison-Wesley, 2011
  - ❖Karl Wiegers, Joy Beatty: Software Requirements. Microsoft Press, 2013
- **Object-oriented analysis and design**
  - Craig Larman: Applying UML and Patterns. Prentice Hall, 2004
  - Russ Miles, Kim Hamilton: Learning UML 2.0. O’Reilly, 2006
  - ❖Eric Freeman, Bert Bates: Head First Design Patterns. O’Reilly, 2004
- **Programming in the large**
  - ❖Joshua Bloch: Effective Java. Addison-Wesley, 2008
  - Steve McConnell: Code Complete. Microsoft Press, 2004

Recap: Cluster Assignment #5: Final Product (Presentations & Submissions for Clusters with a T Team)

- On **Wed 20 Apr**, present **your cluster's integrated product** to your classmates
  1. A demonstration of your cluster's integrated product (live, ~5 min)
  2. An overview of your cluster's system architecture (slides, ~5 min)
  3. A retrospective on your cluster's project work (slides, ~5 min)
  4. Q&A (~5 min)

- By **Sun 24 Apr**, submit in Canvas:
  1. A ZIP archive with the code of your cluster's integrated product
  2. A PDF document with the slides of your cluster's presentation

- Note the presentation comes before the submission for this assignment!
  - Don't fix/optimize your product after the presentation; your presented state counts!
  - ➢ Simply submit what you presented

Recap: Cluster Assignment #5: Final Product
(Presentations & Submissions for Clusters without a T Team)

- On **Wed 20 Apr**, present **your team's product** to your classmates
    1. A demonstration of your team's product (live, ~2 min)
    2. An overview of your team's system architecture (slides, ~2 min)
    3. A retrospective on your team's project work (slides, ~2 min)
    4. Joint Q&A with other teams in cluster (~2 min)

- By **Sun 24 Apr**, submit in Canvas:
    1. A ZIP archive with the code of your team's component
    2. A PDF document with the slides of your team's presentation

- Note the presentation comes before the submission for this assignment!
    - Don't fix/optimize your product after the presentation; your presented state counts!
    - ➢ Simply submit what you presented

Recap: Cluster Assignment #5: Presentation Schedule

- Clusters present their products in the Zoom classroom at https://eu01web.zoom.us/j/62847273071 on **Wed 20 Apr**
  - 15:00-15:20 Cluster 1
  - 15:25-15:45 Cluster 2
  - 15:50-16:10 Cluster 3
  - 16:15-16:35 Cluster 4
  - 16:40-17:00 Cluster 5
  - 17:05-17:25 Cluster 6
  - 17:30-17:50 Cluster 7
  - 17:55-18:15 Cluster 8
- All teams are encouraged to attend presentations of other clusters as well
  - to learn from other teams' experiences
  - to give other teams an audience

Grading Policies and Final Exam

Recap: Overall Grading Scheme

- All contributing factors are graded on a scale of 0…11
  - Grading criteria for assignment deliverables and individual presentations
  - Questions in in-class quizzes
  - Questions on final exam
- All factors are averaged using the published weights, without rounding
  - Exceptional performance (11) on some factors can outweigh lower performance elsewhere
- Resulting final course grade is rounded to nearest half point
  - In case of a passed exam, final course grade is capped at 10.0
  - In case of a failed exam, final course grade is capped at 4.5

- Need a passing project grade to be allowed to final exam
  - If project grade is not sufficient, need to do a project again next year
- Need a passing project and exam grade to pass the course
  - Failed exam can be re-taken during the resit period, and re-taken next year if failed again
  - No need (and not possible) to redo a passed project
    - Project grade remains valid for only one year though

Recap: Project Grading

- The project grade depends on the **deliverables** submitted and the **presentation** given for each assignment.
  - Grading criteria will be published together with assignment.
- All team members receive same grade for **deliverables** submitted for an assignment
  - Each assignment weighs **17%** of project grade
- Over the course of the semester, each team member must lead the **presentation** of at least one assignment to tutor
  - Focus: Don’t just tell us what you did, but *why* you decided to do it this way.
  - The presenting team member receives an individual grade for their presentation (“6th assignment”, weighs **15%** of project grade)
    - A presentation must be given by a sole student to count for their presentation grade.
      - Joint presentations are allowed but won’t be counted for anyone’s presentation grade.
    - If a student presents multiple assignments, their best presentation grade counts.
- The resulting project grade weighs 30-70% of the final course grade.

Recap: Optional Peer Feedback on Assignments

- **Process:**
  - Submit anonymized versions of your team's assignment drafts on Canvas
  - Give anonymous feedback on the submissions assigned to you
  - Rate the quality (depth, helpfulness) of the feedback you have received from fellow students
- **Grading policy**
  - Grades of Assignments #1-5 plus Presentation Grade add up to 100% project grade
  - Quality of feedback you provide determines a bonus on project grade weighing extra 7.5%
    - Graded on scale (0, 5…11), with 0 for feedback that was not handed in

Recap: Final Exam

- **Date, Time & Location:**
  Tue 26 Apr 13:30–16:30 on campus
- **Focus:** Understanding of software engineering concepts and methods
- **Scope:** Lecture slides
  - Note: The spoken part is relevant too!
- **Style:** Digital exam
  - Gradescope or Inspera (TBA)
- **Weight:** 30-70% of final course grade

- **Tools:**
  - One double-sided A4 sheet of your notes
    - i.e. handwritten in your own ink
    - No photocopied notes
    - No printed lecture slides
  - Dictionary (in book form)
  - Laptop to access the exam system
  - No collaboration with others allowed!
- **Questions:**
  - 10 questions with same weight
  - Answers that exceed expectations can make up for deficiencies elsewhere
- **Answers:**
  - in English; in your own words
  - short paragraphs of whole sentences
  - possibly small code fragments / models

Answers in Your Own Words

- **No collaboration or use of non-HBV401G resources!**

- I will place strong emphasis on answers being **in your own words.**
  - Answers that reproduce phrases on my slides will receive reduced points.
  - Answers that closely reflect other students' answers or content from external sources will receive reduced (possibly zero) points.
  - I reserve the right to use plagiarism detection software.

- If I have the strong impression that parts of an exam are not your own work, I reserve the right to invite you to an oral exam to determine the exam grade.

- **Don't stress out** about these rules – if you're working on your own, you're fine.

Recap: Optional In-Class Quizzes

- To encourage class attendance: Small quizzes in most lectures
  - To be submitted online on the same day as the lecture
  - Graded on scale (0, 5…11), with 0 for any quizzes that are not handed in
    - two worst quiz grades will be ignored

- Grades of all in-class quizzes will be averaged into one final **quiz grade**
- Quiz grade can improve your final exam grade:
  - Final exam questions add up to 100% exam grade
  - Quiz grade will be counted as an optional additional final exam question worth 7.5%

➢ If you don't participate in any quizzes, you can still get top marks on the exam

➢ If you do participate in quizzes, the quiz grade can improve your exam grade by up to 11 on a 7.5% question, and thereby make up for deficiencies elsewhere

Recap: Grade Weights

- Assessment of team contribution
  - At the end of the semester, all team members assess how much each of their teammates contributed to the project.
  - Contribution votes are normalized to obtain each team member's contribution factor.

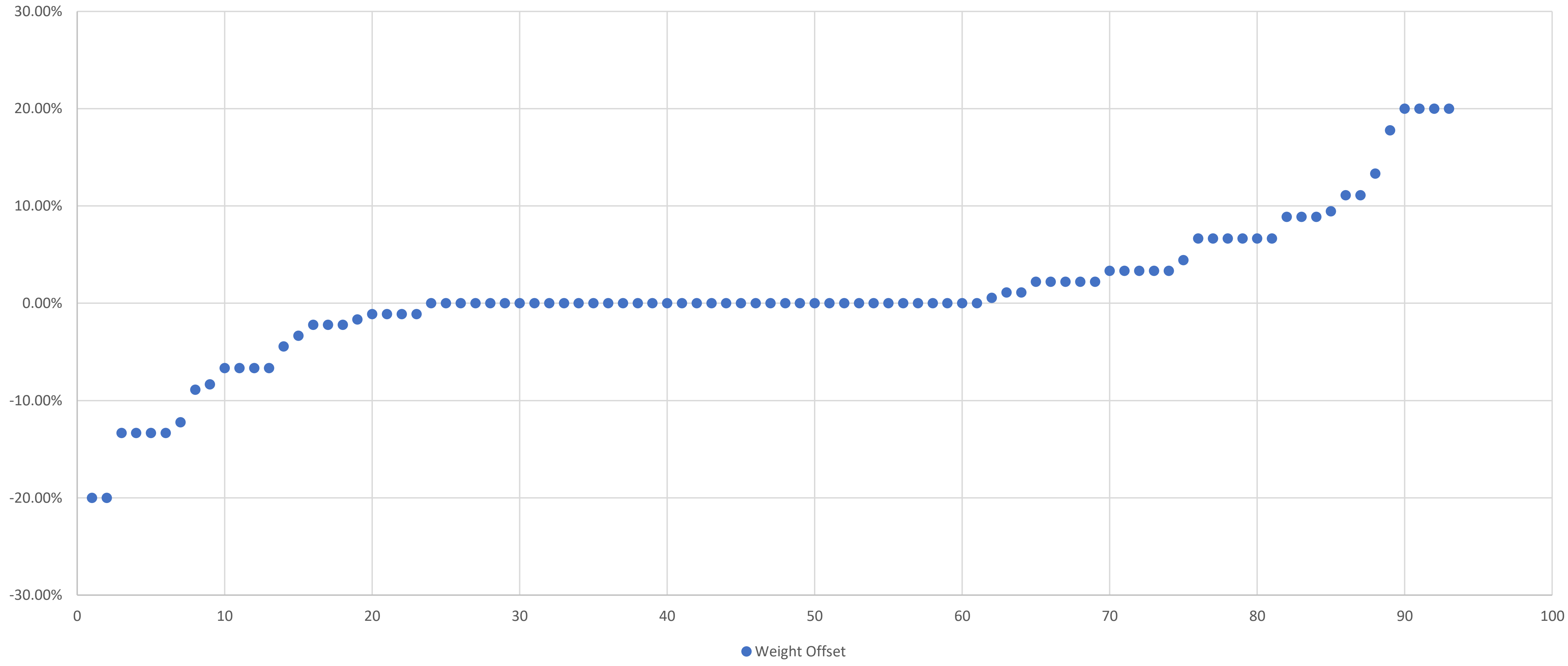
- Depending on team members' individual contributions, the weight of their project and final exam grades will be individually adjusted between 30% and 70%
  - Below-average contribution → lower weight of project grade, higher weight of exam grade
  - Average contribution → equal weight of project and exam grade (50:50)
  - Above-average contribution → higher weight of project grade, lower weight of exam grade

- In rare cases where the above rules would punish someone who contributed above average, or reward someone who contributed below average, the traditional 50:50 weight distribution is used instead.

Peer Assessment

- Each team member can assess the contribution of each of their teammates:
  - +++ : contributed **much more** than others
  - ++ : contributed **more** than others
  - + : contributed **slightly more** than others
  - o : contributed **same** as others
  - - : contributed **slightly less** than others
  - -- : contributed **less** than others
  - --- : contributed **much less** than others
- Assessments for each team member will be averaged,
  and the assessments will be normalized across all team members
  - ➢ if you and your team mates all rate each other as “+++”, your normalized contribution will be “o”
- Ratings should reflect a fair assessment of your own and your teammates’ contribution.
  - Ultimate decision of how assessments are counted remains with tutors.
- Submit your ratings in the “Peer Assessment” assignment on Canvas by **Sun 24 Apr**.

Example: HBV501G 2019 Project Weight Offsets



Break

Sample Exam Questions

- The following review questions are representative of the types of questions that could be on an exam.
- The answers to some of the following review questions can be found immediately on the lecture slides.
  - Caution: An open-book exam will obviously contain very few of these!
- Most questions require more individual reasoning and/or application of learned knowledge.
  - Be precise, justify your answers.

- **Read exam questions carefully**
  - Are you asked to explain or give an example or both?
  - Are you asked to argue for or discuss an issue?
  - Are you asked to explain or discuss an issue, or both?
  - How many concepts are you asked to consider?

- **Answer precisely**
  - i.e. brief, focused, comprehensive

Types of Questions

- **“Explain…”**
  - Give an explanation of what something is or how something works in general.
  - Note: A concrete example is not a substitute for a general explanation.
- **“Argue…”**
  - Make up your mind about a certain issue.
  - Present arguments for your opinion on the issue.
- **“Discuss…”**
  - Consider both sides of an issue.
  - Present arguments for both sides.
  - You can state your own opinion, but should present counter-arguments as well.
- **“Suggest…”**
  - Come up with a solution for an issue.
  - Briefly explain what should be done.
- **“Give an example…”**
  - Give a brief example that illustrates the considered concept.
  - When giving examples for several distinct concepts, make sure the examples highlight the distinguishing characteristics.
- **“Point out issues…”**
  - Examine a given artifact, and explain what is wrong with it.
  - Don’t just say *what* is wrong, but *why* it is.
- **“Draw…”**
  - Draw a UML diagram reflecting the relevant aspects of the given scenario.
  - Pay attention to proper syntax!
- **“Implement…”**
  - Write/modify brief segments of Java code.
  - Pay attention to proper syntax!

Sample Questions: Software Process Models

- Explain two of the main causes of trouble that can lead to budget overruns, schedule overruns, or benefits shortfalls in many large software projects.

- Explain three differences between plan-driven and agile software process models.

- Argue whether an agile or a plan-driven approach would have been more suitable for the project you developed in this course.

Sample Questions: Requirements Elicitation

- Explain the difference between functional requirements, quality requirements and general conditions a system must fulfill.

- Imagine the university contracted you to build a new social networking site for all students. Give examples for two types of conflicts that may come up in the requirements, and suggest how they could be resolved.

- According to the INVEST model, user stories should be valuable, estimatable and negotiable (among other characteristics). Explain what these characteristics mean.

Sample Questions: Effort Estimation

- Explain typical problems in software estimation that teams strive to avoid by playing planning poker.
- Imagine your team came up with an effort spread of 8, 13, 20 and 100 for a particular user story while playing planning poker. Argue how you would interpret and deal with this result.
- Discuss whether efforts should rather be over- or underestimated.

Sample Questions: Project Planning

- Explain the difference between person-days, calendar days and work days.
- Imagine your 2-person team is working at a velocity of 0.7. Calculate how many calendar days you should expect for completing a user story estimated at 10 person-days.
- Explain what can be seen in a burn down chart.

Sample Questions: Object-Oriented Analysis

- Explain what object-orientation means in software development.
- Draw a UML class diagram reflecting the following scenario. […]
- Give an example of a class relationship that can only be represented by an `interface` in Java, and argue why it cannot be represented using an abstract class in that language.

Sample Questions: Object-Oriented Design

- To store in-memory object-oriented data persistently in a database, file or other medium, we need some “persistence logic” that transforms the object-oriented data into a relational, text or other structure, and writes it to the storage medium. Discuss the advantages and disadvantages of implementing this persistence logic in the classes representing the data structures, vs. in classes representing the storage medium.

- Argue why it is discouraged to change the signature or behavior of a class’ public methods after its release.

- Explain the difference between class variables and instance variables.

Sample Questions: Testing

- Discuss the benefits of a test-first vs. a code-first approach.

- Imagine we want to implement a class `Money` that represents a certain amount of money in a certain currency. Using the method `boolean isEqual(Money m)`, any instance of this class should be able to check if a given monetary amount `m` equals the amount stored in the instance itself. Following the test-driven development philosophy, implement `@Test` methods of a JUnit test fixture that prescribe how the `compare` method should behave when it receives an equal amount `m` in the same currency, an unequal amount `m` in the same currency, or an amount `m` in a different currency.

- Explain the motivation for boundary value analysis in generating test inputs, and give examples for boundary values that should be tested when working with arrays and with collections.

Wrap-up

Remaining Class Schedule

✓ **Wed**	**30 Mar**	Assignment #4 (Software Test) presentations
✓ **Mon**	**4 Apr**	Final lecture with information on final exam
▪ **Wed**	**6 Apr**	Assignment #5 (Final Product) draft consultations
▪ **Mon**	**11 Apr**	Spare lecture slot – anything you'd like to recap?
▪ **Wed**	**13 Apr**	Easter break – no consultations
▪ **Mon**	**18 Apr**	Easter break – no lecture
▪ **Wed**	**20 Apr**	Assignment #5 (Final Product) presentations
▪ **Sun**	**24 Apr**	Assignment #5 (Final Product) and peer assessment submission
▪ **Tue**	**26 Apr**	Final exam

Special Thanks

- Jaan Jaerving
- Marcelo Felix Audibert
- Tristan Freyr Jónsson
- Valbjörn Jón Valbjörnsson

Kennslukönnun

Evaluate this course on Ugla!
(Open 8–24 April)

Preview: HBV501/601G Software Project 1/2

- After this first taste of programming-in-the-large…
  - One simple approach to agile software development (HBV401G)
- …we'll take a more in-depth look at different software engineering paradigms:
  - Plan-driven development and web engineering (HBV501G)
  - Agile development and mobile software engineering (HBV601G)

- **Course aims:**
  - Learn about the different software engineering methods at your disposal for
    - requirements, estimation, architecture, design, integration, testing, management
  - Gather more practical experience with these approaches
    - in the context of two different software process models (**plan-driven** and **agile**)
    - in the context of two different technology platforms (**web** and **mobile**)
    - in the context of two different system landscapes (**green-field** and **brown-field**)
  - ➢ Make well-informed method decisions and avoid pitfalls in your future industry projects

Keep in Touch!

- For feedback, questions, projects, theses, events, guest talks, collaboration…

book@hi.is

http://is.linkedin.com/in/matthiasbook

http://www.facebook.com/matthiasbook

http://www.twitter.com/matthiasbook

Takk fyrir
og gangi þér vel!

book@hi.is