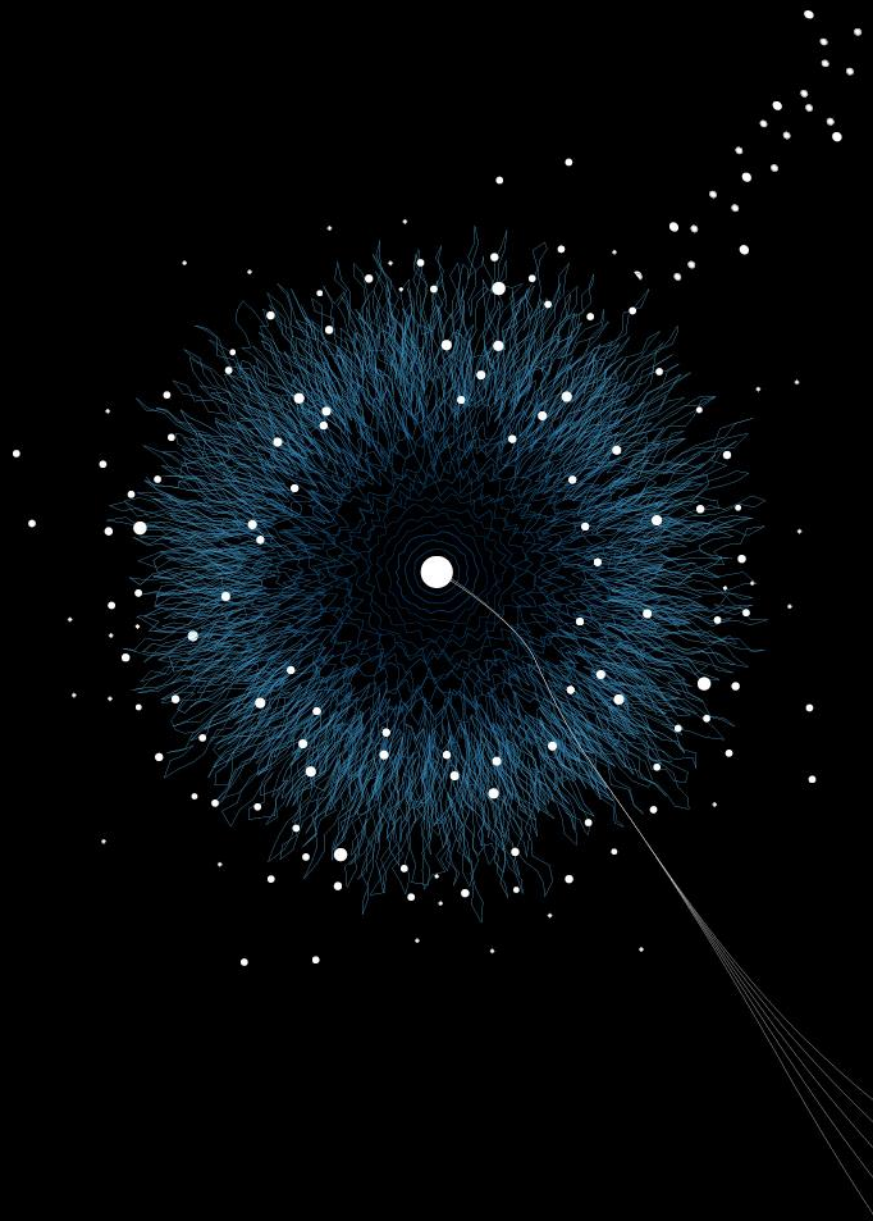
FINAL Q&A

M2 ADVANCED PROGRAMMING

BY TOM VAN DIJK AND FERNANDO CASTOR



UNIVERSITY
OF TWENTE.

TOPICS OF THIS LECTURE

- Overview of topics
- You ask questions
- We give answers

UNIVERSITY
OF TWENTE.

PROGRAMMING OVERVIEW

Week 1 (IP) Procedural programming Arrays Debugging	**Week 2 (IP)** Classes and Objects Documenting and Testing **Exceptions (AP)**	**Week 3 (IP)** Interfaces, Inheritance, Polymorphism Subtyping **Streams and Files (AP)**
Week 4 (AP) **Collections: Set, List, Map** **Specification with JML**	**Week 5 (AP)** **Design Patterns** **User Interfaces**	**Week 6 (AP)** **Basic Concurrency** Project kick-off
Week 7 (AP) Basic Networking **Defensive programming** **Security**	**Week 8/9 (Project)** Project	**Week 10 (Project)** Project Exam

POTENTIAL TOPICS FOR THE EXAM

- Everything **except networking**
- Don’t forget there can be conceptual / theoretical questions, not everything is “can you program bro”
- Java Notes book
- Topic videos & slides from Canvas
- Exercises and accompanying text in the manual, including JML appendix

EXCEPTIONS

- Throwing and catching exceptions
- Declaring thrown exceptions
- Explicit exceptions vs implicit exceptions
- Typical runtime exceptions
- try … catch … finally …
- Making a custom exception
- Handling an exception: ignore it, handle it immediately, pass it on, or pass it on wrapped in another exception

UNIVERSITY
OF TWENTE.

STREAMS AND FILES

- InputStream and OutputStream for **bytes**
- Reader and Writer for **chars**
- Parsing files / streams with Scanner and/or BufferedWriter
- Writing files / streams using a PrintWriter
- Closing streams
- try-with-resources
- Using String.split, converting between String and int / long / float / double, etc.

UNIVERSITY
OF TWENTE.

JML

- Design by contract: method contracts via preconditions and postconditions
- Contracts vs inheritance (looser preconditions, tighter postconditions)
- Logic paradigm vs imperative programs
- Quantifiers: \forall, \exists, \num_of, \min, \max, \sum
- Defining range for quantifiers: collection.contains() etc.

COLLECTIONS

- Collection framework hierarchy
- Using List, Set, Map and their implementations
- Picking the right tool for the job
- Using generics
- Special **for** syntax for iterables

DESIGN PATTERNS

- Three types: structural, behavioural, creational
- Structural: Bridge
- Behavioural: Strategy, State, Observer/Publisher-Subscriber/Listener
- Model-View-Controller

CONCURRENCY I

- Creating threads and waiting for their completion
- Data races (multiple simultaneous accesses with at least one write)
- Race conditions (situations where certain interleavings give wrong results)
- Interleaving, reasoning about interleavings, reasoning about possible outcomes

CONCURRENCY II

- **synchronized** keyword (both in front of a method and as a block)
- Lock, ReentrantLock, safely unlocking a Lock
- Using locks / synchronized correctly
- Using wait and notify / notifyAll
- Using Condition for different wait conditions on the same Lock

SECURITY

- Best ask Dipti!
- CIA Triad: **Confidentiality**, **Integrity**, **Availability**
- Security in the development process, threat modeling (STRIDE), attack trees, functionality vs security vs usability
- Security model of Java
- Authentication & authentication factors
- Encoding, hashing, encryption, cryptographic hash functions, commitments, message authentication codes
- Side-channel attacks

UNIVERSITY
OF TWENTE.

THAT'S ALL, FOLKS

UNIVERSITY
OF TWENTE.