CSC207 - Course Wrap-up

Ilir Dema

Summer 2016

The starting point

```
public class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello world!");
    }
}
```

Learning a new language

- A new memory model.
- Static typing rather than dynamic typing.
- Compiled rather than interpreted.
- Primitives vs. only objects.
- Everything belongs to a class.
- Generics.
- Interfaces and abstract classes.
- A philosophy oriented towards safety.
- A huge API.

You also did more of the learning yourself than before.

More languages

Youll see more languages and the underlying language design principles and approaches:

csc209:	Software Tools and Systems Programming
csc324:	Principles of Programming Languages
csc343:	Introduction to Databases
csc309:	Programming on the Web

Value:

- knowing which tool to apply in a given situation
- deeper appreciation of a languages design and how best to use it

Safety

Much of how Java does things (types, casting, etc.) is intended to make programs safer.

Safety was a key principle behind the design of Java.

Safety came up again when we learned about floating point.

Learn more about floating point and numerical algorithms in csc336: Numerical Methods.

Design

We stepped up several levels from designing functions, methods and classes, to design whole software systems.

CRC cards provided a way to brainstorm and communicate about design at a very high level.

UML provided a way to specify a design, still at a high level and independent of language.

Using a highly practical approach, you have, by now, designed your first e-commerce application!

Design Patterns

Language-independent.

Generic solutions to recurring design problems.

Solutions that follow the object-oriented design principles and best practices.

Youll see more design patterns in csc301: Introduction to Software Engineering.

The algorithmic analog: algorithmic design techniques e.g., divide and conquer, and greedy algorithms. Youll learn these in csc373: Algorithm Design, Analysis, and Complexity.

Regular expressions

These have many practical applications:

- Parsing input in a Java program (or another language).
- Many Unix commands, such as grep, use regular expressions.
- Regular expressions and related concepts are important in language design and compilers. csc488: Compilers and Interpreters

Learn more about the underlying theory in csc236: Introduction to the Theory of Computation, and csc448: Formal Languages and Automata.

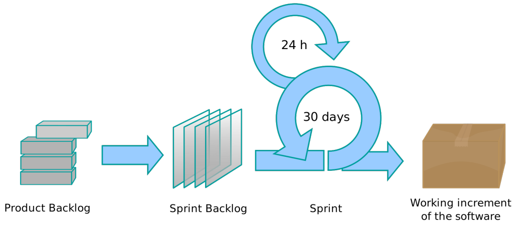
Software Development Process

Working in teams.

Iterative and incremental software development.

Changing requirements.

Elements of state of the art software development practices, such as Scrum and XP.



Software Development Process

You will learn much more through:

- csc301: Introduction to Software Engineering
- csc302: Engineering Large Software Systems
- project courses (csc494/495 or csc490)
- UCOSP: undergraduate capstone open-source project

Learn the formal underpinnings of testing in csc410: Software Testing and Verification.

The Final Exam

- 3 hours
- closed book, but there is an API at the end
- there is also some scrap paper at the end
- Javadoc and internal comments not required
- import statements not required
- helper methods always welcome
- assume all input is valid

Possible types of questions

- Code writing
- Tracing and understanding code where the memory model helps you predict results
- The fundamental concepts of OOP
- Exceptions
- Multiple choice and short answer questions
- Implementing test cases in JUnit
- Design patterns: recognize, explain, write code with the patterns we have covered
- Decode and write UML
- Regular expressions: write and read them
- Agile and scrum

How to best prepare for the exam

- Revisit the assignment and exercises. Learn from what you did previously; improve it.
- Write code that tries to poke holes in your understanding of Java concepts (name lookup, exceptions, etc.).
- Write regular expressions and test them in Java.
- Write code that uses the design patterns we studied.
- Write some UML.
- Work out exercises posted on the course web site.
- Solve old tests and exams.

KEEP
CALM
AND
GOOD LUCK
IN YOUR EXAMS