COMP 413 Fall, 2016, Course Overview / Preliminary Schedule

Some Key Dates
13 Sep (week 3): project 1 due
20 Sep (week 4): quiz 1
23 Sep (week 4): project 2 due
27 Sep (week 5): test 1
11 Oct (week 7): no class (Mid-Semester Break)
21 Oct (week 8): project 3 due
25 Oct (week 9): quiz 2
01 Nov (week 10): test 2
04 Nov (Friday): withdrawal deadline
22 Nov (week 13): project 4 due
29 Nov (week 14): test 3
06 Dec (week 15): last class + quiz 3
13 Dec: test 4 (final) + project 5/6 presentations and final project due date

Week 1: 30 August

Session

- organizational matters
    - introduction: instructor, TA
    - course objectives
    - course texts
    - course roadmap (see the last page of this document)
    - Piazza discussion forum (mandatory subscription – link through Sakai)
    - how to get help
    - prerequisites and review topics: 271 313 overview
- motivation, including embedded systems
    - brief overview of batch versus event-based programming
- requirements
    - functional: y = f(x)
    - nonfunctional: additional properties of f, e.g.
        - testability
            - most important nonfunctional requirement
            - allows testing whether functional requirements are met
            - good architecture often happens as a side-effect (APPP pp. 36-38)
        - performance
        - scalability
            - e.g. performance for large data sets: asymptotic order of complexity (big-Oh) in terms of input size n
        - reliability
        - maintainability
        - static versus dynamic NFRs
- discussion of projects 1 and 2
- course software with demo
- prerequisite assessment

Reading/Podcasts

- Object-Oriented Programming Using Java (OOPUJ) chapters 1, 2.1, 2.2, 3
    - OO Introduction; UML Class Diagrams & Syntax; Inheritance & Method Overriding
- SE (Software Engineering) Radio episode 1 on patterns
    - Full list of all SE Radio episodes: http://www.se-radio.net/feed

Week 2: 6 September

Session

- announcements
- data structures
  - linear vs. nonlinear
  - position-based vs. policy-based (see also here)
  - performance
  - tying data structure choices to requirements
- data abstraction
  - addressing: pointers, references
  - aggregation (product types): structs, records
    - example: node in a linked list
  - variation (sum types): tagged unions, multiple implementations of an interface
  - example: mutable set abstraction
    - add element
    - remove element
    - check whether an element is present
    - check if empty
    - how many elements
  - several possible implementations
    - reasonable: binary search tree, hash table, bit vector (for small underlying domains)
    - less reasonable: array, linked list
    - see also here
- group activity: problem 4 on prerequisite assessment

Reading/Podcasts

- OOPUJ chapters 4, 5
  - Object Roles and Polymorphism; Method Overloading

Week 3: 13 September

Session

- announcements
  - project 1/software installation check-up
  - Quiz 1 next week
- discussion of project 2
- basics of object-oriented programming up to genericity (Generics)
  - Inheritance and Composition
  - Interfaces
  - Abstract Classes
- More on Test-Driven Development
  - JUnit Annotations, test methods, and examples

Reading/Podcasts

- OOPUJ chapter 6
  - OO Software Analysis and Design
- SE Radio episode 2 on dependencies

Homework

- **Project 1 due Tuesday, September 13**

Week 4: 20 September

Session

- ***Quiz 1: short quiz on first 2 SE Radio episodes, patterns and dependencies***
- announcements
  - test 1 roadmap on Sakai
- basics of object-oriented programming - through the rest of the online document
  - Generic types
  - Supplemental material: Java Collections, data structures, and Object-inherited methods (see course slides)
  - Optional topic - Coad: modeling with UML and color: overview, book chapter
- reverse engineering JUnit tests (test 1 topic)
- introduction to Design Patterns
  - Overview
  - Factory method, Strategy, Visitor
  - Resources
    - tutorialspoint website
    - Bob Tarr pdf slides on Sakai
    - APPP Design Pattern references
      - Factory: Chapter 29
      - Strategy: Chapter 22
      - Visitor: Chapter 35
- project 3 introduction (only if time)

Reading/Podcasts

- Bob Tarr pdf slide sets on Factory, Strategy, and Visitor Design Patterns
- Agile Principles, Patterns, and Practices in C# (APPP) chapters 1-3 and Design Patterns chapters as above
  - Agile Practices; Extreme Programming Overview; Planning
  - Factory, Strategy, and Visitor Design Patterns

Homework

- **Project 2 due Friday, September 23**

Week 5: 27 September

Session

- ***test 1***
- announcements
    - project 3 team formation
- more Design Patterns
    - Decorator and Composite
    - Visitor revisited
    - Resources
        - tutorialspoint website
        - Bob Tarr pdf slides on Sakai
        - APPP Design Pattern references
            - Decorator: part of Chapter 35
            - Composite: Chapter 31
            - Visitor: Chapter 35
- project 3
    - Shapes interface and Visitor<Result> generic interface
        - concrete Shapes: Circle, Rectangle, …
        - concrete Visitors: Draw, Size, and Bounding Box
    - project 3 TODOs (Android Studio: Tools => View => TODO)
    - project 3 Decorators: Outline, Stroke, Location, …
    - Android Canvas and Paint classes and online documentation
    - Unit tests using Gradle and Mockito; the Fixtures class

Reading/Podcasts

- Bob Tarr pdf slide sets on Decorator, Composite, and Visitor Design Patterns
- APPP chapters 4-6 and Design Patterns chapters as above
    - Testing; Refactoring; A Programming Episode
    - Decorator, Composite, and Visitor Design Patterns
- SE Radio episode 167 on unit testing

Week 6: 4 October

Session

- announcements
    - team members posted on Sakai and Piazza
    - team repositories: *cs413f15teamNp3*
- discussion of test 1
- continued project 3 detailed discussion
    - more classes, including Fixtures
    - Mokito "white box" unit tests (esp. for the Bounding Box Visitor)
        - how to run the unit tests
    - expressions and vexpressions Java examples - using a Visitor<Result> interface to visit arithmetic expressions
    - coding guidelines
- UML diagrams and 30-minute in-class group activity
    - create a UML class diagram for project 3 (hand-drawn is best)
    - Submit (a picture of) the diagram plus a brief write up about how you did it on Sakai - one per group
- Agile development (if time)
    - agile development principles
    - MVP (Minimal Viable Product – low risk) versus BUFD (Big Up-Front Design – high risk)
        - indirection: performance versus flexibility

Reading/Podcasts

- APPP chapters 13 & 14
    - Overview of UML for C# Programmers; Working with Diagrams
- Mokito overview
- Android 4 App Development Essentials, Chapters 1-4, available here: **http://www.techotopia.com/index.php/Android_4_App_Development_Essentials**
    - **Note: even though this reading describes installing and using Eclipse, we will use only Android Studio in this course**

Week 7: 11 October (Mid-Semester Break)

Week 8: 18 October

Session

- announcements
  - any remaining questions about project 3
  - reminders: Quiz 2 next week (2 SE Radio podcasts), test 2 the following week
- principles of object-oriented programming: SOLID
  - S - Single Responsibility Principle
  - O - Open Closed Principle
  - L - Liskov Substitution Principle
  - I - Interface Segregation Principle
  - D - Dependency Inversion Principle
  - References
    - Uncle Bob's Principles of OOD
    - Pablo's SOLID Software e-book
    - SOLID Principles in C#
- SOLID and other basic object-oriented design principles ("SOLID + 2"): presentation
  - overview
  - extended overview by Uncle Bob with links to detailed articles
  - information hiding/minimize coupling/Law of Demeter (Tarr p1-)
  - favor composition over inheritance (Tarr p9-)
    - Coad's rules (Tarr p22-)
    - role-based design (Tar p23-)
  - dependency inversion principle/design with interfaces (Tarr p33-, Coad)
  - open-closed principle (Tarr p40-)
  - Liskov substitution principle (Tarr p51-)
  - single-responsibility principle (cohesion part 1)
  - interface segregation principle (cohesion part 2)
  - package-level principles: cohesion and coupling
    - acyclic dependencies
- Android example programs
  - Android framework
    - architecture
    - overview
    - activities and their lifecycle (scroll about 60% down)
    - tutorials
  - examples - search for "android-java"
  - hello-android-java - notification
    - HAXM (see recent post)
    - creation of AVD
    - roles of hg and Gradle
  - simplebatch-android-java - scrollable text output
    - functionality: scrollable
    - Android framework and activity life cycle
    - preview of agile process
  - simpledraw-android-java - drawing simple shapes based on lines

Reading/Podcasts

- Android 4 App Development Essentials, Chapters 5-8
- SE Radio episode 46 on refactoring **- will be on Quiz 2**

Homework

- **Project 3 due Friday, October 21**

Week 9: 25 October

Session

- ***Quiz 2: short quiz on second 2 SE Radio episodes, unit testing (167) and refactoring (46)***
- announcements
  - test 2 next week
- more design patterns
  - Adapter
  - Facade
  - Observer
  - State - including a review
  - Command
- modeling and introduction to Project 4
  - Model-View-Adapter (MVA) architectural design pattern
  - modeling dynamic, event-driven behavior with state diagrams
  - model states versus view states
  - state diagram examples
  - stopwatch model (hardware perspective)
  - our stopwatch model
  - Project 4 introduction and overview
  - stopwatch-android-java overview
- Android framework (if needed)
  - architecture
  - overview
  - activities and their lifecycle (scroll about 60% down)
  - tutorials

Reading/Podcasts

- APPP chapters 33, 23, 32, 15, 21
  - Adapter - 33
  - Facade - 23
  - Observer - 32
  - State - 15
  - Command - 21

Week 10: 1 Novemeber

Session

- ***test 2***
- Android
  - details of the activity lifecycle (scroll down about 60%)
  - how to rotate the emulator (in Genymotion, just click the rotate icon!)
  - saving the activity state
  - clickcounter - event-based interaction
- UML Extended State Machines (with guards)
  - the implicit clickcounter state machine
- in-class group exercise: create a dynamic UML extended state machine model for Project 4
  - capture these to submit as part of each 2-person team's Project 4 submission

Reading/Podcasts

- APPP chapters 21, 23, 32, 36
  - Command and Active Object: Versatility
  - Façade and Mediator
  - Observer: Evolving into a Pattern
  - State
- SE Radio episode 65 on embedded systems

Week 11: 8 November

Session

- announcements
    - reminder: you should have listened to SE Radio episode 65 last week on embedded systems!
- test 2 discussion - as needed
- detailed discussion of testing in clickcounter and stopwatch examples
    - see the Android new build system user guide for info about build.gradle and Android testing
- in-class group exercise: create a comprehensive set of unit tests for Project 4
    - also capture these to submit as part of each 2-person team's Project 4 submission
- more Android examples - only if time
    - simplebatch - scrollable text output
    - simpledraw
- possibly time to work on Project 4 in your teams

Reading/Podcasts

- same as week 10
- relevant architectural/design patterns
    - State pattern (APPP chapter 36)
    - event listener/callback
        - one versus multiple listeners
        - Observer pattern (APPP chapter 32)
    - UI architectural patterns
        - Model-View-Adapter (MVA)
        - Model-View-Controller (MVC)
        - Model-View-Presenter (MVP) (see also APPP chapter 38)
        - comparison between MVA and MVC

Week 12: 15 November

- brief review of extended state diagrams
  - ClickCounter and Stopwatch
- Project 4: saving and restoring Activity state
- event-driven programming - Test 3 roadmap items
  - threads, runnables, the run and start methods, ...
  - MVP and MVVM
- Model-View-Adapter in ClickCounter and Stopwatch
- possibly time to work on Project 4 in your teams

Reading/Podcasts

- APPP chapters 7-9, 18, 19
  - What is Agile Design; The Single-Responsibility Principle; The Open/Closed Principle
  - Sequence Diagrams
  - Class Diagrams
- SE Radio episode 12 on concurrency

COMP 413 Fall, 2016, Course Overview / Preliminary Schedule

Week 13: 22 November

- announcements
    - Project 4 and state machine/testing extra credit assignments due tonight
    - test 3 next week - practice test is in Week 13 on Sakai
- project 5 introduction
- agile design: process, not event: presentation
    - design smells: usually subjective, sometimes objective
        - rigidity: difficult to change
        - fragility: easy to break
        - immobility: difficult to reuse
        - viscosity (of software, of environment): it is difficult to do the right thing
        - accidental complexity: e.g., overdesign
        - needless repetition (DRY)
        - opacity
    - overview of SOLID design principles
    - design perfume
- refactoring
    - code smells and refactoring
    - introduction: code smells (also here, summarized here) and refactoring
    - real-world significance
        - most projects legacy/evolve over time
        - new members join existing teams
        - smells arise
        - economy of scope (requirements) versus economy of scale (standardization) (see also this presentation)
    - IntelliJ IDEA supports some code refactoring
- immutability

  Other topics

    - agile development principles and process/practices
        - general overview of software testing
        - test-driven development
        - continuous integration/delivery

Homework

- **Project 4 due Tuesday night, November 22, by 11:55pm**

Reading/Podcasts

- PA chapters 6 and 8
    - Building a View
    - Drawing 2D and 3D Graphics
- APPP chapter 10-12
    - The Liskov Substitution Principle (LSP); The Dependency-Inversion Principle (DIP); The Interface Segregation Principle (ISP)

Week 14: 29 November

- ***test 3***
- announcements
  - course IDEA survey opens November 28, closes December 6
  - Quiz 3 next week
- concurrency
  - interleaving
    - Scala example
    - calculating the number of possible interleavings
  - nondeterminism
  - race conditions
    - Scala example: increment of shared variable
  - key difference between these two examples?
  - Java threads
  - Java examples
  - overview
  - physical versus logical concurrency
  - CPU-bound versus I/O-bound activities
    - CPU-bound example
    - I/O-bound example
  - run-to-completion versus coordination
  - (conflicting) design forces:
    - safety
    - liveness
    - performance
      - throughput
      - latency
      - jitter
- example: prime number checker
  - direct execution
  - asynchronous (background) execution
  - cloud-based execution
- Cloud Computing - XaaS (X as a Service)
  - Cloud services and benefits

Reading/Podcasts

- same as week 13
- SE Radio episode 23 on software architecture

Week 15: 6 December

- ***Quiz 3: short quiz on next 3 SE Radio episodes, embedded systems (65), concurrency (12), and software architecture (23)***
- announcements
  - **IDEA survey reminder**
  - test 4 (final exam) next week
  - Project 5/6 presentation and submission next week
- test 3 discussion, if needed
- final topics from Week 14, if needed
- possible Java 8 overview
- possible in-class time to work on Project 5/6

Reading/Podcasts

- SE Radio episode 110 on roles in software engineering

Final Session (Week 16): 13 December

- announcements, if any
- ***test 4 (final)***
- **in-class review of Project 5/6 implementations**

Reading/Podcasts

- SE Radio episode 150 on software craftspersonship (OK to wait until break)

Homework

- **Project 5/6 due Tuesday, December 13 at 5pm**

COMP 413 Fall, 2016, Course Overview / Preliminary Schedule

Dr. Läufer's Course Outline

Overall Outline of Topics (subject to revision)

- organization, motivation, introduction (1 week: 1 total)
  - what makes software good?
  - requirements: functional vs. nonfunctional
  - the importance of testing
- basics of object-oriented programming (2 weeks: 3 total)
  - semantics: reference vs. value, equality vs. identity
  - types and classes: relationships, polymorphism
  - code organization: member access, packages/namespaces
- agile development process (1 week: 4 total)
  - overview
  - testing
  - refactoring
  - continuous integration and delivery
- object-oriented design principles (2 weeks: 6 total)
  - overview
  - SOLID
  - designing with interfaces
- agile object-oriented modeling (2 weeks: 8 total)
  - main UML diagrams: class, state machine, sequence
  - archetypes and colors
- software design patterns (2 weeks: 10 total)
  - key patterns from APPP and HFDP
- concurrent programming (3 weeks: 13 total)
  - events
  - threads
  - sharing
- distributed programming (1 week: 14 total)
  - overview and principles
  - connecting to web services

Typical structure of a weekly session

- OOPUJ or APPP or PA topics
- project discussion and related topics
- pair/group presentation or other activity

Typical assignments over a two-to-three-week period

- reading
- listening to SE (Software Engineering) Radio episodes
- programming project