CMPT 276 Midterm Exam Review

This summary is meant to highlight what type of material is and is not testable. Many questions may rely on you understanding and applying this knowledge; it is not sufficient to memorize this list, you must be able to use the material. Some items may have been specifically mentioned in class as being testable and may not be mentioned here.

- Most topics listed refer to the slide headings.
- “Know” means have memorized.
  - ⋆ “Know the days of the week” means have memorized Monday, Tuesday, . . . ; and “understand” each of them.
  - ⋆ You do not need to memorize explanations word-for-word; you should know in your own words.
- “Understand” means: once told the topic (or items), be able to talk about it/them.
  - ⋆ “Understand all flavors of ice-cream” means given a flavor (say chocolate mint chip), be able to describe it; compare (similarities) and contrast (differences) it to other flavors. But, don’t memorize all the flavor names (not asked “List all 31 flavors”).
- The test will focus on material from lecture, but there will be some on Android development and tools.
  - ⋆ This guide summarizes content from lecture and the Android assignments.

1. Lecture Content

0) **Admin**
   - Review expectations and consequences for academic honesty.
   - Nothing testable.
1) **Intro to Software Engineering (SE)**
   - Know what is software engineering, and the importance of software.
   - Know the four fundamental software process activities.
   - Understand each of the four essential attributes of good software.
   - Compare and contrast generic software and custom software.
   - Understand software engineering diversity
   - Understand each type of application.
2) **Ethics and Case Studies**
   - Understand the issues of professional responsibility
   - Understand code of ethics and each item in the ACM/IEEE code of ethics
   - Nothing on specific case studies, but you should understand the different types of applications which they are, and be able to discus general differences between those types.
3) **Software Processes**
   - Know what a software process is.
   - Know the four software process activities (same as “Intro to SE slides).
   - Understand each of the steps in the “Software Specification” overview.
   - Know the basic process to get from System Specification to an Executable System.
   - Understand the different design activities.

- Know three testing stages.
- Know the two software development (planning) paradigms
- Know the two software delivery (timing) options.
    - ⋆ Able to explain how each software development paradigm relates to each software delivery option.
- Know the waterfall model and its phases (don't need exact names).
    - ⋆ Understand waterfall model's ability to cope with change.
- Know incremental development, and how it can be used by either paradigm.
    - ⋆ Know 2 benefits of incremental development over waterfall.
    - ⋆ Know 2 drawbacks to incremental development.
- Understand refactoring.
- Understand reuse-oriented SE.

4) **Version Control**

- Purpose and need for version control.
- Know what each of the following means:
    - ⋆ repository, clone, import/add, commit, pull, push revert, history, diff, merge, merge conflict, file lock.
- Know lock-edit-unlock and/vs checkout-edit-merge.
- Understand atomic operations, branch/fork, tag/label/baseline.
- Know the first rule of being a good team member
- Know the three points about how coding with source control is different, and why.

5) **Change Risk**

- Know why change is inevitable, and the cost of change.
- Know change avoidance vs change tolerance, and one technique suited to each.
- Prototyping:
    - ⋆ Know when and how it is used.
    - ⋆ Know what can be ignored.
    - ⋆ Know why it is a throw-away.
    - ⋆ Know how it avoids change.
    - ⋆ Understand the ways it improves a system
- Incremental delivery:
    - ⋆ Know how it tolerates change.
    - ⋆ Know use and benefits of incremental development, and incremental delivery.
    - ⋆ Know Incremental delivery cycle.
    - ⋆ Know how it reduces risk of project failure.
    - ⋆ Understand problems with incremental delivery.
- Rational Unified Process
    - ⋆ Able to explain its use of iteration, phases, and workflows.
    - ⋆ Understand and able to explain each RUP good practice.

6) **Agile**

- Know the "inspiration" for agile methods
- Know the four "values" of the Agile Manifesto. Able to explain what each one is a choice between, and why agile methods make the choice it does.
- Know the five principles of agile methods.
- Able to explain applicability of agile methods and their limitations.
- Understand problems with agile methods.

- Able to discuss how agile methods apply to software maintenance.
- Able to discuss the choice between plan-driven and agile development and the factors involved.

7) **Extreme Programming**
   - Know what XP is.
   - Understand each of the 10 XP practices.
   - Understand XP use of user stories (scenarios).
   - Understand XP tasks, refactoring, and not designing for change.
     - ⋆ Know four refactorings.
   - Know test first development, automated test harness, and their advantages.
   - Understand pair programming and it's benefits.
   - Know XP testing: test first development, customer involvement, and test automation.

8) **Agile Management**
   - Understand how management will differ between agile and plan-driven processes.
   - Scrum:
     - ⋆ Know the 4 activities in a sprint cycle
     - ⋆ Understand teamwork in scrum.
     - ⋆ Know what information is shared in a Scrum meeting.
     - ⋆ Know scrum master's role
   - Understand challenge of scaling up Agile to larger systems.

9) **Requirements Engineering (RE)**
   - Know what RE is.
   - Know user requirements and system requirements
   - Know functional vs non-functional requirements
     - ⋆ Which applies to the whole system vs specific portion
     - ⋆ Ambiguity, completeness, consistency.
     - ⋆ Know how non-functional requirements may lead to functional requirements.
     - ⋆ Able to apply metrics for quantifying non-functional requirements.
   - Understand domain requirements problems.

10) **Requirements Document**
    - Know what a requirements document is.
      - ⋆ Understand and discuss its uses.
    - Know what is requirements specification
      - ⋆ Understand ways of writing requirements.
    - Able to discuss advantages and disadvantages of stating requirements in natural language.
    - Understand guidelines for writing requirements
    - Able to discuss limitations of using natural language.
    - Understand structured and tabular format for expressing requirements.

2. Android and Tools

- Revision control ideas covered above in lecture section.
- Know different Android tools. Know what abbreviations stand for, and able to describe function of each:
  - ⋆ Android Studio IDE, git, GitLab, Java (JDK)

    - ⋆ Android SDK, emulator, LogCat
        - — Do not need to know any specific details such as how to create a repository using git.

- Know the purpose of each file in `src/` and `res/` directories of the project created in the assignments, such as:
    - ⋆ layouts, activity classes (`.java`), android manifest, `string.xml`,
    - ⋆ Where to put images?
- Understand the `build/` folder
    - ⋆ What is `R.java`
- Know the code how to:
    - ⋆ Given the name of a string defined in `string.xml` (such as `menu_help`), access the string via Java code.
    - ⋆ Given the name (id) and class type of a layout element (such as a `TextView` with id `TextView_MenuTitle`), write the Java code to a create a variable reference to the object (using `findViewById()`)
    - ⋆ Given the resource id (such as `R.id.btn_test`) of a button, register an `OnClickListener` that prints a `Log` message when the button is clicked.
    - ⋆ Display a `Toast` message
- Know what each of the following are: activities, intents.
    - ⋆ Understand the process of creating a second activity.
    - ⋆ Able to write the call to switch to a new activity (using `startActivity()`)
    - ⋆ Understand the stack of activities, how the back button works, and how `finish()` works.
- Understand application life cycle (diagram in chapter 2) with respect to when the following methods are called: `onCreate()`, `onPause()`, and `onResume()`