CS/SE 2XB3 Computer Science Practice and Experience: Binding Theory to Practice

Reza Samavi

Jan. 1, 2019

1 Learning Objectives: Postcondition

A learning objective for a course is something the student is expected to know and understand or to be able to do by the end of the course. The learning objectives for this course are given below. Taken together, this set of learning objectives constitute the postconditions of the course.

1. Students should know and understand
   - (a) the basic Java programming model for implementing algorithms.
   - (b) definition of abstract data types (ADTs) in Java.
   - (c) using ADTs in the service of modular programming.
   - (d) the concept of design by contract.
   - (e) basics of scientific methods for experimental studies of algorithms.
   - (f) the concept of software revisions and version control.
   - (g) application of basic sorting, searching, and graph algorithms in commercial data processing and in modern scientific computing.
   - (h) application of software engineering principles on designing Java programs with algorithmic contents.
   - (i) software engineering methodologies and life cycle of completing a small software project with algorithmic contents.
2. Students should be able to
   - (a) use integrated development environments (IDEs) to develop Java programs.
   - (b) implement in Java fundamental and broadly useful data structures: bags, queues, stacks, search trees, heaps, hash tables.

(c) implement in Java basic user defined ADTs, basic sorting, searching and graph algorithms.

(d) implement in Java basic algorithm cost models and run experiments in Java to analyze running times of different algorithms.

(e) construct client programs in Java that satisfy a set of given specifications for an application.

(f) formulate and implement test plans using JUnit.

(g) use SVN and version control to coordinate work of multiple programmer in a team.

(h) explore and formulate an engineering problem that requires algorithmic solutions and communicate the problem with peers (in a form of a project pitch) to recruit members to form a team.

(i) form a software development team and play the role of a software engineer to implement small software projects with algorithmic contents.

(j) report the outcome of the team work in the form of a presentation and a written report.

(k) evaluate a software development team dynamics.

2 Mapping to Attributes with their Indicators

A01 Knowledge

(3) Competence in Engineering Fundamentals	2c, 2b, 2d–2i
(4) Competence in Specialized Engineering Knowledge	1b–1i

A02 Problem Analysis

(1) Ability to identify the essential characteristics of a technical problem, including scope	2e– 2i, 1g–1i
(4) Ability to decompose and organize a problem into manageable sub-problems	1c, 1i, 1h, 2i
(5) Ability to obtain substantiated conclusions as a result of a problem solution including recognizing the limitations of the solutions	1e, 2d
(6) The ability to use modern/state of the art tools	2a–2c, 2f, 2g

A03 Investigation

(1) Capable of selecting appropriate model and methods and identify assumptions and constraints	1a, 1e, 2e
(5) Assess the accuracy and precision of results and recognize limitations of the approach	2d, 2f
(6) Properly documents and communicates processes and outcomes	2i, 2f, 2b–2e

A04 Design

(1) Recognizes and follows an engineering design process	2h
(2) Recognizes and follows engineering design principles	2i
(7) Properly documents and communicates processes and outcomes	2i
(9) Able to work in a group, taking a leadership role as appropriate and relinquishing the leadership role as appropriate	2i

A05 Use of Engineering Tools

(2) The ability to use of modern/state of the art tools	2a–2c,, 2f, 2g

A06 Individual and Team Work

(4) Able to work in a group, taking a leadership role as appropriate and relinquishing the leadership role as appropriate	2i
(6) Understands the need for accountability internally and externally	2k

A07 Communication skills

(2) Presents instructions and information clearly and concisely	2h, 2j
(3) Constructs effective written arguments	2h, 2j
(5) Cite appropriately the works of others	2h
(6) Uses appropriate visual aids and presentation techniques to engage the audience	2h, 2j

A11 Economics and project management

(2) Can plan and effectively manage time, resources, and scope	2h, 2i

3 Course work

The course work consists of:

(1) Assignments	30%
(2) In-Lab Quizzes and Lab participation	30%
(3) Final Project Proposal (Team work)	40%
Total	**100%**

4 Prerequisite and co-requisite learning objectives

(All references are to the 2013-2014 versions of the course reports)

- Prerequisite
  - CS/SE 2XA3: 1a, 1b, 2a, 2b, 2c
  - CS/SE 2S03:1a-1k, 1m, 2a-2r, 2t-2x
- Co-requisite
  - CS/SE 2C03: 1b-1e, 2a-2e
  - CS 2ME3 / SE 2AA4: 1a, 1d, 1e, 1g, 2a-2c, 2g-2h

5 Learning outcomes and indicators

Topic	Below	Marginal	Meets	Exceeds
1a, 2a, 2e	does not understand a simple Java program model to implement basic algorithms	understands simple Java program model (APIs, Implementations, and client code), can fix simple errors, and import/export Java projects to/from an IDE	can design, create, and execute simple Java program model (APIs, Implementations, and client code) with Input/Output files, can fix simple errors, and import/export Java projects to/from an IDE	can design, create, and execute simple Java program model (APIs, Implementations, and client code) with Input/Output files. The student's program is well commented and encapsulation and modularization are well preserved

Topic	Below	Marginal	Meets	Exceeds
1b, 2b, 2c, 1c	cannot implement the fundamental data structures: bag, queue, stack	can implement fundamental data structures: bag, queue, stack	can implement fundamental data structures: bag, queue, stack and simple user defined ADTs	can implement and add new methods to fundamental data types and implement more complex user defined ADTs given a set of specifications
1e, 2d	does not understand the scientific method of running experiments and cannot measure simple algorithm running time	understands the scientific method but has difficulty in implementing simple experiments for measuring different algorithms running times	understands the scientific method and correctly implement simple experiments for measuring different algorithms running times	understands the scientific method and correctly implement simple experiments for measuring different algorithms running times, and can reason about and predict a program behaviour
1d, 2f	does not understand the concept of design by contract	understands the concept of design by contract, can handle basic exceptions but cannot create sufficient test cases using JUnit	understands the concept of design by contract, can handle Java system exceptions and create simple test cases using JUnit	understands the concept of design by contract, can handle Java system and user defined exceptions and create all necessary test cases using JUnit to verify the correctness of an implemenation

Topic	Below	Marginal	Meets	Exceeds
1g, 2h	does not understand basics of requirements engineering and software specifications to define a problem that requires algorithmic solutions	understands the basics of requirements engineering and software specifications to define a problem that requires algorithmic solutions but cannot sufficiently formulate the problem	understands the basics of requirements engineering and software specifications to define a problem that requires algorithmic solutions and can correctly and sufficiently formulate the problem and communicate the problem with the peers	understands the basics of requirements engineering and software specifications to define a problem that requires algorithmic solutions and can correctly and sufficiently formulate the problem, well communicate the problem with the peers and propose a feasible plan to implement the solution
1h,1i, 2i, 2j, 1f, 2g	does not understand the software engineering principles and methodologies to complete a team project and cannot effectively communicate the design processes with the peers	relative understanding of software engineering principles and methodologies to complete a team project with algorithmic content and marginally communicating the team dynamics and the design, verification, and validation processes with the peers	correctly apply software engineering principles and methodologies to complete a team project with algorithmic content and communicate the team dynamics and the design, verification, and validation processes with peers	correctly apply software engineering principles and methodologies to complete a team project with algorithmic contents, communicate the team dynamics and the design, verification, and validation processes with the peers, and formulate a set of extensions for the project and lessons learned from technical perspective and team dynamics

Topic	Below	Marginal	Meets	Exceeds
2k	submission of the review only with some scores	submission of the review with all scores but incomplete justifications for each score	submission of the review with all scores and complete justifications for each score	submission of the review with all scores and complete justifications for each score accompanied by a summary of recommendations to improve team dynamics