L. Mayfield
FA 2023

COMP325 Course Competencies

Knowledge Areas

1. FPL/Functional Programming
    a. Lambda Expressions and Evaluation
    b. Effect-Free Programming
    c. Processing structured data via functions with cases for each data variant
    d. Higher-Order functions
2. FPL/Object-Oriented Languages
    a. Definition of Objects and Classes
    b. Subclasses, inheritance, and method overriding
    c. Traits and Mixins
    d. Dynamic Dispatch
    e. Dynamic vs Static Properties
    f. Composition vs inheritance
3. FPL/Language Translation and Execution
    a. Interpretation vs Compilation
    b. Language Translation Pipeline
4. FPL/Program Representation
    a. Components of a language: Syntax v. Semantics
    b. Basic Programming abstractions: constants, variables, declarations, command, expression, assignment, selection, iteration, functions
    c. Mutable vs. Immutable Variables
    d. L-Values and R-Values
    e. Scope Rules
    f. Environment vs Store
    g. Data Structures to represent code for execution and translation
5. FPL/Type Systems
    a. Types: primitive and compound types
    b. Type safety
    c. Static vs dynamic typing
    d. Type checking
6. FPL/Event-Driven Reactive Programming
    a. Procedural vs Reactive Programming
    b. Components of Reactive Programs: event-source, event signals, listeners and dispatchers, event-handlers.
    c. Behavior model of event-based programming
7. FPL/Advanced Programming Constructs
    a. Lazy Evaluation
    b. OO Mixins and Traits
    c. Metaprogramming with Macros
    d. Dynamic Code Evaluation (eval)

Course competencies derived from and otherwise based on ACM curricular guidelines. By and large, these course competencies borrow directly from the March 2023 Version Beta draft of the *Computer Science Curricula 2023*. For more information see https://csed.acm.org/cs2023-beta/

Professional Dispositions

- **Meticulousness**: Attention to detail is essential when using and designing programming languages.
- **Self-Directed:** Become self-motivated to acquire complementary knowledge from language documentation.
- **Inventive:** Programming is inherently a creative process and programmers are called on to show some innovation in the way they approach solving a problem.
- **Perseverance:** A process of refinement may be necessary to reach a solution given that the correct approach is not always self-evident.

Skill's Hierarchy

1. **Explain** - define, describe, discuss, enumerate, express, identify, indicate, list, name, select, state, summarize, tabulate, translate
2. **Apply -** backup, calculate, compute, configure, debug, experiment, install, iterate, interpret, manipulate, map, measure, predict, randomize, restore, schedule, solve, test, trace
3. **Evaluate** - analyze, compare, classify, contrast, distinguish, categorize, differentiate, discriminate, order, prioritize, criticize, support, assess, choose, defend, rank
4. **Develop** - combine, compile, compose, create, design, generalize, integrate, modify, organize, produce, rewrite, refactor, write

Competencies

1. **Task: Make an informed decision regarding which programming language/paradigm to select and use for a specific application.**
    a. *Competency:* Apply knowledge of multiple programming paradigms and select an appropriate paradigm and language.
    b. *Required Knowledge:* FPL/Functional Programming, FPL/Object-Oriented Programming, FPL/Type Systems, FPL/Event-Driven and Reactive Programming, FPL/Advanced Programming Constructs
    c. *Required Skill: Explain/*Evaluate
2. **Task: Translate input from a high-level language into a lower-level form suitable for use by a computer.**
    a. *Competency:* Analyze input, determine its correctness and meaning, and translate into an alternative form appropriate for the application, possibly employing an intermediate form.
    b. *Required Knowledge:* FPL/Language Translation and Execution, FPL/Program Representation
    c. *Required Skill: Apply/Evaluate/Develop*
3. ***Task:*** **Write a report comparing and contrasting features of two or more programming languages.**
    a. *Competency:* Analyze and explore a language and its documentation in order to identify syntactic and semantic choices made in the design and implementation of the language and their impact on the pragmatic use of the language.
    b. *Required: Knowledge:* FPL/Object-Oriented Programming, FPL/Functional Programming, FPL/Event-Driven and Reactive Programming, FPL/Advanced Programming Constructs, FPL/Type Systems, FPL/Language Translation and Execution
    c. *Required Skill*: Apply/Evaluate
4. **Task: Write a paper explaining how a safe and secure program effectively utilized programming language features to make it safe and secure:**
    a. *Competency:* Apply knowledge of programming paradigms, type systems, static and dynamic semantics, and the compilation/interpretation process to explain how a program executes as it should and does so in a safe and efficient manner.
    b. *Required Knowledge*: FPL/Object-Oriented Programming, FPL/Functional Programming, FPL/Event-Driven and Reactive Programming, FPL/Advanced Programming Constructs, FPL/Type Systems, FPL/Language Translation and Execution, FPL/Program Representation
    c. *Required Skill:* Apply/Evaluate

Course competencies derived from and otherwise based on ACM curricular guidelines. By and large, these course competencies borrow directly from the March 2023 Version Beta draft of the *Computer Science Curricula 2023*. For more information see https://csed.acm.org/cs2023-beta/