McCormick
Northwestern Engineering
Department of Electrical Engineering and Computer Science

EECS 395/495 – Object-Oriented Languages and Environments

Winter 2013 Syllabus

Instructor: Goce Trajcevski **(**goce@eecs.northwestern.edu)

Class meets: Mon./Wed. 4:00-5:20PM at L150 Tech.

Course description: The main goal of this course is to expose the students to the fundamentals of the Object-Oriented programming paradigm, focusing on the balance between breadth vs. depth when studying the syntax, semantics and run-time behavior of different representative languages. Towards that, the course will have the following main parts:

1. The introductory part of the course (approximately 1.5 weeks) will present a formal coverage of some programming languages' issues such as scoping, typing/binding, and introduce the design-based motivation for the O-O paradigm. Subsequently, different programming paradigm and representative languages will be briefly surveyed, and the O-O paradigm will be placed in context, followed by a discussion of the properties of O-O languages.
2. The next major part of the course (approx. 4 weeks) will be dedicated to the Smalltalk programming language. As somewhat of a "precursor" of the OO languages, Smalltalk is an example of a pure OO language plus among the first ones (historically) to offer the "grand vision" of all-in-one combination of a language and development environment. We will cover the basic syntactic elements in detail and overview the Smalltalk programming environment (library/hierarchy of the basic system classes). Furthermore, we will see first-hand the impact of having dynamic typing and binding; how does that affect the control-constructs and their implementation; Metaclasses, as means to ensure "everything-is-an-object"; etc.
3. The next portion of the course (approx. 3.5 weeks) will be dedicated to the C++ language. Note that this is *not an introductory* course in C++. We will discuss the "C+" (syntax) vs. "C++" aspects of the language, however, for the most part we will focus on the O-O features of this "hybrid" language (inheritance, RTTI, the brew of static typing and dynamic binding, etc.). As for the "C+" part, we will predominantly address the differences with Smalltalk. We will also address the difficulties that arise from the quest of having exceptions management in C++ inheritance settings.
4. Time-Permitting: Last few lectures will be dedicated to two "overviews": (1) the spectrum of the O-O languages, taking Smalltalk (pure) as one end, and C++ (hybrid) as the other. Specifically, we will try to position Java along this spectrum; and (2) Design Patterns.

I. Required text: "*Object-Oriented Programming with C++ and Smalltalk*", by Caleb Drake (Prentice Hall).

II. Reference text and/or other materials:
   a. Plethora of books and online sources available;
   b. Handouts (articles) will be provided in class;

III. Required Prerequisites: A background in Data Structures and Algorithms is assumed (an equivalent to EECS 311), along with the corresponding programming experience.

IV. Course Outcomes: After finishing the course, the students should be a lot more comfortable with programming in different languages from the O-O paradigm, with extra awareness about some of the executional implications of the code that they write in a particular language. As a specific example, the students will be able to recognize (and utilize the fact) that although the syntax of Java is similar to the one of C++, the executional behavior of the

programs is quite different. More importantly, the intellectual "thought-transition" between the design and coding should be highly improved as a result of this course.

V. Tentative Course Outline

Week1	- Introduction/Motivation Reading Assignment: "On Criteria to be Used when Decomposing a System into Modules" (D. Parnas, CACM) - Scoping, typing, binding - Programming paradigms overview
Week2	- Features of languages from the O-O family - Introduction to Smalltalk[1] (message expressions) *Homework #1 assigned*
Week 3	- Smalltalk Intro (cont.) Lexical Elements; Identifier vs. object semantics; Control Flow; - Introduction to Smalltalk Classes *Homework #1 due; (Mini) Project#1 assigned*
Week4	- Smalltalk Classes (cont.) - Inheritance - Metaclasses and implementation of inheritance and "everything is an object" semantics. *Project #1 due; Project#2 assigned*
Week5	- Overview of Foundation Classes The Object class and Unique objects classes; Numeric Classes, operations and cast/conversion (Double-Dispatching vs. Coercive Generality)
Week 6	- Overview of Smalltalk Collection Classes Ordered vs. Unordered *Midterm; Project#2 due*
Week 7	- Overview of C+ Improvements of C; Overloading; Pointers to Functions; - Overview of C++ classes
Week 8	- C++ Classes (cont.) - Classes and scoping rules; Operators overloading; Smart Pointers; - Inheritance in C++ *Project #3 assigned*
Week 9	- Inheritance in C++ (cont.) Inheritance and static typing; RTTI and Dynamic binding; Multiple inheritance (vs. the single one in Smalltalk) - Potpourri: Exceptions, Templates
Week 10	- Design Patterns; Placing Java in the context of O-O languages; Course wrap-up *Project #3 due*
Week 11	- Final Exam

[1] An extra lab-lecture may be held as part of the introduction during week 2.

VI. Grading

Your grades will be based on:

- 1-2 homeworks (6%)
- Project 1 (10%)
- Project 2 (18%)
- Project 3 (18%)
- Midterm (24%)
- Final Exam (24%)

Notes: The first project will not be too complicated, however, it will be a challenge of familiarizing with the Smalltalk language and environment. The 2nd and 3rd projects will be more application-oriented, with somewhat more focus on design-to-coding transition.

Closing Remarks (Awareness and Academic Responsibilities): Please be advised that it is each student's individual responsibility to keep him/herself up-to-date with the announcements made in class, distributed via email, or otherwise posted. A particular assignment may be allowed to be done in teams, however, if an assignment is indicated as individual – although you are encouraged to always discuss high-level (e.g., design, solution) issues with your classmates – it is your responsibility to ensure that the work (entire coding) is done individually. The policies for cheating are well-defined. In addition, notwithstanding our willingness to be understanding for the students' commitments and time-constraints, please do not attempt to obtain an incomplete grade for the course, based solely on your poor performance – it is against the University regulations.

Welcome and good luck!!!