Faculty of Information and Communication Technologies
Higher Education Division

Unit of Study Outline

HIT3315

Languages in Software Development

Semester 2, 2008

Version date (31 July, 2008)
## Unit of Study Outline

<table>
<thead>
<tr>
<th>Unit of study code</th>
<th>HIT3315</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unit of study name</td>
<td>Languages in Software Development</td>
</tr>
<tr>
<td>Teaching Term/Semester &amp; Year</td>
<td>Semester 2, 2008</td>
</tr>
<tr>
<td>Contact Hours (hrs/wk) or total contact hours</td>
<td>4</td>
</tr>
<tr>
<td>Prerequisites</td>
<td>two university-level programming subjects</td>
</tr>
<tr>
<td>Corequisites</td>
<td></td>
</tr>
<tr>
<td>Credit Points</td>
<td>12.5</td>
</tr>
</tbody>
</table>

## Aims

HIT3315 – Languages in Software Development studies modern programming languages, with the emphasis on design and semantics. The study of programming languages is primarily concerned with the following questions:

- What are the expressive ways of specifying computational processes?
- How can a given problem be effectively expressed?
- What are suitable formal frameworks for the definition and the implementation of a programming language?

Programming languages are the main technology that makes it possible for humans to instruct computers; however current programming languages are too hard to use. They take years of study to master, and there are not enough people with the necessary training. It also seems that there won’t be enough people with this training anytime soon.

Knowing how to solve problems using the different paradigms is important for several reasons. You can find solutions to problems more surely if you have many different ways to approach problems. In the twenty-first century you will not necessarily be programming in FORTRAN or C; if you can program in a language such as Java, C#, or Ada, or other new languages you will be much in demand. As parallel programming becomes more important, the use of functional (and declarative) languages may increase. Functional programming is also a key technology for supporting domain-specific languages. See also “Why Functional Programming Matter” by Prof. John Hughes.

Understanding the strengths and weaknesses of the various paradigms is important in applying them to solve problems. Problems in the real world are not labelled with the paradigm that should be used to solve them, so the choice of paradigm will be important. In programming language and software engineering research, understanding the strengths and weaknesses of the existing paradigms is important for designing better ways to program.

Even if you do not become a programmer, the ideas like function abstraction, infinite data structures, continuations, referential transparency have important applications in all areas of professional software development and in many other contexts such as mathematics and engineering.
Learning Objectives

After successfully completing this unit, you should be able to:

1. Solve programming problems using the imperative, functional, and object-oriented paradigms.
2. Interpret the tradeoffs and issues involved in the design of various language features.
3. Assess strengths and weaknesses of compiled and interpreter-based languages.
4. Describe the strengths and limitations of the imperative, functional, and object-oriented paradigms for solving different kinds of problems (or in different application domains), especially in relation to each other.
5. Explain and answer questions about specific languages that illustrate different paradigms, including questions about relevant concepts and major features.
6. Evaluate programming language features and designs based on their use in building domain-specific abstractions or in doing meta-programming.
7. Design, define, and evaluate parts of programming languages or similar systems and justify your design decisions. Justifications can be by:
   • Referring to known programming language concepts,
   • Referring to the semantics of the features,
   • Making analogies to features in specific languages that illustrate the different paradigms and their success or limitations, or
   • Making some more direct argument or proof.

Content

- Introduction
  - Basic concepts
  - History of programming languages
- Inductive sets of data
  - Sets, set builders, BNF, EBNF
  - Classification of grammars
  - Induction
  - Recursive program specification
- Introduction to Lambda Calculus
  - What is computable?
  - Lambda calculus – syntax and semantics
  - Evaluation orders - the Church-Rosser Property
- Functional Language Concepts
  - Functions
  - Call-by-Value
  - Conditionals
  - Loops
  - Lazy evaluation
- Imperative Language Concepts
  - Assignment
  - Call-by-Reference
  - Sequencing
- Object-Oriented Language Concepts
  - Objects
  - Classes
  - Inheritance
- Hypertext Language Concepts
  - Links
- Typing
  - Manifested types
  - Type inference
Learning and Teaching Structure

Lecture: Wednesday 11:30 – 13:30, BA801 (2 hours per week)
Laboratory: Wednesday 13:30 – 15:30, BA601 (2 hours per week)

In a Semester, you should normally expect to spend, on average, twelve and a half hours of total time (formal contact time plus independent study time) a week on a 12.5 credit point unit of study.

Teaching Staff

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Campus &amp; Room No.</th>
<th>Phone No.</th>
<th>Email Address</th>
<th>Consultation Times</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dr. Markus Lumpe</td>
<td>Convenor</td>
<td>EN 508c</td>
<td>5272</td>
<td><a href="mailto:mlumpe@swin.edu.au">mlumpe@swin.edu.au</a></td>
<td>By appointment</td>
</tr>
</tbody>
</table>

Resources and Reference Material

- Robert W. Sebesta, *Concepts of Programming Languages, 8th Edition*
- Seyed H. Roosta, *Foundations of Programming Languages – Design and Implementation*
- Daniel P. Friedmann et al., *Essentials of Programming Languages, 2nd Edition*
- David Flanagan, *Java in a Nutshell, 5th Edition*
- Tom Copeland, *Generating Parsers with JavaCC*
- Daniel P. Friedman and Matthias Felleisen, *The Little Schemer, 4th Edition*
- David Watt, *Programming Language Concepts and Paradigms*
- David A. Schmidt, *The Structure of Typed Programming Languages*
- Terrence W. Pratt and Marvin V. Zelkowitz, *Programming Languages - Design and Implementation*
- Harold Abelson et al., *Structure and Interpretation of Computer Programs*
- Carl A. Gunter, *Semantics of Programming Languages*
- John C. Mitchell, *Foundations for Programming Languages*
- Richard Bird, *Introduction to Functional Programming using Haskell*

Additional references will be given occasionally during the semester.

Blackboard Site for this Unit of Study

Important information concerning this unit of study is placed on the Swinburne course management system (Blackboard), accessible via [http://blackboard.swinburne.edu.au/](http://blackboard.swinburne.edu.au/)

It is your responsibility to access on a regular basis
- the Blackboard site for your unit of study,
- the Announcements section on Blackboard, and
- any emails sent by the teaching staff to your student email address via Blackboard.

If you access your email through a provider other than Swinburne, it is your responsibility to ensure that your Swinburne email is redirected to your private email address. To redirect your Swinburne email, go to [https://www.swin.edu.au/chdets/login.php](https://www.swin.edu.au/chdets/login.php)

- Type in your ID number and your password. Note your ID number is the first six numbers of your student ID and your password is initially set to your date of birth in the format DDMMYY
- Under ‘Email forwarding’ enter the email address to which you would like your Swinburne emails to be forwarded.
- Now click ‘Change’
Assessment

There will be regularly scheduled problem sets and programming assignments to help you learn the material and to allow us to evaluate your progress.

Assessment Task Details:

Problem sets will be handed out roughly every week or two. Most assignments will require laboratory work. You should expect to work on a problem set between two and four hours. If you have trouble finding a solution, ask for help! All assignments are fair and reasonable. No problem set will require more than six hours. Handouts for all problem sets will be made available online.

<table>
<thead>
<tr>
<th>Assessment Task</th>
<th>Individual/Group Task</th>
<th>Related Learning Objective(s)</th>
<th>Weighting</th>
<th>Due Date</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-10</td>
<td>individual</td>
<td>1-7</td>
<td>25%</td>
<td>As specified on the assignment handouts</td>
</tr>
</tbody>
</table>

Minimum Requirements to pass this Unit of Study:

There will be one mid-term test and a final exam. The final grade for the course will be weighted towards exams.

In order to achieve a pass in this unit of study, you must achieve a total overall mark of:

- **50% or more** for the assignments,
- **50% or more** for the mid-term test, and
- **50% or more** for the final exam.

Failure to submit assignment work may lead to disqualification from special examinations.

Assessment Criteria:

The final grade is calculated as follows:

\[
\text{Final grade:} = \frac{25\% \text{ homework grade} + 25\% \text{ mid-term test grade} + 50\% \text{ final exam grade}}{3}
\]

Your grade is independent of anyone else's grade in this class. That is, we do not grade on a curve, and everyone can get an HD. Our purpose in grading is to uphold a standard of quality and to give you feedback: it is not to rank students.

All problem sets are equally weighted. In general every problem set is worth a different number of points. The maximum number of points equals 100%. You receive one grade for all problem sets, which calculated as follows:

Homework grade = sum of all homework’s / number of homework’s

Example:

- Problem set 1: 50 out of 75 = 67%
- Problem set 2: 67 out of 80 = 84%
- Problem set 3: 89 out of 90 = 99%

Grade: 67% + 84% + 99% / 3 = 83%

Occasionally, you may earn some extra marks for a problem set, if this problem set is particular challenging. Therefore, the final homework grade may be greater that 100%. Extra marks can help you to improve your final grade.
Submission of Assignments:
Homework problems are due on paper at the date and time specified on the handouts. In general, problem sets are due just before the lecture starts. If you have problems with a particular assignment, talk to the instructor before the deadline.

Extensions and Late Submissions:
Absolutely no marks for late homework will be given at any time, unless otherwise negotiated in advance with the subject convener!

Late assignment submissions will not be marked.

Assessment Results:
Students must retain all assessed material that contributes to the final result up until such time as the final results are published.

Plagiarism:
Swinburne University of Technology defines Plagiarism as the action or practice of taking and submitting or presenting the thoughts, writings or other work of someone else as though it is your own work. Plagiarism includes any of the following, without full and appropriate acknowledgment to the original source(s):

(i) The use of the whole or part of a computer program written by another person;
(ii) the use, in essays or other assessable work, of the whole or part of a written work from any source including but not limited to a book, journal, newspaper article, set of lecture notes, current or past student’s work, any other person’s work, a website or database;
(iii) the paraphrasing of another's work;
(iv) the use of musical composition, audio, visual, graphic and photographic models,
(v) The use of realia, that is objects, artefacts, costumes, models and the like.

Plagiarism also includes the preparation or production and submission or presentation of assignments or other work in conjunction with another person or other people when that work should be your own independent work. This remains plagiarism whether or not it is with the knowledge or consent of the other person or people. It should be noted that Swinburne encourages its students to talk to staff, fellow students and other people who may be able to contribute to a student’s academic work but that where independent assignment is required, submitted or presented work must be the student’s own.

Enabling plagiarism contributes to plagiarism and therefore will be treated as a form of plagiarism by the University. Enabling plagiarism means allowing or otherwise assisting another student to copy or otherwise plagiarise work by, for example, allowing access to a draft or completed assignment or other work.

Assessment and Appeals Policy and Procedure:
The information outlined in the Assessment sections above is covered in more detail in Swinburne’s Assessment and Appeals Policy and Procedure. Students must be familiar with the Policy and Procedure, found at http://ppd.swin.edu.au/stuinf/AssessmentAndAppealsHigherEducation.htm.

The Policy and Procedure provides details about:

• Assessment issues such as the conduct of examinations, plagiarism policies and details explaining how to apply for a review of results and other appeals, and
• Student progress issues such as unsatisfactory academic progress and early intervention procedures, and
• Information for students with disabilities and special needs and procedures for applying for special consideration.
Students should make themselves familiar with all aspects of the Policy and Procedure, as failure to do so is not grounds for appeal.

Students are advised to seek advice from the staff at the Swinburne Student Amenities Association SSAA (http://www.swinburne.edu.au/ssaa/) if they require assistance with advocacy for Sections 12 (At-Risk and Progress Review) and 13 (Appeals) of the Policy and Procedure.

**Student Feedback:**

Swinburne seeks student feedback in a number of ways, including through periodic “Student Feedback on Units” and “Student Feedback on Teaching” surveys, as part of the university’s approach to quality assurance and improvement. Possible improvement based on both student and staff feedback is considered by Unit Convenors, Unit Panels made up of relevant teaching staff, Program Panels, Faculty Academic Committees, and the Academic Programs Quality Committee, as appropriate.

**Safety Standards and Conduct Requirements:**

The University executes safety drills without warning. Be prepared to follow instructions from staff and/or wardens to evacuate the building in a safe and orderly manner.

All students are expected to respect the rights and sensibilities of their fellow students and teaching staff. This also applies in respect of the content of video and audio work submitted for assessment. The University has rigorous anti-discrimination and harassment policies and procedures [http://ppd.swinburne.edu.au/humres/AntiDiscrimination.htm](http://ppd.swinburne.edu.au/humres/AntiDiscrimination.htm).

Safety procedures in laboratories must be followed. Open-toed shoes are not permitted in certain laboratories. Drink or food is not permitted in teaching spaces. The supervisor is authorised to exclude students for dangerous or disruptive behaviour which would result in forfeiture of all marks for the laboratory activity. The playing of computer games is not allowed in the computer labs.

**Special Needs**

If you have special needs you should advise your Faculty and the Unit of Study Convenor by the end of the second week of the teaching period. In addition, you are recommended to notify the Equity Office if you have not already done so.

See also the "Students with Disabilities and Special Needs" Section of the Assessment and Appeals Policy & Procedure, at [http://ppd.swin.edu.au/stuinf/AssessmentAndAppealsHigherEducation.htm](http://ppd.swin.edu.au/stuinf/AssessmentAndAppealsHigherEducation.htm).