QFO-AP-FI-MO02	اسم النموذج: Course Syllabus	جامعة فيلادلفيا Philadelphia University
رقم الاصدار : 1 (Revision)	الجهة المصدرة: كلية تكنولوجيا المعلومات	
التاريخ :2017/11/05	الجهة المدققة: عمادة التطوير والجودة	
عدد صفحات النموذج:		

Course Syllabus

Course Title: **Software Construction**	**Course code: 721420**
Course Level: 3	**Course prerequisite (s) and/or corequisite (s):** 721284, 760261
Lecture Time: 11:15-12:30 Mon. & Wed.	**Credit hours: 3**

Academic Staff Specifics

Name	Rank	Office Number and Location	Office Hours	E-mail Address
Dr Samer Hanna	**Assistant Professor**	**IT - 313**	10:00-11:00 Sun., Tues. & Thur.	shanna@philadelphia.edu.jo

Course/ module description

This course presents general principles and techniques for disciplined low-level software coding: mapping design outcomes into code, mapping architecture to code, selection of appropriated programming language to specific application, code development with errors avoidance techniques, code development with errors tolerance techniques, code development with API, and code development environments programming and GUI builders. An introduction to code testing will close the course

Course/ module objectives

This course aims to:

- The needed strategies for mapping the architecture and design of applications into source code
- Develop code with errors avoidance techniques
- To discuss the latest technologies for building software applications and knowing which of these technologies to use depending on the requirement specifications of our software problem.
- Preparing the student to pursue their career in software construction by introducing to them the needed software skills to achieve this target.

Course/ module components

- **Books (author (s), title , publisher, year of publication)**

1. S. McConnell Code Complete- Software Construction, Microsoft Press, 2nd edition, 2004. Available as a *Google* book at [http://books.google.jo/books?id=3JfE7TGUwvgC&printsec=frontcover&hl=ar&source=gbs_ge_summary_r&cad=0#v=onepage&q&f=false]

2. B. Buegge & A. Dutoit, "Object-Oriented Software Engineering, using UML, Pattern and Java", Second Edition, 2004.

3. R. N. Taylor, N. Medvidovic and E. Dashofy, "Software Architecture Foundations, Theory, And practice" John Wiley & Sons, 2010.

- **Support material (s)**
- **Study guide (s) (if applicable)**
  -Part1/ mapping design to code techniques
  - Part II/coding techniques and tools
- **Homework and laboratory guide (s) if (applicable)**
  - attached Homework and Practical work sheet

Teaching methods

Duration: 16 weeks, 60 hours in total. Lectures: 30 hours, 2 per week. Tutorial: 12, 1 per week. Laboratories: 3 in lab and 15 hours in total, 1-hour per week (personal). The last week is reserved to practical works examination.

Learning outcomes

A student completing this module unit should be able to:

Knowledge and understanding of

1. The methods and processes that should be followed to map a certain requirement, architecture or design of a software application to the corresponding implementation. (A2, A3)
2. The current available software technologies and justifying which of these technologies to choose depending on the problem to be solved. (A2)
3. The needed methods to correctly implement and document solutions to significant computational problems. (A2, A3)
4. The application of computing in a business context. (A8)
5. The needed methods to debug a program using appropriate debugging strategies and distinguish between error types. (A2)
6. Evaluation of the tools and techniques in software construction. (A2)
7. The knowledge to develop software applications in development environment that makes use of commonly supported tools. (A2)

Cognitive skills (thinking and analysis)

8. Analyze a software problem or architecture and think of the most proper way to map them to the suitable programming language and technology. (B2)
9. Design, write and debug computer programs in appropriate languages. (B3)
10. Solve real software construction problems from the software industry. (B3)
11. Identify a range of solutions and critically evaluate and justify proposed design solutions. (B4)
12. Analyze, transform, improve, and validate software applications. (B2)
13. Evaluate software applications in terms of general quality attributes and possible trade-offs presented within the given problem. (B6)

Practical skills

14. Use API libraries for software construction. (C1)
15. Write code in both Java and C# programming languages (C3)
16. Develop desktop and Web applications (C3).
17. Prepare and deliver coherent and structured verbal and written technical report.(C7)
18. Use the scientific literature effectively. (C8)

Transferable skills

19. Process data. (D1)
20. Solve problems (D3)
21. Use creativity (D2)
22. Communicate effectively with non-specialist as well as computer scientist, (D4)
23. Give oral presentations and write report and technical documents. (D5)

Assessment of Learning Outcomes

Learning outcomes (1-7) are assessed by examinations. Learning outcomes (8-23) are assessed by tutorials, laboratory work, projects and examinations.

Assessment instruments

- **Class works**: 15 (project and quizzes)
- **Practice** (construction tool): 05
- **Final examination:** 40
- **Short Examinations:** 2 x 20

Allocation of Marks	
Assessment Instruments	**Mark**
First examination	**20**
Second examination	**20**
Final examination: 40 marks	**40**
Reports, research projects, Quizzes, Home works, Projects	**20**
Total	**100**

Documentation and academic honesty

- Documentation style (with illustrative examples)
  - Practical works reports must be presented according to the style specified in the homework and practical work guide
- Protection by copyright
- Avoiding plagiarism
  - Any stated plagiarism leads to an academic penalty

Course/module academic calendar

week	Basic and support material to be covered	Practical Work (PW) and Examinations
(1)	a. Introduction to Software Construction such as: its importance, involved tasks and activities, types etc. b. Background about the architecture and detailed design of software applications	
(2)	Implementation Strategies (Available techniques to assist in the development of the implementation from the design including: generation technologies, frameworks and middleware, and reuse.	
(3)	Mapping design outcomes into Code/ derivation of code from design models - Application of Object Model (Class Diagram): UML	**1st PW**

(4)	Mapping the Object Model to a Database Schema	
(5)	Construction tools technology/ Techniques and tools supporting code dev. API use, Code reuse and libraries	
(6)	Steps of Building a Routine Constructing a routine by using the Pseudocode Programming Process (PPP)	**First examination**
(7)	Choosing the appropriate programming language depending on the requirement specifications (Discussing the characteristics of the well-known programming languages and when each of them can be used)	
(8)	Defensive Programming	**2nd PW**
(9)	Error Handling Techniques	
(10)	Code-Tuning Strategies	
(11)	Error avoidance→ Vulnerable techniques: Program Flow Control breaking (goto, continue, exit, break, …), Recursion, Global data, parameters by value and by references	**Second examination**
(12)	Service Oriented Architecture (SOA) and Web Services (A new paradigm of building software applications)	**3rd PW**
(13)	Distributed Middlewares	
(14)	Software construction Tools/ **Development environments, GUI builders -1**	
(15)	- Unit testing tools	
(16)	Practical Exam	**Final Examination**

Expected workload

On average students need to spend 2 hours of study and preparation for each 50-minute lecture/tutorial.

Attendance policy

Absence from lectures and/or tutorials shall not exceed 15%. Students who exceed the 15% limit without a medical or emergency excuse acceptable to and approved by the Dean of the relevant college/faculty hall not be allowed to take the final examination and shall receive a mark of zero for the course. If the excuse is approved by the Dean, the student shall be considered to have withdrawn from the course.

Module references

Books

1. M. Blaha and J. Rumbaugh, “Object-Oriented Modeling and Design with UML”, Second editions PEARSON Prentice Hall 2005.
2. D. Brugali and M. Torchiano, "Software Development Case Studies in Java", Addison Wesley, 2005.
3. P. Tahchiev, F. Leme, V. Massol and G. Gregory, "JUnit in Action", Second Edition, MANNING, 2011.