Curriculum Autumn 2017*

Bachelor in IT - Programming
Faculty of Technology

All information revised as of October 2016. Note that all curricula may be updated*
1 Introduction

Programming means developing software. In this process we use a programming language that tells the machine how to perform the operations, as well as giving instructions to be carried out after actions have been completed. Good software depends on successful cooperation between humans and machines, and competent programmers and system developers do the job of developing innovative, effective and functional software.

As programming languages and techniques change, new dialects or generations of programming languages emerge. Architectures are all the time being challenged to implement new concepts (Web 2.0, 3rd generation script language, SOA, social web). Programming and architecture are used as a foundation for most services in the public and private sectors. There is a need for proficient programmers with broad knowledge, from local applications to distributed SOA solutions with mobile clients. And there is a strong need for the capability to build not only by plan but to see possibilities or solutions when designing the architecture. Paradigm shifts in programming need to be watched and weighed against existing technology.

Having studied advanced programming you can be the programmer to make the next trendsetting application. Demand is great for programmers and system developers with the right qualifications. The Faculty of Technology at Westerdals Oslo ACT is a leading technology education provider and will be well suited for those who have a logical mind and love to develop. There is great demand for our candidates in the market.

In the bachelor’s programme in programming you learn to build advanced technological solutions for computers and mobile phones. You are educated to construct architectures from the bottom up, to design systems and develop attractive applications – all of it through a thorough education in the programming language Java. The IT daily life is much more complex today than it was a few years ago. Now it is important to be able to see how applications can cooperate across machine boundaries, between websites, and on mobile phones. Programming will give you the tools and knowledge you need and at the same time giving you a thorough introduction to support technologies such as XML, script language and storage media. All this will prepare you for taking future digital life a step further.

The specialisation in programming educates competent programmers on all levels of the architecture Java Stack. You will be capable of viewing your own information architecture in a critical way, arguing strengths or weaknesses in it or alternatives to it.
1.1 Curriculum overview

The Programming study builds on the first year of Bachelor in IT at Westerdals Oslo ACT. The programme is a 3-year bachelor’s programme awarding the degree Bachelor in IT.

The structure of the curriculum 2016-2017 is as shown in the matrix below.

*Please note that the course Enterprise Programming 1 in the 5th semester will be replaced by the course Web Development and API Design in academic year 2017-2018. In academic year 2016-2017 Enterprise Programming 1 is taught in the 4th as well as the 5th semester.*

<table>
<thead>
<tr>
<th>Sem6</th>
<th>Research Methods (7.5 ECTS)</th>
<th>Bachelor's Thesis (22.5 ECTS)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sem5</td>
<td>Elective (7.5 ECTS)</td>
<td>Programming in C for Linux (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enterprise Programming 1 (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enterprise Programming 2 (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem4</td>
<td>Project Software Engineering (15 ECTS)</td>
<td>Ruby on Rails (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mobile Development (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Enterprise Programming 1 (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem3</td>
<td>Advanced Java Programming (7.5 ECTS)</td>
<td>Software Design (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Algorithms and Data Structures (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem2</td>
<td>Elective (7.5 ECTS)</td>
<td>Object Oriented Programming 2 (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Information Security (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Iterative Web Project (7.5 ECTS)</td>
</tr>
<tr>
<td>Sem1</td>
<td>Databases 1 (7.5 ECTS)</td>
<td>Object Oriented Programming 1 (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Digital Technology (7.5 ECTS)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Creative Web Project (7.5 ECTS)</td>
</tr>
</tbody>
</table>

Yellow boxes denote that the course is taught jointly with one or more other study programmes. Green boxes show the location of optional courses (electives) in the curriculum. In the 2nd semester students choose an elective from a pool of optional courses across faculties and programmes, in addition to the electives offered by the Faculty of Technology.

**Electives in the 2nd semester (Spring)**
- DS2100 Animation (7.5 ECTS)
- DS2200 Digital Culture (7.5 ECTS)
- PG2201 Unity Development (7.5 ECTS)

**Electives in the 5th semester (Autumn), Bachelor in Programming**
- PG5200 Tools Programming (7.5 ECTS)
- PRO300 Virtual Reality Project (7.5 ECTS)
PG5600 iOS Programming (7.5 ECTS)

It is possible to choose other electives than the three Faculty of Technology electives listed above, however, students must then make sure that the chosen course does not collide with other teaching.

The first year of study is jointly taught for all programmes and gives solid basic qualifications in programming, project work, system development, data technique and databases.

The second year continues the Java learning from the first year. Further, there is an introduction to operating systems, and a thorough introduction to mobile programming (Android), web programming in Ruby and data structures with XML. Last but not least, there is project work in the course Project Software Engineering, which gathers previous learning in the study into an extensive, group-based project over two semesters.

In the third year the emphasis is on total understanding of distributed systems and architecture, by thorough training in distributed Java, represented by Java Enterprise Edition. Algorithms also have a central place on this level of the study programme, and additionally the third year offers jointly taught courses in business knowledge and research methods. The third year also includes an elective. The study finishes with a main project, a bachelor’s thesis, carried out in companies where the students solve a real-life case provided by an employer.

1.2 About the programme

The aim of Bachelor in IT – Programming is to educate people with sound knowledge of the whole spectre from local applications to distributed solutions with mobile clients. The students will learn to see solutions and possibilities when designing software, and will take an interest in following the paradigm shifts in programming. The emphasis on real-life practical work in cross-disciplinary teams provides the students with important experience in the work with complex issues and a broad basis to succeed in their further careers, developing effective, maintenance-easy and functional software that facilitates good human-machine cooperation.

At completion of the programme the candidates will be characterised by a learning outcome consisting of the following knowledge, skills and general competence:

Knowledge – the candidate

- has wide knowledge of programming with an emphasis on object oriented programming, as well as a basic understanding of the functioning of a computer and how we can instruct it by using relevant programming languages
- knows various methods for developing software, and also knows older methods such as the waterfall model, but mainly as a contrast to newer and more flexible development methodology that has particular emphasis in the programme
- understands the purpose of writing automated texts to support production code, thereby enabling flexible development
- knows how a programmer can model and use various types of databases as storage media
- is acquainted with information security and how to use various security measures to protect information
• understands the importance of writing program code with a good design, that can be easily understood and can be maintained over time
• knows how to update their own knowledge of programming through relevant web-based networks, conferences and literature, and also understands that self-development can be a natural consequence of work methodology, such as using par programming, review of code, and other forms of cooperation
• knows the history of programming and the paradigms that have been, and are, leading in this field

Skills – the candidate is able to
• in communication and cooperation with a customer, identify, define and analyse complex information technological questions
• specify, design, implement and evaluate information technological solutions individually and in cooperation with others
• program in several programming languages, using adequate programming tools and development environments
• apply ideas and knowledge from research and development on practical and theoretical tasks in information technology
• reflect on their own work by evaluating feedback from others regarding their work, and also by evaluating the solutions and approaches of others

General competence – the candidate
• understands the concept technical debt, and how to influence the degree of technical debt by developing software
• can communicate in writing and orally with fellow professionals, and also with end users without technical competence, and is able to present professional knowledge, questions and solutions in information technology and how this technology makes an impact on society
• knows and understands how important knowledge sharing is in an industry characterised by continuous change, and understands how important it is that everybody contributes to this sharing of knowledge and how they can contribute.

1.3 Central themes
Bachelor in IT – Programming has the following central themes and research anchoring:
• Programming in Java – from Standard Edition (SE) to Enterprise Edition (EE)
• Research anchoring in the object oriented paradigm
2 Individual courses offered to exchange students Autumn 2017

2.1 PG5100 Enterprise Programming 1

Norwegian name: Enterpriseprogrammering 1
ECTS credits: 7.5
Area of study: Technology/IT
Language of instruction: English
Programme: Mandatory course in Bachelor in IT – Programming
Required prerequisites: The course requires basic object oriented programming in Java and also exception handling, building system and database programming (JDBC)
Recommended prerequisites: None
Semester: The course is taught in the 4th and 5th semesters
Course leader: Per Lauvås

Course outline
The course teaches distributed systems and component-based development. It also teaches Java Enterprise Edition with supporting application architecture. The students will become skilled in developing applications based on Java EE with frameworks for persistence, presentation, and logic. The students will gain the necessary qualifications to use the correct application architecture and analyse benefits and disadvantages related to the choice of technology and implementation technique.

Learning outcome
Knowledge: At completion of the course candidate will
- be able to explain the concept of distributed systems or applications
- have gained knowledge of the concept and theories of component-based development
- know the structure, components and architecture of Java Enterprise Edition
- be acquainted with the Java Persistence library
- know the Bean concept in Java and the uses of different types of beans
- be able to explain the concepts Inversion of control, Context and Dependency Injection, and Aspect Oriented Programming
- know and understand the benefits of Continuous Integration

Skills: At completion of the course candidates will be able to
- develop parts of a distributed web application based on Java EE technology with framework
- develop component-based web interfaces
- develop and implement stateful and stateless session beans on a JEE server
- evaluate benefits and disadvantages by using stateful and stateless session beans
• store data permanently using the Java Persistence library
• develop tests for integration to database
• utilise Continuous Integration to run tests automatically

General competence: At completion of the course candidates will be able to
• use the correct application architecture and analyse benefits and disadvantages related to the choice of technology and implementation technique

**Teaching and learning methods**
Lectures, exercises and a 48-hours home examination

**Recommended workload**
Participation in lectures and tutorials – 48 hours
Self-study – 50 hours
Independent preparation for presentations and discussions in class – 12 hours
Independent exercises, lab work, practical work individually or in groups – 30 hours
Examination and preparing for the examination – 60 hours
**Total recommended workload – 200 hours**

**Technology and tools**
Java EE 7, optional choice of Java EE IDE

**Learning material/Syllabus**
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

**Coursework requirements**
None

**Assessment**
Assessment is based on an individual home examination of 48 hours. The examination tests learning outcome for knowledge as well as skills.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

**Assessment criteria**
See Learning outcome

**Notes**
In academic year 2016-2017 this course is taught in the 4th semester for students in their second year of study, and the 5th semester for students in their third year.
2.2 PG6100 Enterprise Programming 2

Norwegian name: Enterpriseprogrammering 2

ECTS credits: 7.5

Area of study: Technology/IT

Language of instruction: English

Programme: Mandatory course in Bachelor in IT – Programming

Required prerequisites: The course requires completion of the course PG5100 or equivalent previous knowledge

Recommended prerequisites: None

Semester: The course is taught in the 5th semester (Autumn)

Course leader: Per Lauvås

Course outline

The course is a continuation of PG5100 Enterprise Programming 1. While PG5100 gave an introduction to JEE among other things via the use of presentation and persistence, this course continues with security, transaction handling, messages, and web services. The course will provide students with skills in developing full-stack applications based on Java EE. The course makes a point of referring to previously gained knowledge, and of contributing to a community by making one’s self-developed solution available to others.

Learning outcome

Knowledge: At completion of the course candidates will
- know security and transactions in JEE
- be able to draft and explain the concepts synchronous and asynchronous communication
- know how messages function in JEE
- know and understand the benefits of Continuous Delivery
- know how web services function (in SOAP and REST) and when it is expedient to use them
- know how to mock and stub web services

Skills: At completion of the course candidates will be able to
- make a distributed application that handles communication between client and server in a network
- make a distributed web application based on Java EE technology with framework
- develop and implement message beans based on a JEE server
- make active use of security mechanisms from JEE for admission control and user management
- implement transactions in a JEE application
- implement Web Services in a JEE application
- automate acceptance testing via an adequate tool

General competence: At completion of the course candidates will be able to
- present a self-developed JEE application with thorough documentation in a way that will make it possible for as many people as possible to learn how the application was developed and how the various involved technologies function together.
Teaching and learning methods
Lectures and exercises, guidance in connection with self-chosen full-stack applications; final deployment of self-chosen application (prototype) with documentation, and a final examination

Recommended workload
Participation in lectures and tutorials – 48 hours
Self-study – 50 hours
Independent preparation for presentations and discussions in class – 12 hours
Independent exercises, lab work, practical work individually or in groups – 30 hours
Examination and preparing for the examination – 60 hours
Total recommended workload – 200 hours

Technology and tools
Java EE 7, optional Java EE IDE

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
None

Assessment
Assessment is based on two independent examinations as described below:

- A 3-hour individual written examination (25%). No aids permitted. This examination primarily tests knowledge.

- An individual written examination (75%) in which the candidates submit a self-developed application (prototype) that is openly accessible on the Internet. The application should comply with the guidelines for documentation, and with the choice of technology as described by the course leader or lecturer.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

Assessment criteria
See Learning outcome
2.3 PRO300 Virtual Reality Project

Norwegian name: Virtual Reality-prosjekt
ECTS credits: 7.5
Area of study: Technology/IT
Language of instruction: English
Programme: Optional course in Bachelor in IT – Programming; Bachelor in IT – Game Programming; Bachelor in IT – Game Design; Bachelor in IT – 3D Graphics; Bachelor in IT – Interactive Design; Bachelor in IT – Intelligent Systems

Required prerequisites: Basic qualifications in programming, interaction, game or concept development, or 3D design
Recommended prerequisites: As above
Semester: The course is taught in the 5th semester (Autumn)

Course leader: Kim Baumann Larsen

Course outline
The Virtual Reality Project is a cross-disciplinary course where students participate, bringing with them their different backgrounds. The purpose of the project is to develop interactive virtual reality applications for games, art or visualisation. The students will make concepts, design and create interactive Virtual Reality solutions, and thereby understand the technological frames in use.

Learning outcome
Knowledge: At completion of the course candidates will
- know the machine and software platforms for Virtual Reality
- know the platform requirements for different Virtual Reality solutions
- know the possibilities and limitations with Virtual Reality
- know the demands for achieving the largest possible presence in a Virtual Reality solution
- know basic principles for storytelling for interactive Virtual Reality

Skills: At completion of the course candidates will be able to
- draft and plan one interactive Virtual Reality concept for games, art or visualisation
- make prototypes for different platforms
- design and implement one interactive Virtual Reality experience for a chosen platform
- integrate 3D models, sound elements and interaction points into a holistic Virtual Reality experience for a chosen platform

General competence: At completion of the course candidates will be able to
- assess and create an optimal Virtual Reality solution for a chosen platform
- assess different Virtual Reality platforms and the possibility and limitations of solutions
Teaching and learning methods
Lectures and exercises, and one cross-disciplinary project in which students participate with their different study programme backgrounds either in programming, interaction, games/concept development, or 3D design.

Recommended workload
Participation in lectures and tutorials – 25 hours
Self-study – 80 hours
Independent preparation for presentations or discussions in class – 5 hours
Student work with projects, productions, assignments etc. – 20 hours
Independent exercises, lab work, practical work individually or in groups – 70 hours
Total recommended workload – 200 hours

Technology and tools
HTML editor

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
None

Assessment
Assessment is based on a combined examination consisting of the following elements:

- A group project examination (60%) where the groups submit the results of the project they have been working with in the course
- A group written examination (20%) consisting of a group report describing how the project was carried out, as well as reflections in connection with the work (a process document). The report should have a length of 3000-5000 words. The specifications of requirements for the report are handed out three weeks before submission.
- A group oral examination (20%) in which the groups present the project result and the process. The oral examination lasts approx. 25 minutes.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.
2.4 PG5200 Tools Programming

Norwegian name: Tools-programmering

ECTS credits: 7.5

Area of study: Technology/IT

Language of instruction: English

Programme: Mandatory course in Bachelor in IT – Game Programming, optional in Bachelor in IT – Programming

Required prerequisites: PG2100 Programming 2

Recommended prerequisites: Experience with basic C# programming (as the C# part of PG3300 Software Architecture, or equivalent knowledge)

Semester: The course is taught in the 5th semester

Course leader: Kjetil Raaen

Course outline
This course teaches the challenges and solutions for the development of tools to enable members of other professions to produce content for games. The students learn to develop a simple level editor and other tools needed to develop games and other multi media products. The students will also learn about the need for stability, error handling and fast reaction.

Learning outcome
Knowledge: At completion of the course candidates will

- understand basic functionality in a game engine, and what the concept of tool chain means
- know various types of game engines and the difference between pure graphics engines and complete game engines
- understand why good tools are essential for effective production of content
- be able to explain the different categories of tools used in game development, including level editors, property editors, and support tools
- be able to describe the most important functionality in a level editor
- know third party solutions for game tools
- know network protocols and how games communicate over networks

Skills: At completion of the course candidates will be able to

- evaluate the benefit of making development tools oneself as opposed to using third party solutions
- develop specifications of requirement for a selection of tools
- develop tools in C# with WPF
- implement essential functionality for a level editor
- master the use and handling of exceptions for improved stability
- avoid heavy operations in GUI thread
- add support for scripting in existing code
- save game state and log events in a database
- serialise and de-serialise game data

General competence: At completion of the course candidates will

- understand how a good tool chain can improve development efficacy
• be able to reflect critically on the game development process and necessary demands

Teaching and learning methods
The course is taught in the form of 12 lectures of about 2 hours each and 12 teacher-led exercises of 2 hours each. The teacher-led exercises are not mandatory but to reach the competence goals the students are expected to complete the exercises on their own if the scheduled exercises are not sufficient. Additionally, the students have to focus on the home examination.

Recommended workload
Participation in lectures and tutorials – 24 hours
Self-study – 90 hours
Independent exercises, lab work, practical work individually or in groups – 24 hours
Examination and preparing for the examination – 62 hours
Total recommended workload – 200 hours

Technology and tools
Visual Studio IDE

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
None

Assessment
Assessment is based on an individual written home examination lasting 4 weeks.
Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

Assessment criteria
The candidates are assessed by their ability to develop usable tools that can be integrated in a tool chain for the development of games and similar applications.
2.5 PG5600 iOS Programming

**Norwegian name:** iOS-programmering  
**ECTS credits:** 7.5  
**Area of study:** Technology/IT  
**Language of instruction:** English  
**Programme:** Mandatory course for Bachelor in IT – Mobile Programming, optional for Bachelor in IT – Intelligent Systems  
**Required prerequisites:** Advanced knowledge of programming (from 2nd year Bachelor in IT) or equivalent previous knowledge in object oriented programming  
**Recommended prerequisites:** None  
**Semester:** The course is taught in the 5th semester (Autumn)  
**Course leader:** Tor-Morten Grønli

**Course outline**

The course gives an introduction to programming in Swift and the iOS platform. At completion of the course the students will be able to make applications that communicate over networks, store data locally, and use interface elements and patterns that go with the platform.

**Learning outcome**

**Knowledge:** At completion of the course candidates will
- have gained knowledge of the architecture of the iOS platform  
- be able to describe the life cycle of an iOS application using text and drawing  
- know the process of distributing applications  
- know how applications are compiled on iOS  
- be able to describe MVC, the observable and delegate pattern in context with iOS, using text and drawing

**Skills:** At completion of the course candidates will be able to
- program basic Swift, including
  - data types  
  - use of foundation classes  
  - control structures  
  - use of object orientation  
  - use of protocols  
  - use of extensions  
  - use of closures  
  - error handling  
- use iOS APIet actively when programming applications  
- apply the usual GUI components to produce user interfaces  
- use storyboards  
- write automated texts  
- make use of basic animations  
- serialise, de-serialise and persist data
- programme to internet-based services
- account for and implement asynchronous architecture
- debug and deploy an application to simulator and unit

General competence: At completion of the course candidates will be able to
- propose and give reasons for the choice of architecture in an iOS application
- know when asynchronism is relevant
- consume a REST API

Teaching and learning methods
The course is taught with 12 lectures, each of about 2 hours, and 12 teacher-led exercises of about 2 hours each. The teacher-led exercises are not mandatory, but to obtain the competence goals the students are expected to complete exercises and additionally put in some extra effort if the teacher-led exercises are not sufficient.

Recommended workload
Participation in lectures and tutorials – 48 hours
Self-study – 110 hours
Examination and preparing for the examination – 42 hours
Total recommended workload – 200 hours

Technology and tools
iOS SDK (incl Xcode)
iOS

Learning material/Syllabus
Updated information on required reading and other learning material is posted per programme on our electronic learning platform before the semester starts. The information is also available on our website.

In addition to literature and other learning material, scheduled teaching and other scheduled learning activities are part of the syllabus.

Coursework requirements
None

Assessment
Assessment is based on an individual written examination (100%) in which candidates submit an extensive examination paper. The paper tests knowledge as well as skills. The assessment criteria for the examination paper are handed out together with the question paper 2-3 weeks before submission.

Grading scale: A-F with A as the best grade and E as the lowest pass grade. F means failed.

Assessment criteria
See Learning outcome