UNIVERSITY OF Southampton
School of Electronics and Computer Science

COMP1202 Topics Checklist

Thai Son Hoang

ECS, University of Southampton, U.K.

COMP1202
10th December 2019

Lecture 00 - Starting Out

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Object-Oriented Programming:
  - Classes and Objects
- Logic operators:
  - `==, !=, &&, ||`
  - `>, >=, <, <=`

Lecture 01 - Introduction to Java

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- From Java code to Program
  - Compile
  - Execute
- Java Virtual Machine
  - How can Java work on all different platforms
- Writing a Class in Java
  - File name convention
  - Declaration of classes, variables, methods
  - `main` method
- Conditional **if** . . **else** . . . statement
- Using brackets to group statements

Lecture 02 - Variables

UNIVERSITY OF Southampton
School of Electronics and Computer Science

- Primitive (**int**, **boolean**, **char**, **float**, . . .) vs Object types
  - Defined?
  - Stored?
  - Passed?
- Variable Scope
  - Member variables
  - Local variables

Lecture 03 - Methods

- Method parameters and arguments
- Passing arguments: Primitive vs. Object types
- Return type
  - Using collections to return many things
- Overloading
  - Method signature (its name, parameters and order of parameters)
  - Overloading requires unique signature for each method.

Lecture 04 - Computational Thinking

- Algorithm and characteristics: performanace, efficiency, understandability, scalability, reusability, reliability, elegance.
- Coding style

Lecture 05 - Encapsulation

- Accessor and Mutator Methods (aka. Getters and Setters)
- Principle of encapsulation
- **private** vs **public** keywords
- Constructors
  - Naming convention
  - Does not have a return type
  - Can be overloaded
  - Default contructor: No parameter and no behaviour

Lecture 06 - Loops and Arrays

- Loops: Syntax, condition, loop body, termination
  - **while** loop
  - **do** . . . **while** loop
  - **for** loop
- Arrays: Syntax
  - Declaration
  - Initialisation
  - Assignment
  - Retrieval
- `ArrayIndexOutOfBoundsException`
- **for** each loop: Syntax

Lecture 07 - Collections and Iterators

University of Southampton, School of Electronics and Computer Science

- Arrays vs ArrayLists
  - Declaration
  - Insertion
  - Access
  - Removal
- Genericity: Syntax
  - Autoboxing vs unboxing
- **instanceof** keyword, typecasting
- Iterators:
  - `hasNext()` method
  - `next()` method
  - Benefit of iterators

Lecture 08 - Java Library

University of Southampton, School of Electronics and Computer Science

- Implementation vs Interface.
- Importing: classes, packages, etc.
- `String` Objects
- Identity vs. equality (i.e., `==` vs `.equals()` method
- `HashMap`: collection that maps a key to a value
  - `put` method
  - `get` method

Lecture 09 - IDEs

University of Southampton, School of Electronics and Computer Science

- Source code editor
- Build automation
- Debugger
  - Breakpoints

Lecture 10 - Super- and Sub-classes

University of Southampton, School of Electronics and Computer Science

- Inheritance to avoid code duplication
- Inheritance and Encapsulation
  - **public** vs **protected** vs default (no keyword) vs **private**
  - **super** keyword for calling super-class' contructor.
  - Superclass contructor rules
- References and Inheritance
  - Subtitution
- The `Object` class

Lecture 11 - Polymorphism

University of Southampton, School of Electronics and Computer Science

- Overriding methods
- Which method gets called?
  - The first method found from the bottom of the inheritance tree is called
- Dynamic binding: is calling the most specific version of the method, even when the reference has the type of the superclass
- Polymorphism: Substitution + Overriding + Dynamic binding
- **super** keyword to access methods in the superclass.

Lecture 12 - Software Design

University of Southampton, School of Electronics and Computer Science

- Concepts for Code quality:
  - Duplication: Use methods to write code once and call many times
  - Coupling: Aims for loose coupling
  - Cohesion: Aims for high cohesion, i.e. each unit responsible for one single logical task.
- Responsibility-driven design (RDD): leads to low coupling
- Thinking ahead and refactoring

Lecture 13 - Testing and Debugging

University of Southampton, School of Electronics and Computer Science

- Error Handling: Advantage and Disadvantage of each approach
  - Print and default
  - Error codes
  - Exceptions: **try** ... **catch**
- Debugging:
  - Syntax errors
  - Logical errors
- Testing Strategies
  - Unit testing: Equivalence Class vs Boundary Value
  - Manual walkthroughs: Tabulating object state
  - Verbal walkthroughs: Pair programming
  - Print statements:
  - Debuggers

Lecture 14 - Designing Applications

University of Southampton, School of Electronics and Computer Science

- Analysis and Design
  - Noun Verb Analysis
  - Stepwise Refinement
- Software Engineering
  - Documentation
  - Cooperation
  - Prototyping
  - Growth model for iterative development
- Design Patterns
  - Pattern structure
  - Decorator pattern
  - Singleton pattern
  - Factory pattern
  - Observer pattern

Lecture 15 - Abstract Classes and Interfaces

- **abstract** keyword
  - **abstract** class
  - **abstract** method
- Multiple inheritance vs interfaces
- **extends** vs **implements** keyword