COURSE CODE
LECTURER

Joao Almeida-Domingues[*]

University of Glasgow

January 13th, 2020 – March 25th, 2020

Contents

These lecture notes were collated by me from a mixture of sources , the two main sources being the lecture notes provided by the lecturer and the content presented in-lecture. All other referenced material (if used) can be found in the *Bibliography* and *References* sections.

The primary goal of these notes is to function as a succinct but comprehensive revision aid, hence if you came by them via a search engine , please note that they're not intended to be a reflection of the quality of the materials referenced or the content lectured.

Lastly, with regards to formatting, the pdf doc was typeset in LaTeX, using a modified version of Stefano Maggiolo's class

[*]2334590D@student.gla.ac.uk

1 Modelling

Lecture 1
January 13th, 20

Object , Encapsulation , Polymorphism , Inheritance , Abstraction , Class , Methods , Attributes , End-User

This course will focus on the basic concepts of OOP. The main idea being that one can model real world objects as abstractions in software. We take the object's attributes and functions and convert them into a single entity consisting of data and methods which operate on that data

1.1 3 Principles of OOP

1.1 definition. Encapsulation when data and operations exist within the same entity

1.2 definition. Inheritance classes can inherit attributes and methods from other classes

1.3 definition. Polymorphism ability of an entity to take many forms

As seen in JP2, the data and methods are defined within the class, and are often protected from outside access unless via getter and setters. If a class is a subset of another class then it inherits all (or most) of its behaviours and attributes and so it can be *subclassed*. This ability of the superclass to take many forms depending on which child is called is one of the crucial aspects of OOP. When methods are *overridden* by child class a method of an object reference to a superclass is invoked at compile time, and it is later dispatched to the overridden of the specific class instance at run time

1.2 OO Design

Software design relies on a symbiotic relation between the end-user and the designer. Often one is given a spec/problem statement by a client, or given a certain user story (recall HCI:1F) and from there the designer will use its modelling knowledge to interpret it in the light of OOP paradigms

1. Identify real world objects (look at the nouns in the spec)
2. Identify relationships between objects

   Generalization : Abstract common features (e.g *move*)

   Containment : Object A $\subseteq$ B (e.g *Dog* $\subset$ *Animal*)

   Multiplicity : Quantity relation (e.g Dog (1) $\leftrightarrow$ (Many) Paws)
3. Identify operations and associate them with objects (this is usually done by looking at the verbs in the spec)
4. Create an Interface

   it is essentially a contract which guaranteed that each object represented by a given class will behave in a specified manner

it must include a *return type* , *purpose/description* , *pre-conditions* , i.e what must be true prior to the method being called , *post-conditions* , what must be true when returning

5. Object Encapsulation , which describes how objects communicate via operations and how this affects the end-user

References

openedge

URL: https://documentation.progress.com/output/ua/OpenEdge_latest/index.html#page/dvoop/using-polymorphism-with-classes.html.

Java - Polymorphism **tutorialspoint**

Java - Polymorphism. URL: https://www.tutorialspoint.com/java/java_polymorphism.htm.