CISC124 – Today's Topics

- **OO design summary**
  - Software core Qualities
  - Software's desired qualities
  - Software development approaches
  - Software modularity
  - Objects
  - Classes
  - Encapsulation

•1

Software's core qualities

Excellent quality software is required to have a set of:

CORE QUALITIES

- **Correct** → Accurately meets the specifications of behaviour and required outputs
- **Safe** → Maintains the integrity of systems and users
- **Efficient** → Acceptable response times to requested actions
- **Reliable** → Behaves as expected under routine and exceptional situations
- **Maintainable** → Efficiently supports the insertion of new features, improvements, corrections

•2

Software's Desired qualities

DESIRED QUALITIES

- **Extensible** → Easily supports modifications to handle scaled up problems within a limited scope (modular reusability)
- **Portable** → Capable to operate on different hardware and software with minimal modifications
- **Testable** → Easy to modularize and segregate components for testing and integration
- **Verifiable** → Easy to trace code back to desired functionality and confirm validity
- **Understandable** → Self documenting, well structured components for documentation, good documentation system

•3

Software Development Approaches

TWO MAIN APPROACHES → MAIN GOAL IS MODULARITY

- **Functional Decomposition** → Software performs a main function that can be decomposed into multiple functions, which in turn can be decomposed into functions
- **Object-Oriented Development** → Software is implemented by a set of cooperating objects that exchange functionality request messages through standardized interfaces (method invocations!)

•4

Software modularity

MAIN ADVANTAGE OF MODULARITY IS LARGE-SCALE REUSE

- **Reuse of pre-built components** → It is easier to build a hierarchy of components (i.e., libraries of classes) and establish common ways to reuse functionality (types, inheritance, objects)
- **Focused testing** → It is easier to test units of functionality in the software (i.e., testing individual classes and methods)
- **Focused debugging** → It is easier to isolate bugs to blocks of functionality that interact with each other only through pre-defined interfaces.
- **Tightly controlled modifications** → It is easier to keep under control any side effects caused by changes →minimal regression testing! (Is the system still working after the changes?)

•5

Objects

AN OBJECT IS AN OPERATIONAL ENTITY IN AN EXECUTING COMPUTER PROGRAM

- **State** → A collection of attributes holding current and relevant information about the object
- **Behaviour** → A collection of operations (methods) that the object supports.
- **Identity** → One or more attributes that uniquely identify an object as a distinct entity

•6

Objects

OBJECTS REPRESENT PROGRAM ABSTRACTIONS OF REAL (PHYSICAL) AND ABSTRACT ENTITIES

- **Problem Domain**
  - Collections of similar entities (i.e. databases)
  - Aggregations of specialized components (i.e. teams)
  - Hierarchies of specialization (i.e. Java libraries)
  - Physical systems (i.e. embedded systems)
- **Software Environment**
  - Managed software environments (.NET, Java)
  - Styled application environments (Web, GUI)
  - Specific development environments (Production lines)

Winter 2018 CISC124 – Section 2 7

•7

Classes

A CLASS IS A TEMPLATE FOR A COLLECTION OF OBJECTS WITH SIMILAR ENCAPSULATION AND BEHAVIOUR

- **Encapsulation**
  - Definitions: identification, basic (inherent) properties of objects
  - State: as a set of attributes that can change their values during program execution
  - Reference data structures
- **Behaviour**
  - Constructors: initialize object's state or configure its operational environments
  - Utility behaviours (static methods implementing algorithms)
  - Instance behaviours (instance methods changing data and state)

Winter 2018 CISC124 – Section 2 8

•8

Classes

CLASS SPECIALIZATIONS

- **Tangible things** → Physical artifacts, animals, etc.
- **Agents** → Conversion devices, decoders, sorters, etc.
- **Events** → GUI events, sensory events
- **Transactions** → Database updates, ticket reservation, etc.
- **Users and Roles** → Security, Access control
- **Systems** → Email, video-conference, etc.
- **Interfaces** → To peripherals (printers, files, displays)
- **Foundational** → Object, Strings, Math

Winter 2018 CISC124 – Section 2 9

•9

Encapsulation

PROCESS OF DEFINING A CLASS WITH AT LEAST ONE CUSTOMIZABLE ATTRIBUTE.

- **Abstraction**
  - Hide the details of the data and methods
  - Standard interface to attributes
  - Accessor and mutator methods
  - Specified interface to access methods
- **Encapsulation**
  - Reusability of code
  - Integrity and privacy of encapsulated data
  - Modularity for design, testing and expansion

Winter 2018 CISC124 – Section 2 10

•10