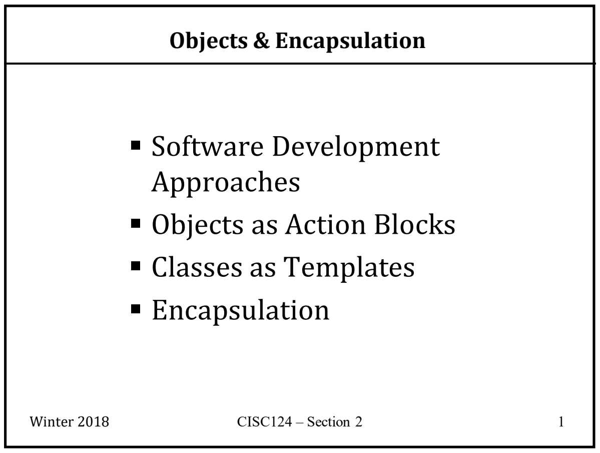
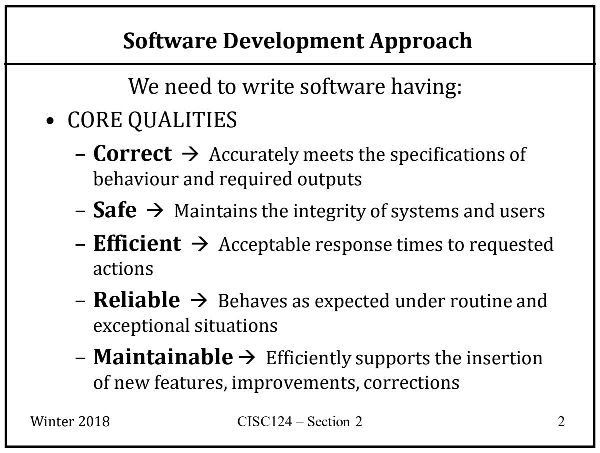


Objects & Encapsulation

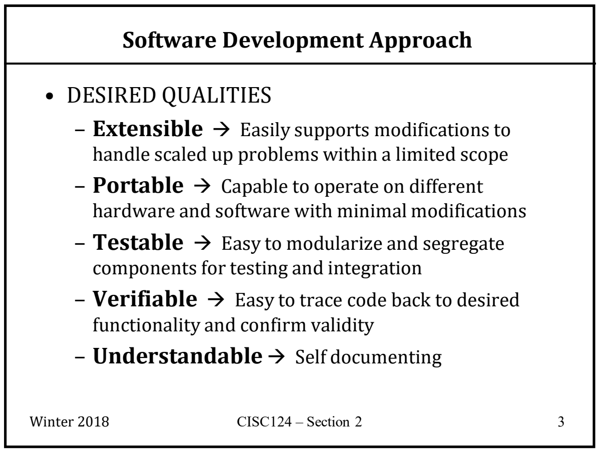
- Software Development Approaches
- Objects as Action Blocks
- Classes as Templates
- Encapsulation



Software Development Approach

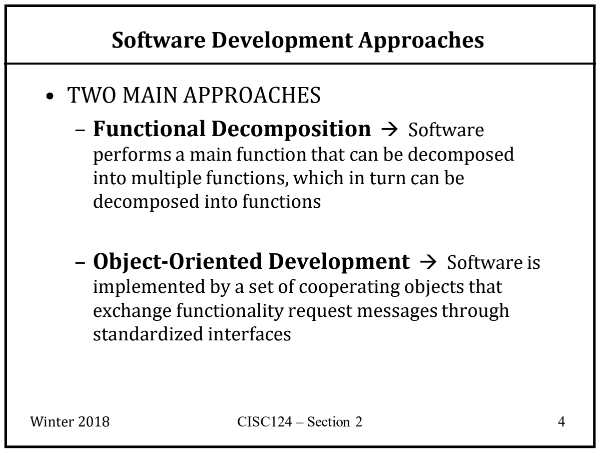
We need to write software having:

- 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



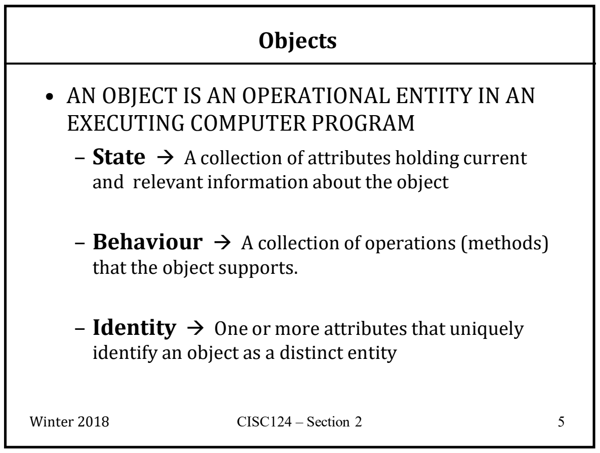
Software Development Approach

- DESIRED QUALITIES
  - **Extensible** → Easily supports modifications to handle scaled up problems within a limited scope
  - **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



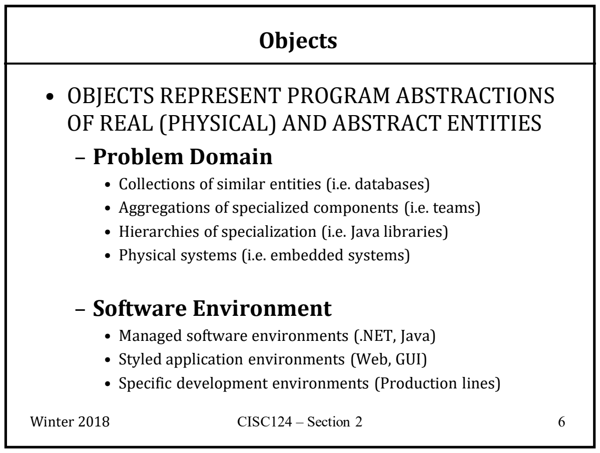
Software Development Approaches

- TWO MAIN APPROACHES
  - **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



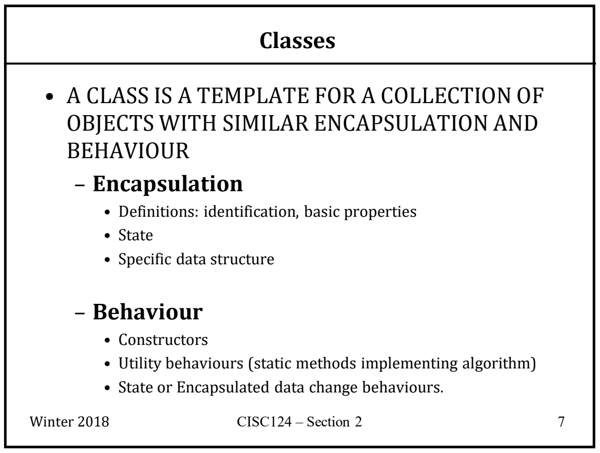
Objects

- 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



Objects

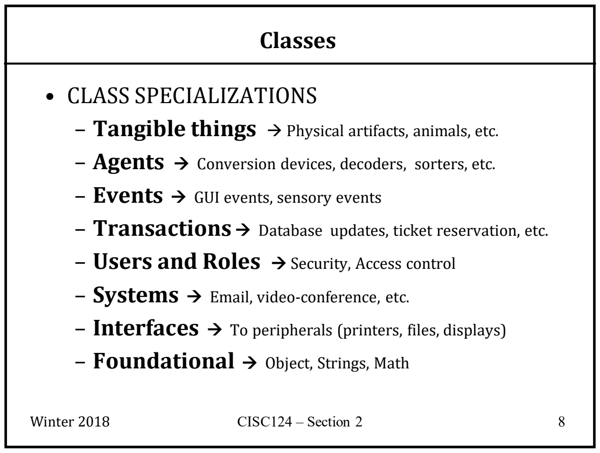
- OBJECTS REPRESENT PROGRAM ABSTRACTIONS OF REAL (PHYSICAL) AND ABSTRACT ENTITIES
  - **Problem Domain**
    - Collections of similar entities (i.e. databases)
    - Aggregations of specialized components (i.e. teams)
    - Hierarchies of specialization (i.e. Java libraries)
    - Physical systems (i.e.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



Classes

- A CLASS IS A TEMPLATE FOR A COLLECTION OF OBJECTS WITH SIMILAR ENCAPSULATION AND BEHAVIOUR
  - **Encapsulation**
    - Definitions: identification, basic properties
    - State
    - Specific data structure
  - **Behaviour**
    - Constructors
    - Utility behaviours (static methods implementing algorithm)
    - State or Encapsulated data change behaviours.

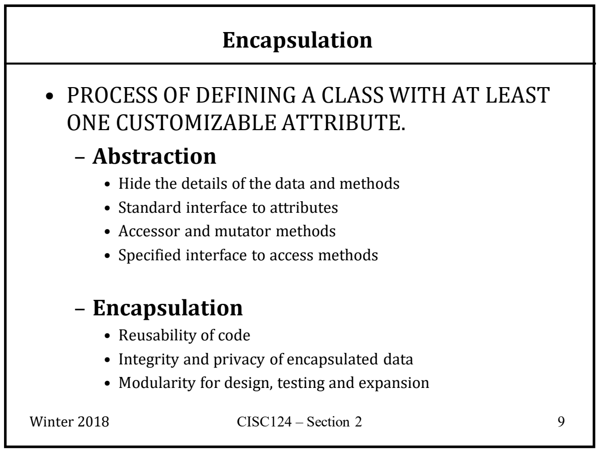
Winter 2018 CISC124 – Section 2 7



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 8



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 9