CSUS
COLLEGE OF ENGINEERING AND COMPUTER SCIENCE
Department of Computer Science

CSc 133 – Object-Oriented Computer Graphics Programming
Spring 2018
Dr. Doan Nguyen

Study Guide — Midterm Exam

The Midterm Exam will be *closed book, closed notes,* except that you will be allowed to use a single ***hand-written*** sheet of 8.5x11" paper. Note that the sheet must be ***hand-written*** (by you); computer-printed or photocopied notes are not recommended. If you use a note sheet you will be required to put your name on it and to hand it in along with your exam.

To be well prepared for the Midterm Exam, you should have *read all lecture notes, attended all lectures, completed all the reading and programming assignments* due so far. In addition, you should be able to answer questions based on the topics listed below.

Object-Oriented Concepts

(1) Understand and correctly use the basic Java/Codename One (CN1) elements that we have utilized in class such as objects, access modifiers, constructors, references, arrays, vectors, parameter passing, etc. Understand relevant aspects of the compile-execute environment such as built-in frameworks (e.g. CN1's UI package).

(2) Explain the difference between a class and an object (note that this is not Java/CN1-specific).

(3) List the four components generally recognized as comprising the "Object-Oriented" (OO) programming paradigm. Give a general description of their meaning and be able to give examples of their application.

(4) Explain what is meant by "information hiding" and "bundling", and give an example.

(5) Understand the definitions of association, aggregation, composition and dependency, and give an example of each. Explain the similarities and differences between composition and aggregation.

(6) Explain and use the elements of a UML class diagram, including general associations (using names, directions, and multiplicities), aggregation and composition, dependency, inheritance, and interfaces. Given a description of a program (set of classes), be able to draw the corresponding UML class diagram, and vice-versa.

(7) Explain the notion of "accessor" (including "mutators"), and how they relate to the OO notions of abstraction and encapsulation. Be able to explain why providing public accessor methods is not equivalent to making fields public.

(8) Show how to construct a set of classes which have associations between them so that fields are private and yet the classes can reference each other and use accessors appropriately.

(9) Explain what is meant by *overloading* a method, and give an example. Explain why constructors are commonly overloaded.

Inheritance

(10) Describe the purpose of inheritance (in the OO sense), and give an example. Include a description of the notions *superclass* and *subclass*. Tell how inheritance is expressed in Java/CN1 code, and explain the meaning and purpose of the keyword *super* in Java/CN1.

(11) Give examples of the "is-a" and "has-a" relationships, and explain their difference.

(12) List three major uses (purposes) of inheritance and give an example of each.

(13) Explain what is meant by *overriding* a method, including how it differs from *overloading*.

(14) Explain what is meant by *multiple inheritance*, including its advantages and disadvantages.

(15) Explain the OO notion of separation of interface from implementation. Describe how this is carried out in Java/CN1, and its relationship to multiple inheritance.

(16) Explain what is meant by an *abstract class* in the general OO sense. Describe how abstract classes are implemented in Java/CN1 and what constraints exist on such classes and on other related classes.

(17) Explain what is meant by an *abstract method* in the general OO sense. Describe how abstract methods are implemented in Java/CN1, and what constraints exist on such methods, on the classes which contain them, and on other related classes.

Polymorphism and Interfaces

(18) Give examples of both predefined CN1 interfaces and user-specified interfaces.

(19) Explain what is meant by polymorphism in a programming language. Give examples of the application of polymorphism in a program.

(20) Explain the meaning of the term "*dynamic* (*late*) *binding*", and how it applies in Java/CN1.

(21) Explain the difference between the *apparent type* and *actual type* of an object, and be able to use and explain each of those notions correctly in Java/CN1 code.

(22) Explain the difference between an abstract class and an interface in Java/CN1, and how Java/CN1 interfaces provide support for increased polymorphism in a program.

Displays and Color

(23) Describe how the abstraction "color" is implemented in CN1. Include a description of the notion of the "RGB color cube".

Design Patterns

(24) Explain what is meant by the term "design pattern". List the three major categories of design patterns, and give an example of each.

(25) Explain the organization, purpose and context of each of the design patterns which have been discussed in class and in the assigned reading (e.g., *iterator*, *composite*, *singleton, observer*, *command*, *strategy, proxy, factory, etc.*). Give a specific example of using each design pattern. Give appropriate UML describing each design pattern and, for those design patterns which appear as part of the CN1 language definition, be able to explain the relevant CN1 interfaces and/or classes and how they are used.

(26) Know how to implement each of the design patterns used in homework assignments.

(27) Explain what is meant by the "MVC architecture". Include an explanation of the difference between an *architecture* and a *design pattern*.

Graphical User Interfaces and Event-Driven Programming

(28) Be able to write CN1 code showing the basic steps involved in building a Graphical User Interface consisting of components such as labels, buttons, checkbox, text field, side/overflow menu items, title bar menu item, and containers.

(29) Explain the purpose of a "Layout Manager", and be able to describe the general operation of at least two CN1 layout managers.

(30) Be familiar with the event-driven operations of basic CN1 GUI components which have been discussed in class, including how events are generated by these components and how these events are handled by listeners that implement **`ActionListener`** interface. Describe two listener approaches: (i) a container which is an "event listener" for its own components, and (ii) separate listener object(s).

(31) Explain the relationship between the CN1 **`Command`** and **`Button`** classes and the elements of the Command design pattern.

(32) Explain what key bindings are and how they are implemented in CN1.