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 — Final Exam
(Draft Version: 5/2/2018)

The Final Exam will be *closed book, closed notes, with* ***NO Phone/Computer/Calculator allowed*** except that you will be allowed to use two single ***hand-written*** sheets 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 Final 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 the definitions of association, aggregation, composition and dependency, and give an example of each. Explain the similarities and differences between composition and aggregation.

(2) 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.

Inheritance

(3) 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.

(4) 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

(5) 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.

(6) 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.

(7) Know the usage of upcasting and downcasting.

Design Patterns

(8) 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.

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

(10) 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

(11) 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).

Interactive Techniques

(12) Know the CN1 Graphic Class and its methods. Know how the component's Repaint and Paint work. Drawing coordinates. Maintaining Graphic State. Implement Object selection. Review sample programs in Canvas.

Introduction to animation

(13) Be able to explain "Frame based animation"? Utilize the CN1's UITimer to schedule ticks to run drawing programs. "Self-animating" – Form consists of self-drawing objects. Compute animated coordination for a given current location. **Collision detection algorithms**: Detecting collisions (Bounding circle, Bounding rectangle), dealing with (responding to) collisions (Modify heading, Change appearance, etc). Know how to apply and use the ICollider, IDrawable, IMovable interfaces.

Introduction to Sound

(14) Know the popular sound APIs, Playing sounds in CN1. Create background sound that loops.

Transformation

(15) Explain what transformation is. Affine Transformations: Translation, Rotation, Scaling, Reflection. Matrix Representation of Transforms. Translate, rotate scale and reflect objects using matrices.

Rework the Transformation attendance quiz.

(16) Applications of Affine Transforms: CN1 Coordinate Systems and Transforms, Local Coordinate Systems, Display-mapping Transforms, Transformable Objects, Composite Transforms, Hierarchical Object Transforms, Dynamic Transforms.

Apply these techniques to build a new Hierarchical object such as a Windmill (i.e. with multiples instances in various sizes). Please refer to a demonstration in lecture.

(17) Viewing Transformation: World & Display Coordinate Systems; World-to-Display Mapping, World Window, and the Viewing Transformation (VTM); Zoom and Pan; Display-to-World Mapping; Clipping (Cohen-Sutherland algorithm).

Practice the clipping in class exercise per lecture material. Refer to zoom and pan demonstration programs showed in lecture.

Lines and curve

(19) Lines and curve: Rasterization, DDA & Bresenham Algorithms; Parametric Line and Curve Representation; Bezier Curves; Bezier curve drawing algorithms.

Practice the lines and curves practice in class exercise per lecture material.