CSSE 221 – Introduction to Software Development Honors
Final Exam Topics

The exam has a closed-book portion and an open-book portion.

Closed-book topics: You can use a single 8.5 inch by 11 inch "cheat sheet" (back and front, with whatever you want on it). (Alternatively, you can use 2 sheets, but fronts only.)

You may work with others to make your cheat sheet, but you will probably find it most useful if you do much or all of it yourself.

The following topics are the same as from Exam 1:

1. Given a class, identify the:
    a. ***Fields***
    b. ***Constructors***
    c. ***Methods***
    d. ***Type*** of variable *blah*
    e. ***Visibility*** of variable *blah*
    f. ***Return type*** of method *blah*
    g. Number of ***parameters*** of method *blah*, and the types of those parameters
2. What is the ***signature*** of a method?
3. What is the ***scope*** of a field? Of a parameter? Of a local variable that is not a parameter?
4. What is the ***lifetime*** of a field? Of a parameter? Of a local variable that is not a parameter?
5. Give four examples of ***primitive types***.
6. How do ***object-type variables*** differ from ***primitive-type variables***?
7. What keyword indicates that a ***method does not return a value***?
8. How does a ***static method*** differ from a non-static method?
9. How does a ***static field*** differ from a non-static field?
10. What does the ***final*** keyword mean when applied to a variable, as in the following?

```
private static final int DEFAULT_RADIUS = 25;
```

11. What does the ***final*** keyword mean when applied to a method, as in the following?

```
protected final double getWidthAndHeight()
```

12. What is the ***implicit parameter*** in the following? The ***explicit parameters***?

```
father.replace('a', '8');
```

13. In a class definition, what keyword refers to the ***implicit parameter***?

14. What does the keyword ***this*** mean, as in the following?

```
return this.widthAndHeight;
```

15. What does the keyword ***this*** mean, as in the following?

```
world.addBall(this);
```

16. What does a ***constructor*** do? How do you invoke a constructor?
17. Explain what space is allocated by the following:

```
Rectangle box = new Rectangle(5, 10, 20, 30);
```

18. What are the four ***visibility*** levels? For each, what restrictions does it impose?
19. Generally, what ***visibility*** should ***fields*** have?
20. Generally, what ***visibility*** should ***methods*** have?
21. Generally, what ***visibility*** should ***constructors*** have?
22. If class B ***implements interface X***, what does that imply?
23. What is the purpose of having ***interfaces***? Give a concrete example of an interface and explain why it is useful.
24. If class B ***extends class Y***, what does that imply?
25. What is the purpose of having ***inheritance***? Give a concrete example of inheritance and explain why it is useful.
26. What does ***foo overrides blah*** mean?
27. What does it mean for a method to be ***overloaded***?
28. What class is at the top of the ***inheritance hierarchy***?
29. Does Java allow ***multiple inheritance***?
30. In a constructor's definition, how does one refer to the ***superclass' constructor***?
31. In a constructor's definition, how does one refer to ***another constructor of the class***?
32. In a method's definition, how does one refer to the ***same-named method in the superclass***?
33. What does `this(...)` mean? `super(...)`? `super.foo(...)`?
34. What are Java's ***two dirty little secrets regarding constructors***? That is, Java supplies invisible code regarding constructors in which two situations?
35. What do the words ***subclass*** and ***superclass*** mean?
36. What does it mean for a variable in a subclass to ***shadow*** a variable in a superclass? What does it mean for a local variable to ***shadow*** a field?
37. Every object has a ***declared type***, and every non-null object has an ***actual type***. Give an example that shows the difference.
38. Why do we often declare objects using ***interface type*** instead of the object's actual type?
39. Give an example of ***polymorphism***.
40. What does the word ***cast*** mean?
41. What is the ***notation for doing a cast***?

42. What is the operator for determining whether a given object is an ***instance of*** a given class?
43. What ***annotation*** indicates that a method is a ***JUnit 4 test***?
44. Give at least two reasons why ***UML class diagrams*** are useful.
45. What is the UML notation for a class? Give an example.
46. What is the UML notation for ***is-a (extends)***?
47. What is the UML notation for ***is-a (implements)***?
48. What is the UML notation for ***has-a***?
49. What is a ***getter***? A ***setter***?
50. What does it mean for an object to be ***immutable***?
51. What is ***cohesion*** in class design? Do you want high cohesion or low cohesion in the classes that you design?
52. What is ***coupling*** in class design? Do you want high coupling or low coupling in the classes that you design?
53. What is a ***package***? Why are they useful?
54. What is ***version control***? Why is it useful?
55. What is ***developing by using documented stubs***? Why is it useful?
56. What is ***test-first development***? Why is it useful?
57. What is ***pair programming***? Why is it useful?
58. What is ***iterative enhancement***? Why is it useful?

New for the Final Exam:

59. What is the definition of ***big-Oh***? Why is it useful?
60. How do you order the following functions, and combinations thereof, from least to most, in terms of their asymptotic behavior?

    $\log(\log n)$ $\log n$ $\log^k n$ n^k k^n

    (Answer: in the order listed, for $k > 1$.)
61. What function of *n* equals the number of times it takes to get down to 1 if you start with *n*, divide it by 2, divide that result by 2, divide that result by 2, and so forth? (Answer: it takes ***log n*** such divisions.)
62. What is ***binary search***?
63. What is a ***recursive*** method? (Answer: a method that calls itself.)
64. Draw a picture of a linked list. Show how the picture changes if you add an element to or remove an element from the beginning of the list; the middle of the list; the end of the list.

Open-book topics: You should be able to do the following:

The following are the same as for Exam 1:

1. ***Construct*** an object.
2. ***Use an object***'s methods and/or fields.
3. Use ***assignment***.
4. Read, understand and use the ***API*** (Application Programming Interface) of a class that you have not seen before.
5. Explain the implications of object variables being ***references***.
6. Output to the console by using the ***System.out*** object.
7. Read from the console by using the ***Scanner*** class.
8. Use ***conditional*** statements.
9. Write *for* and *while* ***loops***.
10. Write ***nested loops***.
11. Process a loop using a ***sentinel value***. Use ***break*** and/or ***continue*** in a loop.
12. Use an ***array*** of a generic type: declare, fill, iterate through (new style and old style), get/set elements (by iterating and/or by directly accessing elements).
13. Use an ***ArrayList*** of a generic type: declare, fill, iterate through (new style and old style), get/set elements (by iterating and/or by directly accessing elements).
14. ***Copy an array*** or ArrayList, by using a loop or by using the ***Arrays*** or ***Collections*** class.
15. Use the ***counting***, ***summing***, **min/max** and ***histogram*** looping patterns.
16. ***Implement a class***, including implementing an ***interface*** and/or ***extending*** a class.

New for the Final Exam:

17. Read a ***UML class diagram*** and implement it.
18. Implement a GUI using ***frames***, ***panels*** and ***buttons***.
19. Implement Listener's per ***event-driven programming***.
20. Implement ***threads***.
21. Use a ***try/catch clause*** to catch an ***Exception***.
22. Do ***big-Oh analysis*** at the level tested in Exam 2.
23. Write simple ***recursive*** functions.
24. Optimize a recursive function by using a ***memory table***.
25. Do the following ***in C***:
    a. Write and call ***functions***.
    b. Write a ***main*** function.

c. Use ***printf*** and ***scanf***.

d. ***Return a value*** from a function in the ordinary way (using a ***return*** statement).

e. ***Return a value*** from a function ***using a parameter*** (using & and * appropriately).

f. ***Declare a dynamic array***.

g. ***Allocate memory for a dynamic array***.

h. ***Use a dynamic array***.

i. Declare a ***structure*** with ***fields***, using a ***typedef***.

j. ***Allocate memory for a structure instance***.

k. ***Use a pointer to a structure instance***, including referencing the fields of the pointer to the structure instance.

l. ***Combine*** the above ideas.

26. Implement a simple ***linked list***, including specifying the structure/class used and methods for adding or removing elements to/from the list.

On-the-computer topics: You should be able to do the following:

1. ***Implement a UML class diagram***, including:

   a. Choosing and implementing the necessary fields

   b. Choosing and implementing the necessary constructors

   c. Choosing and implementing the necessary methods

   d. Implementing the relationships shown on the UML class diagram, along with other relationships needed to implement the ones shown on the diagram.

   e. Implement a GUI including frames, panels and buttons.

   f. Implement ActionListener's and MouseListener's and MouseMotionListener's.

   g. Using items from the Java API that are new to you.

The on-the-computer problem on the final exam will be similar to the one on Exam 2.