Unit 6 Test Review

1. What does OOP stand for?	Object Oriented Programming
2. What does ADT stand for?	Abstract Data Type
3. What does LIFO stand for?	Last In First Out
4. What does FIFO stand for?	First In First Out
5. Name 6 Stack functions	Pop, push, peek, size, isFull, isEmpty,
6. Name 6 Queue functions	Dequeue, enqueue, peek, size, isFull, isEmpty
7. Name 4 functions shared by queues and stacks.	peek, size, isFull, isEmpty
8. Speed of all Stack Functions?	O(1)
9. Speed of all Queue Functions?	O(1)
10. Speed of binary search?	O(log n)
11. Speed of quicksort?	O(n log n)
12. Speed of merge?	O(n)
13. Speed of mergesort?	O(n log n)
14. Return type of toString?	String
15. Return type of mutators?	void
16. Return type of compareTo?	Int
17. Return type of equals?	Boolean
18. What is the name of variables inside an Object's class?	Instance variables
19. A keyword restricting variables access to inside a class	Private
20. A method that is automatically called when the variable's name is printed	ToString
21. A method that checks if two objects are the same	equals
22. A method that returns 1, 0, -1	CompareTo
23. The opposite keyword to private.	Public
24. A template for an object or a type. Also contains a java programs.	Class
25. An object method type that returns the values of instance variables	Accessor

26. An object method type that sets up dynamic memory.	Constructor
27. An object method type for methods that don't easily fit the other groups.	Facilitator
28. An object method type that sets aside RAM for the instance variables and initializes them.	Constructor
29. An object method that converts the object to a String and returns it.	toString
30. An object method type that changes the values of instance variables	Mutator
31. A non-primitive type	Object
32. Data and methods that act upon data.	Object
33. Programmers can think of a problem at a higher more removed level.	Abstraction
34. Keeping an object's code self-contained and independent of other code.	Encapsulation
35. Removing direct access to instance variables from other programmers.	Information Hiding
36. Keeps the code stable.	Information Hiding
37. Keeps the code moveable.	Encapsulation
38. Keeps the code useable.	Abstraction
39. Signs of Information Hiding	1 Private Instance variables 2 Mutators to change instance variables 3 Access to view instance variables
40. Signs of Abstraction	1 Easily represented by a physical model or diagram 2 Code can be used without reading more than signatures
41. Signs of Encapsulation	1 Easy to transfer and share code 2 Easy to divide pieces among team

42. Three uses of class name	1 File name to save as 2 Constructor name 3 Class name on class line
43. Three strange things about a constructor	1 No return type on method signature, it returns itself 2 Same name as class 3 Called using new
44. An ADT used to code an undo button	Stack
45. An ADT used to code a line of people at a grocery store	Queue
46. An ADT used for FIFO	Queue
47. An ADT used for LIFO	Stack
48. An ADT used to store recursive method calls	Stack
49. An ADT used to store a back button.	Stack
50. An ADT used to model cars on a road	Queue
51. An ADT used to model loading an airplane.	Queue
52. An ADT to hold print jobs for a printer	Queue
53. An ADT to hold changes to a bank account	Queue
54. An ADT that could model a pile of books well.	Stack
55. An ADT where you add to the front and remove from the front	Stack
56. An ADT where you add to the back and remove them the front.	Queue
57. An ADT where you add to the front and remove from the back	Queue
58. An ADT where you add to the back and remove from the back.	Stack
59. An ADT where you add to the top and remove from the top.	Stack

60. Stack tradeoff	Positive: All functions O(1), very fast Negative: Only LIFO functions. Speed is achieved through limiting functionality
61. Queue tradeoff	Positive: All functions O(1), very fast Negative: Only FIFO functions. Speed is achieved through limiting functionality
62. How do you switch a Queue to a Queue of a Frog class?	Switch all Objects to Frog
63. Write code that declares 2 stacks	Stack s = new Stack(); Stack s2 = new Stack();
64. Write code that declares a queue	Queue q = new Queue();
65. Write code that prints out a stack	while(! s.isEmpty()) System.out.println(s.pop());

66. Write code that switches the top two elements of a stack	Int holder = s.pop();
	Int holder2 = s.pop();
	s.push(holder);
	s.push(holder2);

67. Write code that empties (and reverses) one stack into another	while(!.s.isEmpty()) S2.push(s.pop());
68. Write code that empties a queue (and reverses) a queue into a stack	while (!q.isEmpty()) s.push(q.dequeue());
69. How do you see the top element of a stack?	System.out.println(s.peek());
70. What is half the size of a stack?	s.size/2