Full Name: ______________________________ NetID: ______________________

Midterm, 02/09/2017

OAKLAND UNIVERSITY, School of Engineering and Computer Science
Object Oriented Computing II
CSE 231 002/CSE 506 002

Please write and/or mark your answers clearly and neatly; answers that are not readable are not considered. Please answer in appropriate details if needed and you may get partial points for the intermediate steps even if a final answer is not correct.

Each question is of 1 point, unless otherwise explicitly specified.

Chapter 1 (20 points):

1. The software life-cycle model where one carefully follows a predetermined succession of stages, one after the other, is called the _______________ model.

2. UML is an acronym for ____

3. Which kind of variable holds a value that cannot be changed?

   (A) instance variable
   (B) final variable
   (C) private variable
   (D) class variable
   (E) protected variable

4. List the four types of access modifiers provided by Java and describe what each controls access to. (4 points)

5. If the Java compiler cannot find a method defined in an object's class definition it will look in the definition of the class's _____________

   (A) superclass
   (B) subclass

6. Java supports double inheritance, i.e., a class can have more than one ___________

   (A) superclass
   (B) subclass

7. The Java reserved word _________________ indicates inheritance, i.e., we code *class subclass* ______________ *superclass*.

8. To access the contents of a package from within a program, you must ___________ the package into your program (answer with a Java reserved word).

9. Which structure is "first in, first out".

   (A) Array
   (B) Linked List
   (C) Stack
   (D) Queue
   (E) Tree

10. In Java, method arguments are passed by

    (A) value
    (B) reference
    (C) creating a copy of the argument and passing it
    (D) using an array
    (E) none of these

11. $3N^3 + 5N^2 + 27N + 17$ is $O(3N^3)$, True or False _________________

12. What is the order of magnitude of the following function, using Big-O notation: $3N^4 + 17N^2$? _________________

13. Describe the order of magnitude of the following code section using Big(O) notation. _________________

```
k = 0;
for (i = 0; i < N/2; i++)
 for (j = (2 * N); j > 0; j--)
  k = k + 1;
```

14. Calculate the number of iterations in the following double-nested loop in best case complexity, worst case complexity and the average case complexity, and then describe the order of magnitude of the code using Big(O) notation. (4 points)

```
k = 0;
for (i = 0; i < N % 32; i++)
 for (j = (2 * N); j > 0; j--)
  k = k + 1;
```

Chapter 2 (20 points):

15. A data type whose properties (domain and operations) are specified independently of any particular implementation is a(n) ___________________________ (Full name, not acronym).

16. What are the three perspectives, or levels, with which we deal with ADTs? On which level do we just need to know how to use the ADT? (5 points)

17. Assumptions that must be true upon entry into a method for it to work correctly are _________________________ .

18. What kinds of constructs can be declared in a Java interface? (2 points)

19. Put the following steps we follow when implementing the StringLog ADT in the order of our development (the normal steps we will do to implement any ADT): (4 points)

    (A) Implement StringLogInterface using Array
    (B) Create UseStringLog application
    (C) Define StringLogInterface
    (D) Implement StringLogInterface using LinkedList
    (E) Create test program and test cases for our implementation

20. For the *ArrayStringLog* class, assuming a list size of N, what is the Big-O complexity for contains method? (2 points)

21. Testing a method by itself is called

    (A) Unit testing
    (B) Inspection
    (C) Desk checking
    (D) Black box testing
    (E) Walkthrough

22. For *contains* methods of *LinkedStringLog* implementation in the following code, is the implementation correct? If not, what is wrong with the code and modify it to make it correct. (4 point)

```
public class LLStringNode {

        private String info;
        private LLStringNode link;
         …………
}
public class LinkedStringLog {

        protected LLStringNode log;.

        public boolean contains(String element) {
                // Returns true if element is in this StringLog, otherwise false.
                LLStringNode node;
                node = log;
                boolean found = false;

                while (node != null) {
                         if (element.equalsIgnoreCase(node.getInfo()))
                                 found = true;
                        else node = node.getLink();
                }

                return found;
        }
```

```
            …………..
    }
```

Chapter 3 (20 points):

23. LIFO stands for ____________________; it is the property of ______ data structure. (2 points)

24. What are the three major steps for Java to handle exceptions? In which steps an exception object is created? (6 points)

25. Show what is written by the following segment of code, given that item1, item2, and item3 are int variables, and stack is an object that fits our abstract description of a stack. Assume that you can store and retrieve variables of type int on stack. (6 points)

```
item1 = 1;
item2 = 0;
item3 = 4;
stack.push(item2);
stack.push(item1);
stack.push(item1 + item3);
item2 = stack.top();
stack.push (item3*item3);
stack.push(item2);
stack.push(3);
item1 = stack.top();
stack.pop();
System.out.println(item1 + " " + item2 + " " + item3);
while (!stack.isEmpty())
{
 item1 = stack.top();
 stack.pop();
 System.out.println(item1);
}
```

Answer:

26. The array-based stack data structure is defined as follows. Complete the implementation of the *push* method: (6 points)

```
package ch03.stacks;

public class ArrayStack<T> implements BoundedStackInterface<T>
{
  protected final int DEFCAP = 100; // default capacity
  protected T[] stack;              // holds stack elements
  protected int topIndex = -1;      // index of top element in stack

  public ArrayStack()
  {
    stack = (T[]) new Object[DEFCAP];
  }

  public ArrayStack(int maxSize)
  {
    stack = (T[]) new Object[maxSize];
  }

public void push(T element)
// Throws StackOverflowException if this stack is full,
// otherwise places element at the top of this stack.
{
  if (!isFull())
  {
    // complete the code here (about 2 statements)




















  } else throw new StackOverflowException("Push attempted on a full
stack.");
}
```

Chapter 4 Recursion (20 points):

27. What the three questions we need to answer to verify a recursive algorithms or functions? (5 points). Given the following function, explain how the three questions are used to verify the recursion of this function. (5 points)

```
int factorial (int n)
{
if (n > 0) return (n * factorial (n – 1));
else if (n == 0) return 1;
}
```

28. What is action record, explains using figures how action record is used to handle function call. (10 points)

Chapter 5 (20 points):

29. True or False? The first element to be stored in a queue is also the first element removed from the queue.

30. The *QueueUnderflowException* is potentially thrown by the following queue method(s).

A. enqueue only
B. dequeue only
C. isEmpty
D. the constructors
E. enqueue and dequeue

31. The *QueueOverFlowException* is potentially thrown by the following queue method(s).

A. enqueue only
B. dequeue only
C. isEmpty
D. the constructors
E. enqueue and dequeue

32. What is the difference between the *enqueue* method defined in our *UnboundedQueueInterface* and the *enqueue* method defined in our *BoundedQueueInterface*? (4 points)

33. Show what will be written by running following segment of code, given that item1, item2, and item3 are int variables, and queue is an object that fits our abstract description of a queue. Assume that you can store and retrieve variables of type int on queue. (7 points)

```
item1 = 6;
item2 = 3;
item3 = 4;
queue.enqueue(item2);
queue.enqueue(item1);
queue.enqueue(item3);
queue.enqueue(item1 + item3);
item3 = queue.dequeue();
queue.enqueue (item3*item3);
queue.enqueue(7);
item2 = queue.dequeue();
queue.enqueue(item2);
item1 = queue.dequeue();
System.out.println(item1 + " " + item2 + " " + item3);
while (!queue.isEmpty())
{
 item1 = queue.dequeue();
 System.out.println(item1);
}
```

34. In the following enqueue method for ArrayUnbndQueue class, what is the purpose of the statement in bold font (rear = (rear + 1) % queue.length)? (6 points)

```
public void enqueue(T element)
// Adds element to the rear of this queue.
{
  if (numElements == queue.length)
    enlarge();
  rear = (rear + 1) % queue.length;
  queue[rear] = element;
  numElements = numElements + 1;
}
```

-------------------------------- The end of the Test --

Full Name: ______________________________ NetID: ______________________

Scratch paper