Review Final Exam Fall 2015

Course Outcomes

By the end of the course students will be able to:

1. Understand the concept of an Abstract Data Type (ADT).
2. Understand and be able to implement the list ADT.
3. Understand and be able to implement the stack ADT.
4. Understand and be able to implement the queue ADT.
5. Understand and be able to implement the priority queue ADT.
6. Be able to determine the time complexity of simple algorithms.
7. Understand and be able to implement several standard sorting techniques.
8. Understand and be able to implement linear and binary search.
9. Understand and be able to use recursion.
10. Have an improved understanding of the Java programming language.

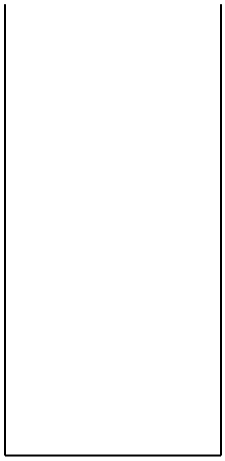
Final Exam Review Questions

1. Covering Outcome #1, #2 a **Linked List** question
2. Covering Outcome #1, and #3, a **Stack** question.
3. Covering Outcome #1, #4 and #5 **Priority Queue** question.
4. Covering Outcome #6, #8, a **Time Complexity** question.
5. Covering Outcome #7, a **Sorting** question.
6. Covering Outcome #1, #2, and #9 a **Linked List, recursion** question.
7. Covering Outcome #10, a **Java GUI,** question.
8. Covering Outcome #1, #10 a **Reference** question

Example Stack Question:

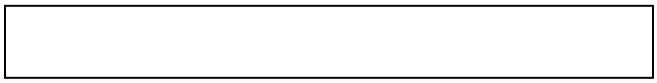
What does the initially empty stack **stack1** look like after the following sequence of operations.

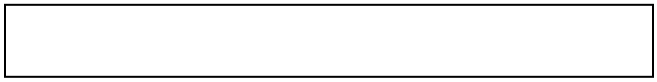
```
stack1.push(23);
stack1.push(34);
stack1.push(35);
stackTop = stack1.pop();
stackTop = stack1.pop() + stack1.peek();
stack1.push(stackTop);
stack1.push(75);
stackTop = stack.pop() + stack.pop();
stack1.push(stackTop);
stack1.push(2);
```



stack1

What do the initial empty queues look like after the following sequences of operations:





```
queue1.enqueue(1)

queue1.enqueue(2)

queue2.enqueue(3)

queue2.enqueue(4)

queueFirst = queue1.dequeue()

queueFirst = queue2.peek()

if (queue1.isEmpty())

    queue1.enqueue(queueFirst)

queue1.enqueue(5)

queueFirst = queue2.dequeue()

 queue2.enqueue(6)
```

A. Name a data structure that has a worst case **search** algorithm that has a time complexity of O(log n)

B. Name a data structure that has a worst case search algorithm that has a time complexity of O(n)

C. Name an sorting technique that has a worst case time complexity of O(n log n)

D. Which has a better worst case time complexity, sorting an array of 1000 elements with insertition sort, or searching an array of size 2000 elements for an integer ?

E. What is the worst-case time for binary search finding a single item in a sorted array?

- A. Constant time
- B. Logarithmic time
- C. Linear time
- D. Quadratic time

F. What additional requirement is placed on an array, so that binary search may be used to locate an entry?

G. How many possible values can num have in the following statement :

```
int num = var%9;
```

1. Name three keywords for visibility in a class. Explain how each impacts on a class method, or data field.

1. **Sorting:** show the steps of one of the following sort methods, Insertion sort, Merge Sort, Bubble Sort, Selection Sort, or Quick Sort. Know the time complexity of each, here' s and example of what the set up will look like:

Given the array below give the array at each step of a selection sort, and draw arrows to explain what's happening. Once a int is in it's correct spot put an X above it so you we can see it is in it's final resting place. Make sure I understand what you are doing at each step. Put the temp variable out to the right.

5	7	6	3	2	8	1

1. **Recursion, follow the code like in the following examples:**

```
public class TryThis
{
  public static void main(String [] args)
  {
    System.out.println(calling(1, 12, 10));
  }

  public static int calling(int first, int last, int n)
  {

      int returnValue;
      System.out.println("Enter: first=" +  first +" last = "+last);
      int mid = (first+last)/2;
      if(mid*mid<=n) returnValue=mid;
      else returnValue = calling(first, mid-1, n);
      System.out.println("Leave: first = " +first + " last = "+last +" mid ="+ mid);
      return returnValue;
  }
}
```

Write a recursive method that takes as parameters a String **s** and an integer **i** and returns a String that has s repeated i times. For example, if the given string is "Go Bobcats! " and the integer is 3 then the return value would be "Go Bobcats! Go Bobcats! Go Bobcats! ". (Note that if the integer is 0, then the empty string "" should be returned.)

Here is a sample main:

```
public static void main(String [] args)
  {
     String answer = recur("Go Bobcats! ",  3);
     System.out.println(answer);

   }
```

Output: Go Bobcats! Go Bobcats! Go Bobcats!

Your code:

Work through the following code and give a solution, also explain what it is doing, Give the variable values through all the steps.

```
public class Recur
{
        public static void main(String [] args)
  {
     int [] a = {6, 5, 4, 4, 5, 8, 9, 6, 4};
     int n = 9;
     int val = 4;
     int ans = subfun(a, n, val);
     System.out.println(ans);

  }
  public static int subfun(int x[], int n, int val)
  {
    int c;
    if (n <= 0)
      return 0;
    else
    {
       if(x[n-1] == val)
          c = 1;
       else
          c = 0;

       return subfun(x, n-1, val) + c;
     }//end of else
  } //end of subfun

} //end of class
```

GUI stuff: Know the mouse, buttons and layouts of Swing for this question.

(a) Explain what the code below does from the standpoint of a user.

```
public class ExamPanel extends JPanel implements MouseMotionListener
{
   ExamPanel()
   {
      setBackground(Color.green);
      addMouseMotionListener(this);
   }

   public void mouseMoved (MouseEvent evt)
   {
      int x = evt.getX();
      int y = evt.getY();
      if ((x < getWidth() / 2) && (y < getHeight() / 2))  // getWidth() returns the pixel width of ExamPanel
      {                                                   // getHeight() returns the pixel height of ExamPanel
         setBackground(Color.black);
      }
      else if ((x > getWidth() / 2) && (y > getHeight() / 2))
      {
         setBackground(Color.white);
      }
      else
      {
         setBackground(Color.red);
      }
   }

   public void mouseDragged (MouseEvent evt)
   {
   }
}
```

Priority Queue.

Implement a priority queue using a Heap (that looks like a binary tree) – Insert the following items into the heap, smaller numbers are higher priority and get removed first (show the heap after each insertion and movement of the number).

Insert in order: 12, 24, 30, 10, 15, 8,

Remove the item with the lowest number from the queue, redraw the heap after each step of removal and reconfiguring the queue so the priorities hold true.