Week 8 Review Stacks and Queues

**Programming with Java - Stacks**

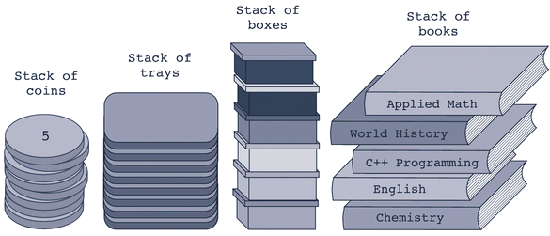
**(1)** Which element of a stack has been in the stack the longest?

**(a) The element at the bottom**

(b) The element at the top

(c) The element in the middle

(d) All elements in a stack have been there the same amount of time



**(2)** Which operation would you use to return the Applied Math book from the stack above?

(a) top (b) push **(c) pop** (d) return (e) free

**(3)** What would the operation **isEmptyStack()** return if applied to the stack above?

(a) 0 (b) true (c) 5 **(d) false** (e) 1

**(4)** What would the pop operation do if applied to the stack above?

(a) Return the Applied Math book **(b) Remove the Applied Math book**

(c) Remove the Chemistry book (d) Nothing

**(5)** If you applied the operation pop to the stack above, which book would be left on top of the stack?

(a) Applied Math (b) Chemistry

**(c) World History** (d) English

**(6) public void operationX(DataElement newItem) throws StackOverflowException {**

**if(isFullStack())**

**throw new StackOverflowException();**

**list[stackTop] = newItem;**

**stackTop++;**

**}**

Which stack operation does **operationX** above define?

(a) top **(b) push** (c) pop (d) initializeStack

**(7) public DataElement operationY() throws StackUnderflowException{**

**if(isEmptyStack())**

**throw new StackUnderflowException();**

**DataElement temp = list[stackTop - 1].getCopy();**

**return temp;**

**}//end top**

Which stack operation is defined by **operationY** above?

(a) top (b) push (c) isEmptyStack **(d) pop**

**(8)** In an array - based stack, which of the following operations has a

time - complexity of *O* ( *n* ) ?

(a) initializeStack **(b) deleteStack**

(c) copyStack (d) constructor

**public class StackProgram {**

**public static void main(String[] args) {**

**StackClass intStack = new StackClass(50);**

**StackClass tempStack = new StackClass();**

**try {**

**intStack.push(new IntElement(23));**

**intStack.push(new IntElement(45));**

**intStack.push(new IntElement(38));**

**}**

**catch(StackOverflowException sofe) {**

**System.*out*.println(sofe.toString());**

**System.*exit*(0);**

**}**

**tempStack.copyStack(intStack);**

**System.*out*.print("tempStack elements: ");**

**while(!tempStack.isEmptyStack()) {**

**System.*out*.print(tempStack.top() + " "); // output 1**

**tempStack.pop();**

**}**

**System.*out*.println();**

**System.*out*.println("The top element of intStack: "**

**+ intStack.top()); // output 2**

**}**

**}**

**(9)** What is **output 1** above?

(a) 23 38 45 **(b) 38 45 23**

(c) 38 23 45 (d) 23 38 43

**(10)** What is **output 2** above?

(a) 23 (b) 45 **(c) 38** (d) 50 (e) None of these

**Programming with Java - Queues**

**(11)** Whenever an application is modeled on a FIFO structure, \_\_\_\_\_\_\_\_\_\_ are

used.

(a) lists (b) stacks **(c) queues** (d) arrays

**(12)** The method deleteQueue does which of the following?

(a) uses one queue to delete another

(b) removes the back element from the queue

**(c) removes the front element from the queue**

(d) removes all elements from the queue leaving an empty queue

**(13)** The method addQueue does which of the following?

(a) adds all the contents from one queue to another

(b) appends one queue to the back of another

(c) adds a new element to the front of the queue

**(d) adds a new element to the rear of the queue**