Computer Science Fundamentals II  
Saturday, October 26, 2019

Midterm Exam

CS1027A  
University of Western Ontario

Last Name: ________________________________  
First Name: ________________________________  
Student Number: ____________________________  
Section Number (1 - Sarlo, 2 - Bloch-Hansen): _____

<table>
<thead>
<tr>
<th>PART</th>
<th>Marks</th>
</tr>
</thead>
<tbody>
<tr>
<td>I</td>
<td>25</td>
</tr>
<tr>
<td>II</td>
<td>15</td>
</tr>
<tr>
<td>III</td>
<td>5</td>
</tr>
<tr>
<td>IV</td>
<td>5</td>
</tr>
<tr>
<td>V</td>
<td>5</td>
</tr>
<tr>
<td>VI</td>
<td>8</td>
</tr>
<tr>
<td>BONUS</td>
<td>8</td>
</tr>
<tr>
<td>Total</td>
<td>63</td>
</tr>
</tbody>
</table>

Instructions

• Fill in your name, student number, and section.

• Verify that your exam paper has 18 pages and 39 questions.

• The exam is 2 hours long and is scored out of 63 marks. In each of Parts III, IV, and V, only answer one question. There is a bonus question worth 8 marks, which can bring your score above perfect.

• PART I: Multiple choice questions. For each question circle only one answer.

• PART II: Short answer questions; answer each question only in the space provided.

• PART III: Complete only one stack question.

• PART IV: Complete only one queue question.

• PART V: Complete only one linked list question.

• PART VI: Complete only one algorithm design question.

• BONUS: Complete the other algorithm design question.

• When you are done, raise your hand and one of the TA’s will collect your exam.
1 (25 marks) Part I. Multiple Choice Questions

Each multiple choice question is worth 1 point; circle only one answer.

1. Every class you create, whether or not it contains extends, is a sub-class of one or more parent classes.
   (A) True  (B) False

2. A Java interface specifies every method that must be implemented and provides details about how each method should be implemented.
   (A) True  (B) False

3. You MUST include a toString() method when making a new class for it to compile.
   (A) True  (B) False

4. The following two println() method calls invoke the exact same method signature:

   1 int count = 5;
   2 System.out.println("The total is: ");
   3 System.out.println(count);

   (A) True  (B) False

5. Polymorphism can be achieved using interfaces instead of inheritance.
   (A) True  (B) False

6. As described in lecture, a person can drive/use a car without knowing how a car works. This is an example of encapsulation.
   (A) True  (B) False

7. Which of the following variable types should be used for a class’ member variables so that they are accessible by itself and its child classes but not any other classes?
   (A) private  (B) public  (C) protected  (D) static  (E) final

8. Which of the following is not one of the core Object-Oriented Programming fundamentals/principles?
   (A) Encapsulation  (B) Modularity  (C) Inheritance  (D) Information Hiding

9. A method name can be used for multiple methods as long as:
   (A) The number of parameters is distinct
   (B) The types of the parameters are distinct
   (C) The order of the types of the parameters is distinct
   (D) The return type is distinct
   (E) At least one of A, B, or C is true

10. Method overriding is when:
    (A) The Java compiler chooses a better method for you than what you tried to invoke
    (B) Two methods have the same name but different method signatures
(C) A child class defines a method with the same method signature as a parent method
(D) An exception is generated and the method is over-written by the catch clause

![Inheritance Hierarchy](image)

Figure 1

11. Consider the inheritance hierarchy shown in Figure [1] instance variables for each class are included. How many instance variables does a CityBus have?
(A) 0  (B) 1  (C) 2  (D) 3  (E) 4

12. Consider the following code fragment:

```java
1  public class Foo {
2      public static void divide(int x, int y) {
3          try {
4              System.out.println(y / x);
5          } catch (NumberFormatException e) {
6              System.out.print("A");
7          } finally {
8              System.out.print("B");
9          }
10             System.out.print("C");
11         }
12     public static void main(String[] args) {
13         int x = 0;
14         int y = 5;
15         try {
16             divide(x, y);
17         }
18         catch (Exception e) {
19             System.out.print("D");
20         }
21         catch (IOException e) {
22             System.out.print("E");
23         }
24     }
25 }
```

3
What does the code fragment above output when it is executed?

(A) "A"  (B) "AB"  (C) "B"  (D) "ABC"  (E) "BC"  (F) "AC"  (G) "C"
(H) "D"  (I) "BD"  (J) 5  (K) 0  (L) "BCD"  (M) "ABCD"  (O) "AD"

13. Consider a stack \( s \) and a queue \( q \) storing integer values. Consider now the following code fragment:

1. for (int i = 0; i < 6; i++) s.push(i);
2. for (int i = 0; i < 3; i++) q.enqueue(s.pop());
3. for (int i = 0; i < 2; i++) s.push(q.dequeue());
4. System.out.println(s.pop());

What does the code fragment above output when it is executed?

(A) 2  (B) 3  (C) 4  (D) 5

14. Consider the following code fragment:

1. public class BankAccount {
2.   public BankAccount() { }
3.   public void m() { System.out.println("BankAccount"); }
4. }
5. public class SavingsAccount extends BankAccount {
6.   public SavingsAccount() { }
7.   public void m() { System.out.println("SavingsAccount"); }
8. }

Consider now the following code fragment:

1. BankAccount bacc = new SavingsAccount();
2. bacc.m();

What does this code fragment print when it is executed?

(A) "BankAccount"  (B) "SavingsAccount"
(C) There is a compilation error  (D) There is a runtime exception

15. Using the classes defined in the previous question, consider now the following code fragment:

1. SavingsAccount sacc = new BankAccount();
2. sacc.m();

Which line has a compilation error in the above code fragment?

(A) Line 1  (B) Line 2  (C) Both of the above  (D) None of the above
16. Consider the following code fragment:

```java
public class Foo {
    private int x;
    public Foo(int x) {
        this.x = x;
    }
    public static void main(String[] args) {
        int x;
        for (int i = 0; i < 10; i++) {
            x = i;
        }
        Foo foo = new Foo(5);
    }
}
```

What is the scope of the variable ‘x’ used on line 9?
(A) Class  (B) Constructor  (C) Main  (D) For loop

17. Consider the following code fragment:

```java
public class TestBankAccounts {
    public static void main(String[] args) {
        BankAccount bacc = new BankAccount(100);
        bacc = new SavingsAccount(100);
        bacc = new CheckingAccount(100);
        double x = Math.random();
        if (x < 0.5)
            bacc = new SavingsAccount(500);
        else
            bacc = new CheckingAccount(500);
        System.out.println(bacc.toString());
    }
}
```

Lines 6-11 are an example of:
(A) Abstraction  (B) Dynamic binding  (C) Casting  (E) None of the above
18. Consider the following code fragment:

```java
int len = 4; sum = 0; i = 4;
int[] arr = new int[len];
try {
    while (i >= 0) {
        i = i - 1;
        arr[i] = i;
        sum = sum + arr[i];
    }
} catch (ArrayIndexOutOfBoundsException e) { sum = sum + 1; }
catch (NullPointerException e) { sum = sum - 1; }
catch (Exception e) { sum = sum * 2; }
```

What value does sum have at the end of the execution of the above code?

(A) 5 (B) 6 (C) 7 (D) 12

19. Consider the following code fragment:

```java
public class Main {
    public void foo(int x) { x = -1; }
    public static void main(String[] args) {
        Person[] p = new Person[2];
        int x = 0;
        foo(x);
        for(int i = x; i < p.length; i++)
            p[i].setLastName("Andrew");
        for (int i = x; i < p.length; i++)
            System.out.print(p[i].getLastName());
    }
}
```

What is the output produced by the above code fragment?

(A) NullPointerException (B) ArrayIndexOutOfBoundsException (C) ”AndrewAndrew” (D) Nothing. There are compilation errors.
20. Consider the stack shown in Figure 2. How many elements are in the stack?
   (A) 0  (B) 1  (C) 2  (D) 3  (E) 4

21. Consider the stack shown in Figure 2. What would happen/return if we called s.pop();?
   (A) 2  (B) 1  (C) ”Banana”  (D) ”Apple”  (E) NullPointerException

22. Consider the stack shown in Figure 2. What would happen/return if first ”Carrot”
    was pushed to the stack and then ”Orange” was pushed to the stack?
   (A) NullPointerException  (B) expandCapacity() would be called
   (C) The stack would have 4 elements  (D) top would point to ”Carrot”

23. Consider the following line of code:

   ```java
   1   rear = (rear + 1) % queue.length;
   ```

   What is the % operator called?
   (A) Division  (B) Fraction  (C) Percentage  (D) Modulus

24. What is the importance of the % operation in the context of a CircularArrayQueue?
   (A) Calculates how full the array is as a fraction
   (B) Calculates how full the array is as a percentage
   (C) Helps with the pop() method
   (D) Allows queue[queue.length-1] to precede queue[0]

25. The QueueADT can be implemented using:
   (A) Linked List  (B) Array  (C) Circular Array  (D) All of the above
2 (15 marks) Part II. Short Answer Questions

26. (2 marks) Explain what putting \(< T >\) beside a class name is called and what it does.
This is called generics/generic type.

It allows us to define a class and its methods based on a generic type rather than an actual type. E.g., the class is implemented without know what type of objects it will be receiving as parameters and returning from its methods.

27. (2 marks) List and explain the three main stack operations. Where in the collection does the activity happen?
   push(): add something to the top of the stack.
   pop(): remove something from the top of the stack.
   peek(): look at the top of the stack (but don’t remove).

All of the activity in a stack happens at the top.

28. (2 marks) List and explain the three main queue operations. Where in the collection does the activity happen?
   enqueue(): add something to the rear of the queue.
   dequeue(): remove something from the front of the queue.
   first(): look at the front of the queue (but don’t remove).

All of the activity in a queue happens at either the front/rear.

29. (2 marks) Explain why queues require a pointer to the front and a pointer to the rear. Also, explain why stacks do not have the same requirement.

Queues require a pointer to the front and rear because enqueue() adds to the rear but dequeue() removes from the front.

Stacks do not require a pointer to the top and the bottom because all of the activity occurs at the top of the stack.

30. (2 marks) Compare and contrast the array based implementation of a stack and the linked list implementation of a stack.
(Not an exhaustive list):
Both implementation will produce the same results.
The underlying data structure of the array stack is an array.
The underlying data structure of the linked stack is a linked list.
Array based implementation requires an expandCapacity() method.
Linked list implementation can grow and shrink dynamically.
Array based implementation must use contiguous memory
Linked list implementation does not use contiguous memory.
31. (5 marks) Create the following class for Flight.java.

The Flight class encapsulates an airplane flight between two locations using the private variables:

- flightNo: the number of the flight (of type int),
- source: the departure location of the flight (of type String), and
- destination: the arrival location of the flight (of type String).

and the methods:

- A constructor method, Flight(), with 3 parameters: the flight number, departure location, and arrival location.
- A getter method, getFlightNo(), that returns the flight number.
- A method, toString(), that returns a string containing the values of the object’s private variables.

```java
public class Flight {
    private int flightNo;
    private String source;
    private String destination;

    public Flight(int flightNo, String source, String destination) {
        this.flightNo = flightNo;
        this.source = source;
        this.destination = destination;
    }

    public int getFlightNo() {
        return flightNo;
    }

    public String toString() {
        String str = flightNo + " " + source + " " + destination;
        return str;
    }
}
```
3 (5 marks) Part III. Stack Question
I am answering: Q32 OR Q33 (Circle One)

32. (4 marks) Evaluate the post fix expression by drawing in Figure 3 and showing the contents of the stack after each input.

Evaluation of: 2 5 3 + * 6 -2 + -

(1 mark) Would the answer have been different if the stack was implemented using a linked list instead of an array?

No.
33. Consider a stack implemented using a singly linked list where `top` and `count` are instance variables pointing to the first node of the list and giving the number of data items in the stack, respectively. Consider now the following implementation of the push operation:

```java
1  private void push (T newValue) {
2     LinearNode<T> newNode = new LinearNode<T>(newValue);
3     top = newNode;
4     newNode.setNext(top);
5     ++count;
6  }
```

Give an example showing that the code does not produce the correct result.

(2 marks) Draw the initial stack in your example (show top and count):

```
Figure 4
```

(2 marks) Draw the stack after performing `push(x)` (show top and count):

```
Figure 5
```

(1 mark) How can you fix the above method?

Swap lines 3 and 4.

Swap lines 3 and 4.
4  (5 marks) Part IV. Queue Question
I am answering: Q34  OR  Q35 (Circle One)

34. Consider the circular array queue in Figure 6

(cq)

front

queue

F
G
H
D
E

0 1 2 3 4

rear

3

Figure 6

(3 marks) Fill in the queue in Figure 7 after cq.enqueue("I"); is called. Include the value of front, rear, and all of the queue elements.

(cq)

front

queue

D
E
F
G
H
I

0 1 2 3 4 5 6 7 8 9

rear

6

Figure 7

(1 mark) How many elements are in this new queue?
6.

(1 mark) In a circular queue, index 0 precedes index 1. In the queue shown in Figure 7, which index precedes index 0?
9.
35. (5 marks) Use the repeating key \(\{1, 4, 2\}\) to encode the string "JAVA IS FUN". Fill in the blanks in Figure 8 to show how many times each letter is shifted and what the contents of the queue are after each letter.

\[
\begin{align*}
J + 1 & = K \\
A + 4 & = E \\
V + 2 & = X \\
A + 1 & = B \\
I + 4 & = M \\
S + 2 & = U \\
F + 1 & = G \\
U + 4 & = Y \\
N + 2 & = P
\end{align*}
\]

\[
Q = \{1, 4, 2\} \quad Q = \{4, 2, 1\} \quad Q = \{2, 1, 4\} \quad Q = \{1, 4, 2\} \quad Q = \{4, 2, 1\} \quad Q = \{2, 1, 4\} \quad Q = \{1, 4, 2\}
\]

Figure 8
36. The following code is intended to remove a node p from a doubly linked list. Assume that we know that p is in the list, so the list is not empty.

```java
prev = p.getPrevious();
succ = p.getNext();
if (p == front) {
    front = front.getNext();
    if (front == null) rear = null;
    else front.setPrevious(null);
}
else prev.setNext(succ);
if (p == rear) {
    rear = rear.getPrevious();
    rear.setNext(null);
}
else succ.setPrevious(prev);
```

Consider the following two doubly linked lists shown below:

(a) front
A ——— B ——— C
  ▼
p
(b) front
A ——— ——— C
  ▲
p
(3 marks) If the above code can be executed on list (a) of the figure without crashing, draw the doubly linked list (including front and rear) resulting after it is executed; otherwise explain why the code would crash and which statement would throw an exception.

(2 marks) If the above code can be executed on list (b) of the figure without crashing, draw the doubly linked list (including front and rear) resulting after it is executed; otherwise explain why the code would crash and which statement would throw an exception.

NullPointerException.
Line 13.
37. Consider the singly linked list in Figure 9 where each node stores a String value:

![Singly linked list diagram](image)

Figure 9

The following code fragment is executed on the above list. Method `getValue()` returns the value stored in a node.

```java
LinearNode<String> p = front;
String[] arr = new String[3];
int count = 3;
try {
    while (count > 0) {
        if (p.getValue().equals("B") || p.getValue().equals("F")) {
            arr[count-1] = p.getValue();
            count = count - 1;
        }
        p = p.getNext();
    }
    System.out.println("Finish");
} catch (NullPointerException e) { System.out.println("Null pointer"); } catch (IndexOutOfBoundsException e) { System.out.println("Invalid index"); } catch (Exception e) { System.out.println("Error"); }
```

(4 marks) Fill in the following table while you trace through the code fragment:

<table>
<thead>
<tr>
<th>Loop Iteration</th>
<th>count (line 5)</th>
<th>p.getValue()</th>
<th>Enter if block? (line 6)</th>
<th>count (line 10)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>&quot;B&quot;</td>
<td>Yes</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>&quot;D&quot;</td>
<td>No</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>&quot;F&quot;</td>
<td>Yes</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>NullPointerException</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

(1 marks) What does the code fragment output when it is executed on the above figure?

"Null pointer".
For the following two questions write algorithms in Java or in detailed Java-like pseudocode like the one used in the lecture notes.

38. (8 marks) Consider an array A storing \( n > 1 \) different objects, which are either positive integers or letters of the alphabet. These objects are sorted in increasing order with all of the integers appearing before all of the letters. At least one value in A is an integer and at least one value in A is a letter. Write an algorithm \( \text{invertLetters}(A, n) \) that changes the positions of the letters stored in A so the letters appear in decreasing order. So, for example, if the array A is as in the figure on the left, the modified array must be as in the figure on the right.

You can either use a stack or a queue in your solution, but not both; you need to decide which of these auxiliary data structures you need to use to answer this question. You cannot use any other additional data structures (no additional arrays).

The only operations that you can perform on the stack are \( \text{push()} \), \( \text{pop()} \), \( \text{peek()} \), and \( \text{isEmpty()} \); you can write \( \text{Stack auxStack = new Stack()} \) to create an empty stack. If you decide to use a queue the only operations that you can perform on the queue are \( \text{enqueue()} \), \( \text{dequeue()} \), \( \text{first()} \), and \( \text{isEmpty()} \); you can write \( \text{Queue auxQueue = new Queue()} \) to create an empty queue.

If you use pseudocode, you must use sufficient detail. For example, you can write \( A[i] = x \) or \( s.push(A[i]) \), but you cannot write statements like ”add x to the array A”, or ”remove element x from A”. To determine if an element of the array is an integer, you can use ”if (A[i] instanceof Integer)”. 

![Array A with integers and letters sorted in increasing order](image.png)
public void invertLetters(Object[] A, int n) {
    Stack auxStack = new Stack();
    for (int i = 0; i < A.length; i++)
        if (!(A[i] instanceof Integer)) auxStack.push(A[i]);
    for (int i = 0; i < A.length; i++)
}

public void invertLetters(Object[] A, int n) {
    Stack Queue auxQueue = new Queue();
    for (int i = 0; i < A.length; i++)
        if (!(A[i] instanceof Integer)) auxQueue.enqueue(A[i]);
    for (int i = A.length - 1; i >= 0; i--)
}

public void invertLetters(Object[] A, int n) {
    Stack auxStack = new Stack();
    int i = 0;
    int firstLetter = 0;
    for (i = 0; i < A.length; i++)
        if (A[i] instanceof Integer) i++;
    firstLetter = i;
    for (i = firstLetter; i < A.length; i++)
        auxStack.push(A[i]);
    for (i = firstLetter; i < A.length; i++)
        A[i] = auxStack.pop();
}
39. (8 marks) Consider a singly linked list formed by node objects of class \textit{LinearNode}. This class provides methods \texttt{getNext()} and \texttt{setNext()} to get and set the next nodes in the list, and \texttt{getValue()} to get the value stored in a node; \texttt{LinearNode(x)} creates a new node storing value \texttt{x}. Each node stores an integer value and the values are stored in the nodes in decreasing order, so the first node stores the largest value, the second node stores the second largest value, and so on. Let \texttt{front} be a reference to the first node in the list.

Write an algorithm \textit{insert(x)} that adds a new node storing the integer value \texttt{x} into the list in the proper position so the values in the list appear in decreasing order. You can assume that the list is not empty, that there are no duplicated values in the list, and that \texttt{x} is different from all the values in the list. For example if the list is as list (a) below and we execute \textit{insert(5)} then the resulting list must be as in list (b).

![Diagram](image-url)
public void insert(int x) {
    LinearNode<Integer> newNode = new LinearNode<>(x);
    LinearNode<Integer> current = front;
    LinearNode<Integer> previous = null;
    while ((current != null) && (current.getValue() > x)) {
        previous = current;
        current = current.getNext();
    }
    if (current == front) {
        newNode.setNext(front);
        front = newNode;
    } else {
        previous.setNext(newNode);
        newNode.setNext(current);
    }
}

public void insert(int x) {
    LinearNode<Integer> newNode = new LinearNode<>(x);
    LinearNode<Integer> current = front;
    if (current.getValue() < x) {
        newNode.setNext(front);
        front = newNode;
    } else {
        while ((current.getNext() != null) && (current.getNext().getValue() > x)) {
            current = current.getNext();
        }
        newNode.setNext(current.getNext());
        current.setNext(newNode);
    }
    newNode.setNext(current.getPrevious());
}
7  (8 marks) BONUS: go back and complete the other algorithm design question!
I am answering: Q38 OR Q39 (Circle One)

(Use the rest of this page for rough work; this page will not be marked)