COMPTNG 20A
Spring 2020
Final
6/10/2020
Time Limit: 24 Hours

This exam contains 10 pages (including this cover page) and 9 questions.
The maximum number of points is 100.

The exam is open-book, open-note, and open-Internet. You can use `java` and IDEs. The only restriction is you **must not communicate or collaborate with other students in the class** during the exam period (the time between when the final is uploaded to CCLE and when submissions close). This includes that you may not post questions on forums etc. You may email the instructor or TA with any clarifying questions about the exam during the exam period; the teaching team will not respond to general/conceptual questions but can make clarifications or answer questions about logistics.

Feel free to attach extra pages if needed.

Please submit your exam as a single PDF via CCLE.

Grade Table (for staff use only)

Question	Points	Score
1	5	
2	5	
3	5	
4	5	
5	5	
6	5	
7	25	
8	20	
9	25	
Total:	100	

1. (5 points) Say we write a program using some FileInputStreams and forget to close a few of them. What is the most likely outcome of this mistake?

    A. The program will not compile due to a checked exception.

    B. The program will throw a runtime exception when it exits.

    C. Exceptions will be thrown but Java will still clean up the streams as long as main() ducks the exceptions.

    D. No exceptions will be thrown.

2. (5 points) Imagine we have a runtime exception type A which inherits another runtime exception type B. Also imagine we have a separate type C which implements Throwable. If we write `try { throw new A(); } catch (A a) { System.out.println("hi1"); } catch (B b) { System.out.println("hi2"); } catch (C c) {System.out.println("hi3"); } finally { throw new C(); }`, which of the following values will be printed to the terminal?

    A. Only hi1.

    B. Only hi2.

    C. Only hi3.

    D. hi1 and hi3.

    E. hi2 and hi3.

    F. hi1, hi2, and hi3.

3. (5 points) It is possible to specify where a Swing component is placed in a container using pixel coordinates rather than using a layout manager.

    A. True

    B. False

4. (5 points) The two ways to define action listeners for JButtons are (1) lambda expressions and (2) private inner classes which implement ActionListener.

    A. True

    B. False

5. (5 points) Autoboxing means that Java will automatically convert generic type parameters which are primitive data types to their corresponding wrapper classes.

    A. True

    B. False

6. (5 points) Say we are writing a function that copies a generic List src to another generic List dest. What is the most specific/restrictive method signature that is still valid?

   A. `public <T> void Copy(List<? extends T> dest, List<T> src)`

   B. `public <T> void Copy(List<T> dest, List<T> src)`

   C. `public <T> void Copy(List<? extends T> dest, List<? super T> src)`

   D. `public void Copy(List<?> dest, List<?> src)`

   E. `public void Copy(List dest, List src)`

   F. `public <T> void Copy(List<? super T> dest, List<? extends T> src)`

7. (25 points) A *stack* is a data structure that

   - allows duplicates and
   - supports last-in, first-out (LIFO) access.

   (Queues support FIFO access, while stacks support LIFO access.) With LIFO access, you retrieve the most recently added element.

   Write a generic `class` named `MyStack<E>` with the following specification.

   `MyStack<E>` has one `public` constructor

   ```
   public MyStack()
   ```

   which creates an empty `MyStack<E>`.

   `MyStack<E>` has the method

   ```
   public int size()
   ```

   which returns the number of elements within the stack.

   `MyStack<E>` has the method

   ```
   public void push_back(E e)
   ```

   which adds an `E` to `MyStack<E>`.

   `MyStack<E>` has the method

   ```
   public E pop_back() throws EmptyStackException
   ```

   which retrieves and removes the `E` that was most recently added to `MyStack<E>`. `pop_back` throws `java.util.EmptyStackException` if `MyStack<E>` is empty.

   `MyStack<E>` has the method

   ```
   public E peek()
   ```

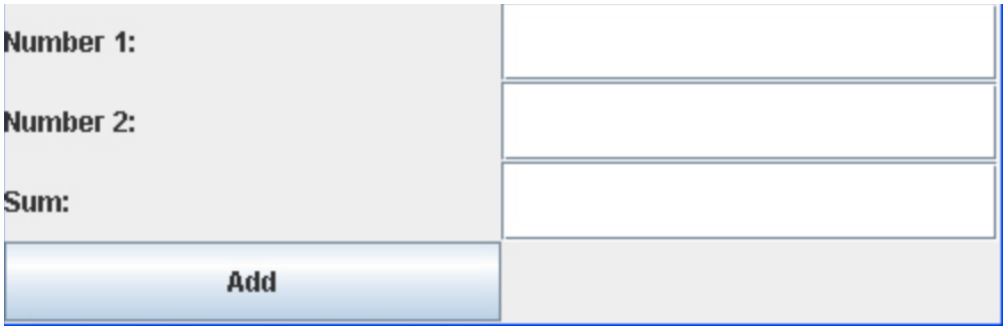
   which retrieves but does not remove the `E` that was most recently added to `MyStack<E>`. `peek` returns `null` if `MyStack<E>` is empty.

For example, we could use `MyStack<E>` as follows.

```
public class Test {
  public static void main(String[] args) {
    MyStack<Integer> stack = MyStack<>();
    stack.push_back(1);
    stack.push_back(2);
    stack.push_back(3);
    System.out.println(stack.pop_back()); //output 3
    System.out.println(stack.peek());  //output 2
    System.out.println(stack.peek());  //output 2
    System.out.println(stack.peek());  //output 2
    System.out.println(stack.pop_back());  //output 2
    System.out.println(stack.pop_back());  //output 1
    stack.push_back(1);
    stack.push_back(2);
    stack.push_back(3);
    System.out.println(stack.pop_back()); //output 3
    stack.push_back(4);
    stack.push_back(5);
    System.out.println(stack.pop_back()); //output 5
    System.out.println(stack.pop_back()); //output 4
    System.out.println(stack.pop_back()); //output 2
    System.out.println(stack.pop_back()); //output 1
  }
}
```

Hint. Make `MyStack<E>` have a `private ArrayList<E>`. (However, do not make `MyStack<E>` inherit `ArrayList<E>`. That's not what I mean.)

8. (20 points) Part 1: Write a class that produces the below GUI (or a close approximation thereof).



Clarification: Ignore the blue borders, that is just to indicate the edge of the window.

Clarification: It is ok if the textboxes and Add button grow when the window resizes.

Clarification: There is nothing (just empty space with a white background) to the right of the Add button.

Part 2: Write code that takes the numbers in the Number 1 and Number 2 textboxes and puts the sum in the Sum textbox. For full credit you must use a lambda expression.

Clarification: Your solution to part 2 can be written within the class you write for part 1. If you write your solution to part 2 separately, please clearly indicate where you would insert your solution into your class from part 1.

9. (25 points) Imagine you have a language written with the 5 letters `k`, `m`, `g`, `j`, and `b`. In this language, the dictionary ordering of the letters are as stated; `k` goes before `m`, `m` goes before `g`, `g` goes before `j`, and `j` goes before `b`.

   You are given an `ArrayList<String>` of words in this language. Write code that sorts these words in the dictionary ordering of this language. Assume the `String`s contain only `k`, `m`, `g`, `j`, and `b`. The `String`s contain no spaces or any punctuation.

```
public class Test {
  public static void main(String[] args) {
    ArrayList<String> str_arr = new ArrayList<>();

    //add Strings into ArrayList
    str_arr.add("mmjmjbmmm");
    str_arr.add("kgkbjjbbgbbjb");
    str_arr.add("bjggjggkjmkmj");
    str_arr.add("bbmmjmb");
    ...

    //start writing code here
    ...

  }
}
```

   Clarification. The 4 words shown as examples have the ordering

```
"kgkbjjbbgbbjb"
"mmjmjbmmm"
"bjggjggkjmkmj"
"bbmmjmb"
```

   Hint. `String` has the methods

```
public char charAt(int index)
```

   which `returns` the `char` at the specified index, and

```
public int length()
```

   which `returns` the length of a `String`.