This is the midterm examination for

CSC212: Data Structures

as taught by Pablo Frank and Nicholas Howe in Fall 2022.

The following materials are **permitted** while taking this examination:

- a single 8.5x11 sheet of paper (double-sided) containing your own handwritten or typed notes
- a blank exam booklet (provided)

Honor code: no other resources are permitted during this exam.
This includes (but is not limited to): textbooks, online materials, tutors, teaching assistants, and other students.

YOUR NAME:

Total points: _____ / 82

Vocabulary (16 points)

Define the following terms:

Interface _____
_____
_____

Child Class _____
_____
_____

Generic Data Structure _____
_____
_____

Exception _____
_____
_____

Tail Recursion _____
_____
_____

Iterative Method _____
_____
_____

Sequence _____
_____
_____

Declaration _____
_____
_____

Java (4 points)

If we compiled the following Code snippet, there would be an error. Indicate the line number and explain the cause of the error.

```
1  public class Context {
2      public static void staticMethod() {
3          nonstaticMethod();
4          Context c = new Context();
5          c.nonstaticMethod();
6      }
7
8      public void nonstaticMethod() {
9          staticMethod();
10         Context c = new Context();
11         c.staticMethod();
12     }
13
14
15     public static void main(String[] args) {
16         staticMethod();
17         nonstaticMethod();
18     }
19 }
```

Line number: _____

Explanation: _____
_____
_____

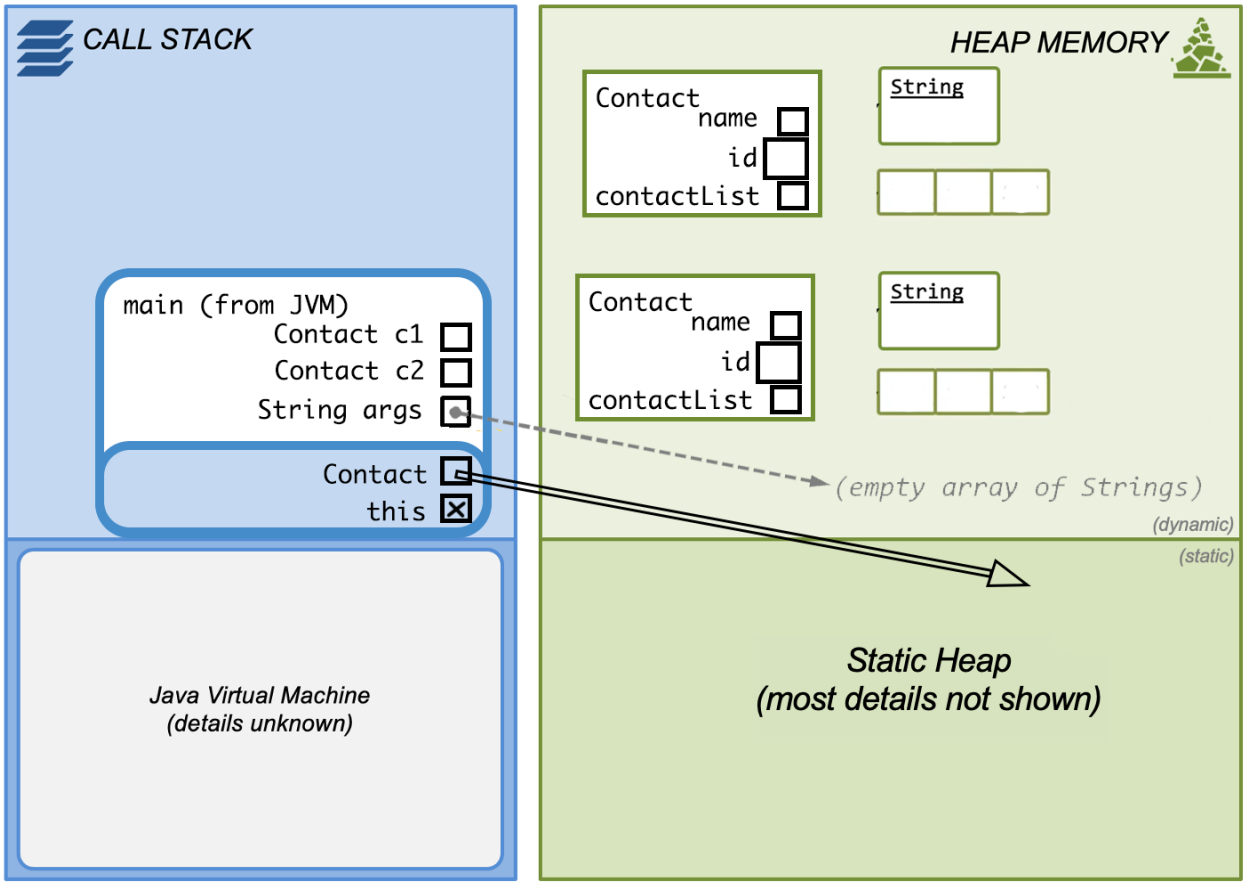
Code Analysis (12 points)

The class `Contact` is shown below.
Using the provided memory map on the next page, complete the diagram showing the state of the dynamic memory at the moment line 38 is executed. (You don't need to do anything with the static memory section.)

```
1  public class Contact {
2
3      public String name;
4      private int id;
5      private Contact[] contactList;
6      private int numContacts = 0;
7
8      public Contact() {
9          this.name = "";
10         this.id = -1;
11         this.contactList = new Contact[3];
12     }
13
14     public Contact(String name, int id) {
15         this.name = name;
16         this.id = id;
17         this.contactList = new Contact[3];
18     }
19
20     public void addContact(Contact other){
21         if (this.numContacts < 10){
22             this.contactList[this.numContacts++] = other;
23         }
24     }
25
26     public String getContactName(int i){
27         if (i>=0 && i < 10){
28             return this.contactList[i].name;
29         }
30         return "No such index";
31     }
32
33     public static void main(String[] args) {
34         Contact c1 = new Contact("Ada", 1);
35         Contact c2 = new Contact("Ben", 2);
36         c1.addContact(c2);
37         // What is the memory like at this point?
38         System.out.println(c1.getContactName(0));
39     }
40 }
```

Code Analysis continued**:**

Memory Map Template: fill in values in the empty boxes as appropriate to show the state of the dynamic memory when the program above reaches line 38.



Classes, Interfaces, and Inheritance (10 points)

Given the class Canine and the Animal interface shown below:

```
1  public class Canine {
2    public void numLegs(){
3      System.out.println("4 legs ");
4    }
5    public void cover(){
6      System.out.println("hair");
7    }
8    public void movement(){
9      System.out.println("runs and leaps");
10   }
11 }
```

```
1  interface Animal {
2    //prints an animal sound (oink, miau, etc)
3    public void animalSound();
4    //prints a movement type (fly, run, etc)
5    public void movement();
6  }
```

Answer the following questions:

(a) If we want to create a new class called `Dog` that extends `Canine` and implements the `Animal` interface, which methods besides the constructor **must** be written explicitly inside the `Dog` class?

_____
_____
_____

(b) If we write the class `Dog` correctly, which methods will be available for an object of type `Dog`?

_____
_____
_____

(c) How could we ensure that objects of the `Dog` class print something other than 'runs and leaps' when the 'movement' method is called?

_____
_____
_____

(d) Can we add new methods to `Dog` if it extends `Canine` and implements `Animal`?

_____
_____
_____

(e) If we create a `Dog` object, and store it within an array of `Animal` objects, which methods can we access directly via the array?

_____
_____
_____

Recursion (16 points)

Consider the recursive method shown below. For each item, predict (i) the return value for the call shown, and (ii) the total number of times the method is called, including the call shown.

```
public static int recursive(int x) {
    if (x == 0) {
        return 0;
    } else if (x < 0) {
        return recursive(-x);
    } else {
        int half_x = x/2;
        if (x==half_x*2) {
            return recursive(half_x);
        } else {
            return 1+recursive(half_x);
        }
    }
}
```

(a) `recursive(2);`

Return value: _____

Number of calls: _____

(b) `recursive(5);`

Return value: _____

Number of calls: _____

(c) `recursive(32);`

Return value: _____

Number of calls: _____

(d) `recursive(-1);`

Return value: _____

Number of calls: _____

Hash Tables (16 points)

Consider the hash table shown below, which uses open addressing with linear probing. For each question part, determine (i) which line(s) of the table would **be examined** in order to fulfill the operation shown, (ii) which line(s) of the table would **change** after the operation shown. If the table contents would remain unchanged following the operation, then indicate <None>. Note: each subpart is independent of the others; you should assume that you always start in the configuration shown. The hash function used is the mod of the table length (7).

Row	Key	Value
0		
1	50	“Banana”
2	22	“Elderberry”
3	73	“Cherry”
4	9	“Durian”
5		
6	48	“Apple”

(a) `lookup(9);`

Examined: _____

Changed: _____

(b) `lookup(57);`

Examined: _____

Changed: _____

(c) `lookup(19);`

Examined: _____

Changed: _____

(d) `store(19, “Fig”);`

Examined: _____

Changed: _____

(e) `store(55, “Grapefruit”);`

Examined: _____

Changed: _____

(f) `delete(20);`

Examined: _____

Changed: _____

(g) `delete(50);`

Examined: _____

Changed: _____

(h) `delete(9);`

Examined: _____

Changed: _____

Data Structure Identification (8 points)

Java's ArrayDeque class offers a large number of methods to choose from. By limiting ourselves to a subset of these, we can simulate the behavior of a different data structure such as a stack or queue. Of these options, which best describes the subset of methods shown for each question part? (You can also answer "neither".)

(a) `addFirst`, `removeFirst`: _____

(b) `addLast`, `removeLast`: _____

(c) `addFirst`, `removeLast`: _____

(d) `addLast`, `removeFirst`: _____

(e) `push`, `pop`: _____

(f) `add`, `remove`: _____

(g) `removeFirstOccurrence`, `removeLastOccurrence`:
_____

(h) `peekFirst`, `peekLast`: _____

This page intentionally left blank as scratch paper.

This page intentionally left blank as scratch paper.

This page intentionally left blank as scratch paper.