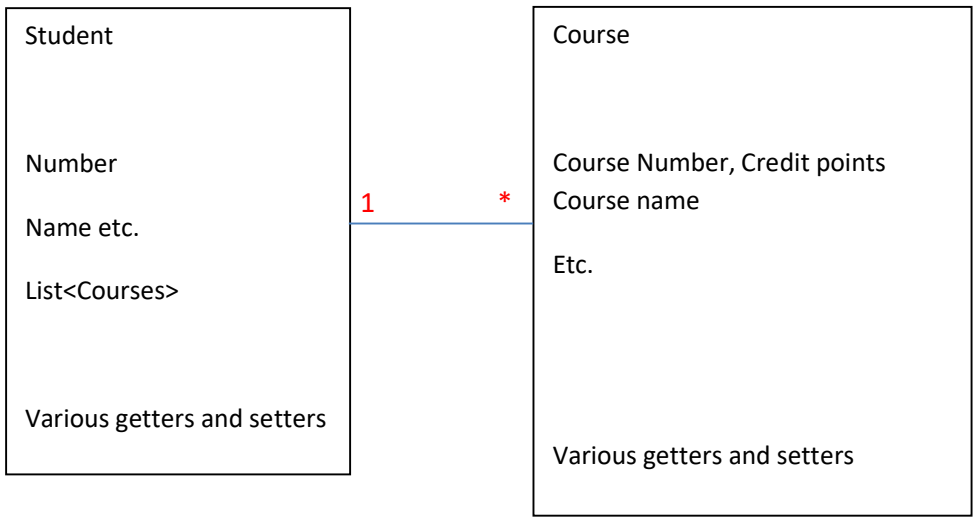
SENG2200/6220 –Programming Languages & Paradigms
Computer Lab for Week 2, Semester 1, 2020

Solutions

Part 1

1. Draw UML



2. Write an implementation

```
Public class Student
{
        Private int number;
        Private int name;
        …
        Public Student()
        {
                …
        }
        gettersAndSetters()
        {
                …
        }
}
```

3. Complete the constructor
   Set the data inside the constructor?

4. Write java code that will instantiate a Student object
   Student myStudent = new Student(params);

5. Write java code for an array of students
   Student[] studArray = new Student[2];

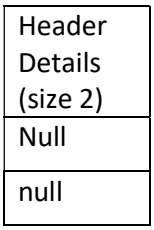
6. Populate the array

```
For(int I = 0; I < studArray.length;  i++)
{
        studArray[i] = new Student( params );
}
```

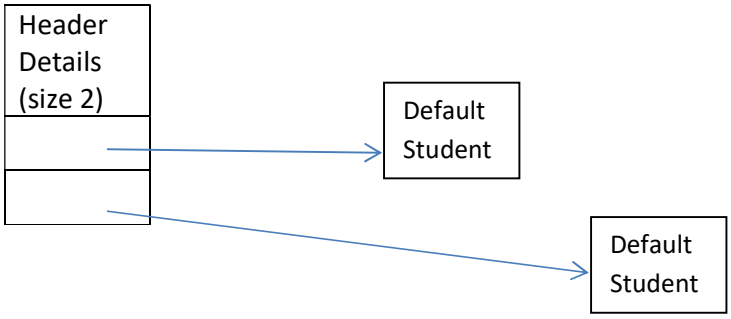
7. Compare this to C++

Sorry if this part isn't perfect, its been a while since ive had to write this stuff down. Especially since I've been doing a lot of C# its all blurred into one a bit.
I have to re-learn this every year too.

In Java, q5 gives

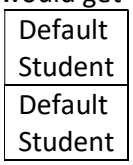


In Java, q6 gives

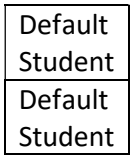


In C++ it would vary. If we declared:

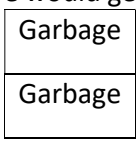
Student myArray[2]; // Two students init with default constrctor
We would get // or a compile error if no def constr.



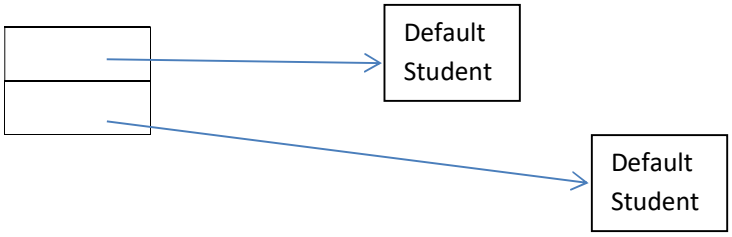
After putting myArray[i] = Student(); // Init with def constr (or error id none)



If instead we declared using **pointers**
Student *myArray[2];
We would get

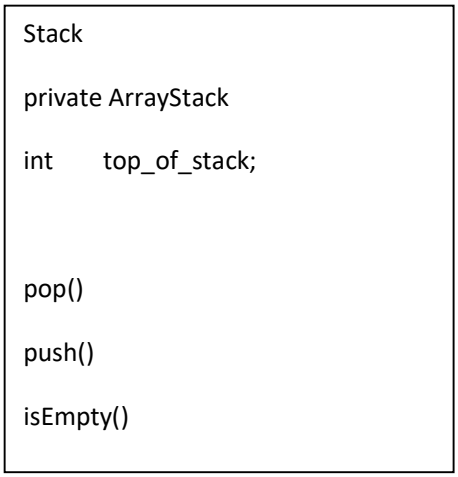


After:
myArray[i] = new Student();



Here the LHS (the array) is Stack, and RHS is heap.

8. How do we make it a Stack?
   Throw an interface around it.



Part 2

We don't cover generics until topic 6 – we'll do this all with "Objects" until then

9. Write Java Node – Objects and casting

```
public Class Node

{

    private Object data;

    private Node nextNode

    Constructor()

    gettersAndSetters()

}
```

10. Write a Queue

```
public class Queue
{
        private int size;
        private Node head;
        private Node tail;

        public Queue(){ tail = head = null };
        public add(Object object)
        {
        if(head == null)
        {
                head = tail = new Node(object);
        }
        else
        {
                tail.next = new Node (object);
                tail = object's node;
        }
        size++;
        }

        Similar for pop()
}
```

11. Expand node for Deque – this involves making the List doubly linked.
    Add 'private Node previous'
12. Same as 10, but considering updating tail, and adding in the new methods.

13. Likely class discussion around here about a few ideas. Actual answer is two sentinel nodes. Explain, probably by drawing on the board, what they are. Advantage being that you can ignore all the ‘if head == null’ and if(head == tail) checks.