MIDTERM EXAM INSTRUCTIONS

1. Midterm Exam: **25 points w/ 0 E.C. points**
2. Due Date & Time: **03-22-2021 at 09:00 AM**

WHAT TO SUBMIT

1. Take-home Exam Report, 1 PDF

HOW TO SUBMIT AND THE RULES TO FOLLOW

- Submit via iLearn, the Assignment and Exam Submission section
- Please follow the exam instructions
- Please follow the Course Policy on Student Conduct and Academic Honesty

PERFORMANCE TRACKER		
ASMT	GRADE	YOUR GRADE
ZOOM	05	
01	20	
02-PREPARATION	25	
02	75	
03	75	
MIDTERM EXAM 01	25	
TOTAL	225	

A: 90-100% **B**: 80-89% **C**: 70-79% **D**: 60-69% **F**: 0-60%
The course grader provides feedback to your assignments on iLearn.

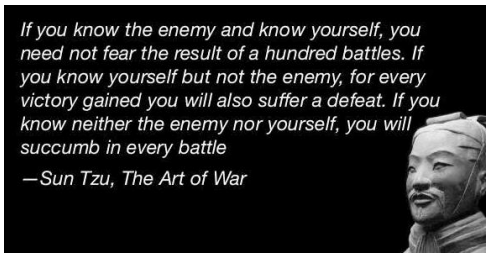
ABOUT

The goal of this take-home exam is for us to **know what we do not know**.

We are taking this exam as seriously as how we take an actual exam in class. Please,

1. Follow all the rules and the guidelines listed at the top of page 1 and page 2
2. Read each question carefully before answering

We will go through the answers to all the exam questions together in class.



STEP A – Take the Exam, **10 points**

1. Allocate 75 quiet minutes to take the exam on page 2 to the last page.
2. Record the date and time when you start.
3. Stop right at minute 75. Record the date and time when you stop the exam.

STEP B – Correct Your Answers, **10 points**

1. Review the related course materials and write code when necessary to find a correct answer for each question. We should be able to find all the answers using the packages, the in-class discussions, our assignments, and the other course materials.
2. At the end of each of your oringal answers, type in *italic* text and:
   - Give your orginal answer a score.
   - List all the mistakes then explain why, you think, you made the mistakes. Add the correct answer you found. Document how you found the correct answer. Document where you found the materials which support the answer.
   - If you did not make any mistakes, please document how you verified that your answer was accurate. Document how you found the correct answer. Document where you found the materials which support the answer. Outline how you could have done better.
   - Record your total score out of 100 points for all the orginal answers.

STEP C – Reflect and Retake the Exam, **5 points**

1. **Problem Solving**: Reflect if you managed the exam time efficiently and if you strategized your test-taking successfully.
2. Repeat steps A to C again if necessary. Please keep appending new contents as directed in Step B.2.
3. Think if the same topics will be tested again in our final exam, what questions we may get.

It is a good idea to do every step of this assignment thoroughly. We are creating a set of materials which we will use to review for the final exam. And this is also the best way to prepare ourselves to succeed in the second half of the semester. Thank you.

1. Section, Date and Time:
   CSC 220.02+03, Due ##-##-#### at ##:## PM

2. Midterm Exam (2 exams, 0 dropped): 100 points

Full Name in CAPITAL LETTERS | SFSU ID

3. To prepare for this exam, please review all the related materials including WEEK 01-08 packages, slides, mock-up exam(s), reading assignments, in-class practices, sample programs posted in the File Manager, and assignments.
4. You do not need to print this exam. No paper. No handwriting. No scanning. Please type up all your answers in the answer space available in the exam. The provided exam will be in Microsoft Word format. Please submit a single PDF via iLearn.
5. All the rules of an actual exam apply to this exam such as: closed books, closed notes, and no communication with anyone except the course instructor. The course instructor will be available on Zoom or email during the exam time: zoom.ducta.net
6. Please ask all your questions, if any, during the review sessions. Thank you.

HONOR CODE:

- Please follow the CS Department's policies: https://cs.sfsu.edu/student-policies

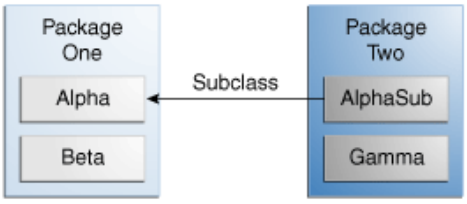
- Please follow the course's policies: http://csc340.ducta.net/00-README-StudentConduct_AcademicHonesty.pdf

---

PART A – 40 Points

A.1 - 10 Points
Please complete the "where the members of the Alpha class are visible" table by filling in Y or N.

Modifier	Alpha	Beta	AlphaSub	Gamma
public				
protected				N
no modifier			N	N
private		N	N	N



If a member of the class **Beta** is **protected**, is this member visible to **Gamma**? If yes, please explain why. If not, please explain what we should do make it possible.

A.2 - 10 Points
- What are the 2 ways to use the keyword "**super**" in a class? - Give code examples to demonstrate your answers.

A.3 - 10 Points
When using a linked list for stack implementation, we use the first Node as the top entry of stack. Explain, in this linked list implementation, how we **add** and how we **remove** an entry from a stack. *Use linked nodes diagrams to save your time.*

add

remove

Explain one major difference between the behaviors of the LinkedBag and the Stack which we implemented.

A.4 - 0 Points – A Practice Problem
What are stored in each activation record and Why? Which method is pushed in the Program Stack first? Which method is popped out of the Program Stack last?

A.5 - 10 Points - Part A of Assignment 03:

What is the ouput (what are in the Bag) when the 2 below lines are executed? **Please show the steps.**

```
String[] items = {"Z", "Y", "Y", "X", "S", "C", "A", "E", "M"};
testAdd(aBag, items);
```

What is the output (what are in the Bag) when the 3 below lines are executed? **Please show the steps.**

```
String[] testString = { "X", "Y", "Z" };
aBag.removeAllOccurences(testString);
displayBag(aBag);
```

PART B – 60 Points

B.1 - 5 Points
Which of the statement(s) are erroneous and why?

```
MidtermExam midtermExam = new Exam();                // A
SFSUStaff person = new Person();                    // B
StackInterface<String> s = new ArrayStack<>();      // C
```

B.2 - 5 Points
What is the output if any?

```
abstract class Person {

    private final String name;

    protected Person(String name) {
        this.name = name;
    }

    public String getName() {
        return this.name;
    }

    public void setName(String name) {
        this.name = name;
    }
}

class Student extends Person {

    public Student(String name) {
        super(name);
    }

    public static void main(String[] args) {
        Student stu = new Student("Mickey");
        System.out.println(stu.getName());
        stu.setName("Super Mouse");
        System.out.println(stu.getName());
    }
}
```

*Attention to the keyword "**final**"*

B.3 - 10 Points
Anything wrong with the code? If yes, how to fix?

```
class CSC220 {
    private CSC220(){}
    private CSC220(int x){}
    private CSC220(int x, int y){}
}
```

And is it possible to create a sub class to this class? Why?

B.4 - 0 Points – A Practice Problem
What is the output if any?

```
    public static void main(String[] args) {
        int x = 12;
        System.out.println(sumOf(x) - 42);
    }

    public static int sumOf(int n) {
        int sum = 15;
        if (n == 0) {
            System.out.println("Base case: n is " + n);
            sum += 5 + n % 2;
        } else {
            sum = sumOf(n - 3) + n;
        }
        System.out.println(sum);
        return sum;
    }
```

B.5 - 10 Points
This program outputs 10 lines. What are they?

```
Stack<String> resume = new Stack<>();
resume.push("JavaScript");
System.out.println("Is empty: \t" + resume.isEmpty());
resume.push("Scala");
resume.push("C++");
resume.push("Dart");
resume.push("Go");

resume.pop();
System.out.println("Stack : \t" + resume);
resume.push("Python");
System.out.println("search() : \t" + resume.search("Scala"));
System.out.println("pop() : \t" + resume.pop());
System.out.println("pop() : \t" + resume.pop());
System.out.println("search() : \t" + resume.search("Dart"));
System.out.println("After pop() : \t" + resume);
System.out.println("pop() : \t" + resume.pop());
System.out.println("Is empty : \t" + resume.isEmpty());
System.out.println("Stack:  \t" + resume);
```

1.

2.

3.

4.

5.

6.

7.

8.

9.

10.

B.6 - 15 Points

Implement *findTheThird* method in linked list that searches the bag for a given *entry*. If found,
- removes the first occurrence
- leave the second occurrence intact
- then replace third occurrence with the string "Found3rd"
- remove the rest of the occurrences

Return *false* if no replacement happened. Otherwise, *true*.

```
public boolean findTheThird (T entry)
```

Note: You may assume that *firstNode* is a private data in list which references to first node.

B.7 - 15 Points
Assume that you have the following Bag object, myBag, with n String data:

BagInterface <String> myBag = new ArrayBag< >();

Write Java statements that create a newBag object which
contains non-duplicate data in myBag and marks the duplicate data.

Example:
if myBag contains data:
"A", "A", "A", "B", "B", "C", "D", " "
newBag object should contain:
"A", "DUP.1.A", "DUP.2.A", "B", "DUP.3.B", "C", "D", " "

Hint: You can use the Bag's methods:
int getCurrentSize (); boolean isFull (); boolean isEmpty (); boolean add (T newEntry); T remove (); boolean remove (T anEntry); void clear (); int getFrequencyOf (T anEntry);
boolean contains (T anEntry); T [] toArray ();