# TRIMESTER 2 2023 COSC120 Object Oriented Programming

## Programming Assignment 1 - Question

**Must Complete:** Yes

**Weight:** 15%

**Assessment Notes:** Programming assignment. All assignments must be attempted.

**This assessment relates to:** Learning Outcomes 1 2 3

## Instructions

- Due date: Sunday the 30th of July

1. explain object oriented programming concepts including classes and objects;
2. apply object oriented design principles to algorithm design and analysis; and
3. develop computer programs using Java, which is an object oriented programming language.

## Introduction

This assignment assesses knowledge and skills relating to content covered Topics 1 to 4 (inclusive). General concepts assessed include the following:

- Variable declaration and usage
- Documentation and Java conventions
- Reading and writing files
- User input and validation
- Loops and decision structures
- Error and exception handling
- Collections and arrays
- Methods
- Classes and objects
- Analysing a user's request to inform program and class design
- Creating a program from scratch using multiple classes
- The enumerator type

⚠ **Submission notice:** your submission will be restricted to **SIX .java files**. No other file types are allowable. As such, it is important that you follow the instructions below carefully.

## Task

In this Assignment, you will design a program from scratch using the knowledge and skills you have gained over weeks 1 to 4 inclusive.

*SeekAGeek* has been a huge hit, and the Greek Geek is now the proud owner of a platform that is receiving ample attention from many geeks. He owns a small fast food takeaway on the side - *Eets 4 Gobbledy-Geeks* - which sells burgers in a geek-friendly environment. Recently, he decided to advertise *Eets 4 Gobbledy-Geeks* on *SeekAGeek*, in a clever attempt to take advantage of his large geek audience. It's been a huge success - many geeks have enjoyed their first date at *Eets 4 Gobbledy-Geeks*, but there is one hiccup: geeks are shy and don't like coming up to the counter to place their orders. As such, he has decided to install several self-help kiosks that allow geeks to search his menu at their own pace and place an order. He has sent you the following message:

> *Hi! I have another project for you. I want you to create an app that geeks can use to search my menu based on the main burger ingredients, i.e., bun-type, meat, sauce, price range, cheese/no cheese and pickles/no pickles. My bun range varies, depending on the mood of the local baker, but I ONLY have beef, chicken and vegan patties. Also, I only stock the following sauces: TOMATO, GARLIC, AIOLI (vegan friendly), BIG MAC, BBQ, CHILLI, RANCH, and my house special sauce. Users MUST be able to choose one or more sauces.*
>
> *I want users to be able to view all the menu items that meet their search criteria in the following format:*
>
> > *Heisenburger (11786)*
> > *Geek-out like a kingpin with 2x extra-thick Angus beef patties, loaded with lettuce, cheese, pickles, tomato and topped with aioli and special sauce.*
> > *Ingredients:*
> > *Bun type: Signature*

*Bun type: Signature*
*Meat: Beef*
*Sauce/s: aioli (vegan friendly), special sauce*
*Other: cheese*
*Price: $19.00*

*Users MUST be able to choose any burger on my menu. If they want to place an order, they must provide their full name, as well as their phone number (which will be used as their order number). Their order details should be written to a file in the following format:*

*Order details:*
*Name: Dr. Walter Shepman*
*Order number: 0486756465*
*Item: Heisenburger (11786)*

*If their search doesn't return any matches, they MUST be able to place a custom order, whereby their search details are added to the file, in the following format:*

*Order details:*
*Name: Walter Shepman*
*Order number: 0486756465*
*Item: CUSTOM ORDER*
**********
*Ingredients:*
*Bun type: wholemeal*
*Meat: Beef*
*Sauce/s: aioli (vegan friendly), big mac*
*Other: cheese*

Here are the all files that you need to complete the task:

- menu.txt - this contains the menu data.
- gobbledygeek.png - the image used as the icon in the sample.

**Note**: for Assignment 3, after you have learned how to use graphics, you will redesign this app in a single window/frame, and display an image of each item along with a text description (rather than having many modal dialogs). Exciting times ahead!

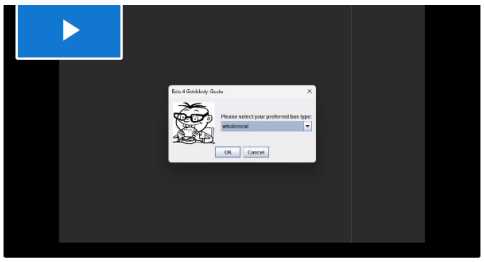
### Instructions

Carefully examine *menu.txt* and read the task description, the Greek Geek's message and the Marking Guidelines below. Using UML diagrams to help you (optional - don't submit them if you use them), design a program similar to *SeekAGeek* (Topic 4) and *FindAPet* (Tutorial 4) that meets the Greek Geek's requirements.

## Marking Guidelines

| **My program compiles AND contains NO package statements!** | **Marks allocated** |
|---|---|
| I have created TWO Enums representing 1) the burger patties AND 2) the sauces. | 1 |
| My enums each contain an appropriate switch statement to provide a visually pleasing representation of the Enum constants. | 1 |
| I have created a class named *Burger*, which contains:<br>1) all the features used to search the menu including *bun-type, meat, sauce, price range, cheese/no cheese and pickles/no pickles.*<br>2) appropriate constructor/s, setters and getters.<br>3) a method that describes the burger in the format required by the Greek Geek.<br>My *Burger* class is used to create BOTH 'real' burgers from the menu AND 'dream' burgers (representing the user's filters). | 3 |
| I have created a class named *Geek* to represent the app user (name and phone number) with appropriate initialisation of and access to fields. | 1 |
| I have created a class named *Menu* which contains:<br>1) an appropriate field (data structure) to store and access *Burger* objects.<br>2) a method to compare burgers in the field (data structure) to a user's 'dream' burger (parameter), returning an appropriate collection of matching *Burgers*. | 2 |
| I have created a class named *MenuSearcher* that contains:<br>1) a method to load the data from *menu.txt*, returning an instance of the *Menu* class.<br>2) a method that requests user input/selection of burger features (*bun-type, meat, sauce, price range, cheese/no cheese and pickles/no pickles*), returning a *Burger* object representing the user's 'dream' burger.<br>3) a method that uses the user's 'dream' *Burger* object to search the menu, presenting matches to the user, and allowing selection of a menu item OR placing of a custom order.<br>4) a method to obtain the user's information (name and phone number) returning a *Geek* object.<br>5) a method to write the Geek user's order information to a text file in the format specified by the Greek Geek.<br>6) a *main* method used to run the program by calling the above methods as appropriate.<br>7) code to handle when 1) matches are found 2) no matches are found AND 3) the user closes the dialog. | 7 |
| **To achieve full marks in the above sections, please ensure you address each of the following:** | **Marks deductible** |
| I have ensured that all mutable types are protected. | 1 |
| My program validates all user input and I've handled all exceptions. | 2 |
| My code is thoroughly documented, with Javadoc blocks for each method. | 2 |
| My program produces the correct output when the test prompts are entered (see video below) and supports searching based on ONE OR MORE sauces. | 2 |

Please watch the following video for an example of how your program should look/function. Run the two test cases shown in the video to ensure that your program is functioning correctly.



Well done! You've completed Assignment 1! Now, submit all the .java files in your *src* file (there should be 6 of them) using the Assignment 1 Submission portal.

## Submission

Ensure that you compile and test your code, handling all errors and exceptions.

ALL submissions containing plagiarism will receive a ZERO grade. All students suspected of plagiarism will be reported. Remember that:

- Copy-pasting code from the internet or other sources IS plagiarism
- Having someone else complete your assignment IS plagiarism.
- You should not under any circumstances share your code with other members of the class.
- If you use code from another source (including the lectures/tutorials provided in this unit) ensure that you include a reference to the original source in your Javadoc. E.g., "*this method was sourced and adapted from COSC120 Lecture 2 SeekAGeek.java loadInventory*"
- If there are instances of plagiarism, it can be difficult to determine who was the original author of the code. Penalties WILL be shared by both parties if the author cannot be determined.

**NOTE: if your code does not compile, you will receive an automatic zero grade. Ensure your program compiles with JAVA 17+. Ensure your code DOES NOT contain package statements, as this prevents it from compiling.**

## Additional Notes

If you are having trouble understanding this assignment, please review the lecture notes and sample code for Topics 1-4 inclusive and ensure you have completed all tutorials. If you have any questions or issues, please post on the forums or send me an email at asheple2@une.edu.au.

**Note:** any requests for extensions MUST be made through AskUNE.

Last modified: Tuesday, 25 July 2023, 6:10 AM

◄ Library subject guide - Computing

Jump to...

Assignment 1 - Submission ►

### Acknowledgement of Country

*The University of New England acknowledges that we are on Country of the Anaiwan people. UNE respects and acknowledges that its people, programs and facilities are built on land, and surrounded by a sense of belonging, both ancient and contemporary, of the world's oldest living culture. UNE also acknowledges the Gumbaynggirr, Kamilaroi, and Dhunghutti nations and pays its respect to the Elders, past and present, of these nations.*

Oorala Aboriginal Centre