# ENGR 101 - Introduction to Programming

## Mini Project 3

*January 22, 2018*

*(Due on Monday, February 5 by 5 pm)*

In Mini Project 2, you have developed a console-based e-commerce application by employing *procedural programming* (i.e., a set of functions and function calls). In this mini project, you are going to re-implement the same console-based e-commerce application by using object-oriented programming practices! The interface from user's perspective will stay the same. However, you will have a significantly different implementation in the source code level. The primary goal of this mini project is to practice using classes and objects. Hence, please use every opportunity to employ object-oriented programming concepts in your code. **<u>At minimum</u>**, you should create and use the following classes implemented with the below-provided list of attributes and methods. You *may add* more classes and methods/attributes that are not in the following list.

**List of Classes with their Attributes and Methods**

- **`InventoryProduct`**
    - o Attributes:
        - `name (str)`
        - `price (float)`
        - `stock_amount (float)`
    - o Methods:
        - `We do not specify any. You may add methods here if you see a need.`
- **`BasketProduct`** `(Bonus: 20 points)`
    - o `(This class should` ***<u>`inherit`</u>*** `from InventoryProduct with one additional attribute, basket_amount)`
    - o Methods:
        - `We do not specify any. You may add methods here if you see a need.`
- **`Basket`**
    - o Attributes:
        - `contents (dict of BasketProducts with key=product name, value= BasketProduct object)`
            - `If you do not want to do the bonus part and skip BasketProduct class, then please use InventoryProduct objects instead of BasketProduct as values here. In this case, values in your dictionary should be a list: [InventoryProduct, basket_amount] where the first item is an InventoryProduct object, and the second item is its amount in the basket.`
        - `total_value (float)`

- Methods:
    - display_contents(): implements “See Basket” functionality.
    - show_basket_submenu(): shows basket sub-menu as specified in p.3 of MP2 document. It also handles user selection, and accordingly take the next action by either calling another method (e.g., remove_item to perform menu item #2 in basket submenu), or doing the task locally (e.g., checking that user entered a valid number). Its different return values (that you will design) will tell the market class to call check_out or show_market_menu methods, if user choose basket menu item #3 or #4, respectively.
    - add_item(product): Takes a InventoryProduct, gets its amount from the user (with validity checks), and adds it to the current basket. It should also update the total_value attribute.
    - remove_item(): Asks the user to select an item to remove, and removes the selected item. It should also update the total_value attribute.
    - update_item(): Updates the amount of a user selected product in the basket. It should also update the total_value attribute.

- **User**
  - Attributes:
    - username (str)
    - password (str)
    - basket (a Basket object)
  - Methods:
    - We do not specify any. You may add methods here if you see a need.

- **Market**
  - Attributes:
    - inventory (dictionary of InventoryProducts with key=product name, value=InventoryProduct object)
      - In the __init__ method of this class, you should manually populate the contents of inventory with the grocery items specified in p.4 of MP2 document. No additional items please!
    - users (dictionary of User objects with key=username, value=User object)
      - In the __init__ method of this class, you should manually populate the contents of the users with the specified users (ahmet and meryem) in p.1 of MP2 document. You may have additional users if you like.

  - Methods:

   - `show_market_menu(user)`: shows the main menu as specified in p.2 of MP2 document. It also handles user selection, and accordingly take the next action by either calling another method (e.g., search to perform menu item #1 in main menu), or doing the task locally (e.g., checking that user entered a valid number).
   - `search(user)`: Performs the search for products as explained in p.2 of MP2 document.
   - `update_stock_amount(product_name, sold_amount)`: updates the stock amount of a given product by decreasing sold_amount from the current stock amount. This method may be called during check out.
   - `check_out(user)`: performs check out as described in p.4 of MP2 document.
   - `print_receipt(basket)`: prints receipt as described in p.4 of MP2 document.
   - `login()`: Handles login as described in p.1 of MP2 document. Returns the user object that represents the currently logged in user.

**Implementation Notes:**

- Outside of class definitions, only 2-3 lines of code should exist that do the following:
  - In one line you will create an object of Market class.
  - In the other line, you will call a proper method on the object that you created above to start the application.
- Please note that although **__init__** methods are not specified for every class in the above list, you should have an **__init__** method in each class.

**Warnings:**

- **Do not** talk to your classmates on project topics when you are implementing your projects (This is serious). **Do not** show or email your code to others (This is even more serious). If you need help, talk to your TAs or the instructor, not to your classmates. If somebody asks you for help, explain them the lecture slides, but do not explain any project related topic or solution. Any similarity in your source codes will have **serious** consequences for both parties.
- Carefully read the project document, and pay special attention to sentences that involve "**should**", "**should not**", "**do not**", and other underlined/bold font statements.
- If you use code from a resource (web site, book, etc.), make sure that you reference those resource at the top of your source code file in the form of comments. You should give details of which part of your code is from what resource. Failing to do so **may result in** plagiarism investigation.
- Even if you work as a group of two students, each member of the team should know every line of the code well. Hence, it is **important** to understand all the details in your submitted code. You may be interviewed about any part of your code.

**How and when do I submit my project? :**

- Projects may be done individually or as a small group of two students (doing it individually is recommended for best learning experience). If you are doing it as a group, only **one** of the members should submit the project. File name will tell us group members (Please see the next item for details).
- Submit your own code in a **single** Python file. Name your code file with your and your partner's first and last names (see below for naming).
    - If your team members are Deniz Can Barış and Ahmet Çalışkan, then name your code file as deniz_can_baris_ahmet_caliskan.py (Do **not** use any Turkish characters in file name).
    - If you are doing the project alone, then name it with your name and last name similar to the above naming scheme.
    - Those who **do not** follow the above naming conventions **will get 5 pts off** of their grade.
- Submit it online on LMS by **5 pm on Monday, February 5, 2018**

  **Late Submission Policy:**
    - -10%: Submissions between 17:01 – 18:00 on the due date
    - -20%: Submissions between 18:01 – midnight (00:00) on the due date
    - -30%: Submissions which are up-to 24 hour late.
    - -50%: Submissions which are up-to 48 hours late.
    - Submission more than 48 hours late will not be accepted.

**Grading Criteria? :**

| **Code Organization** | | | **Functionality** | | | | |
|---|---|---|---|---|---|---|---|
| **Meaningful variable names (3 pts)** | **Proper use of object-oriented programming concepts (4 pts)** | **Sufficient commenting (4 pts)** | **Search functionality with all of its submenus (25 pts)** | **Basket functionality with all of its submenus (30 pts)** | **Checkout functionality with all of its submenus (20 pts)** | **Logout functionality (10 pts)** | **Others (5 pts)** |

**Interview evaluation:** Your grade from interview will be between 0 and 1, and it will be used as a coefficient to compute your final grade. For instance, if your initial grade was 80 before the interview, and your interview grade is 0.5, then your final grade will be 80*0.5 = 40. Not showing up for the interview appointment will **result in** grade 0.

**Have further questions?:**

Please contact your TAs if you have further questions. If you need help with anything, please use the office hours of your TAs and the instructor to get help. **Do not walk in randomly (especially on the last day) into your TAs' or the instructor's offices. Make an appointment first. This is important. Your TAs have other responsibilities. Please respect their personal schedules!**

Good Luck!