IE5600 – Applied Programming for Industrial Systems

AY 2023/24 Semester 2

Individual Assignment 2 - PyWarehouse

Objectives

At the completion of the individual assignment, you should:

1. Develop a better understanding of how to apply the computational problem solving process to a complex problem that requires the use of appropriate data structures and algorithms.

2. Implement procedural and object-oriented programming paradigms in Python.

3. Apply appropriate user-defined data structures and algorithms.

General Requirements

要为一个典型的制造业的进出口贸易公司设计一个仓储管理系统

You are required to develop a Python program known as **PyWarehouse** which is essentially a warehouse management system for a typical manufacturing and import/export company. PyWarehouse will manage the storage locations in the warehouse, the inventory items stored in the storage locations as well as the inbound and outbound movement of these inventory items.

要实现对1.货物库存位置、2.货物存储数量、3.进仓库和出仓库的行为的管理

Figure 1 shows a layout plan of a typical warehouse divided into several zones such as reception, loading & unloading, picking and storage. As illustrated in the layout plan, the storage zone is further organised into multiple storage racks.

整个仓库会被分为三（五）个功能区

存储、装卸与接收、取货与运输

Figure 2 shows a hypothetical layout plan based on Figure 1 which will constitute the actual warehouse that PyWarehouse would be managing. In this hypothetical layout plan, the storage racks are numbered sequentially with the odd numbered racks on one side and the even numbered racks on the other side.

Each storage rack is further divided into one or more levels of storage spaces known as shelves. Each storage shelf can be further divided into one or more compartments known as storage bins. Each storage bin is characterised by various key parameters such as length, width, depth, volumetric space, and load capacity. Refer to Figure 3 for a three-dimensional view of a typical storage rack. An inventory item may only be allocated to a storage bin of sufficient volumetric space and adequate load capacity. In addition, no part of the item should protrude out of the storage bin, i.e., the length, width and depth of an item should all be less than or equal to the corresponding dimensions of the storage bin. For example, an item might meet the volumetric space and load capacity of a storage bin but the depth might be too high. In this case, the item should not be allocated to this bin.

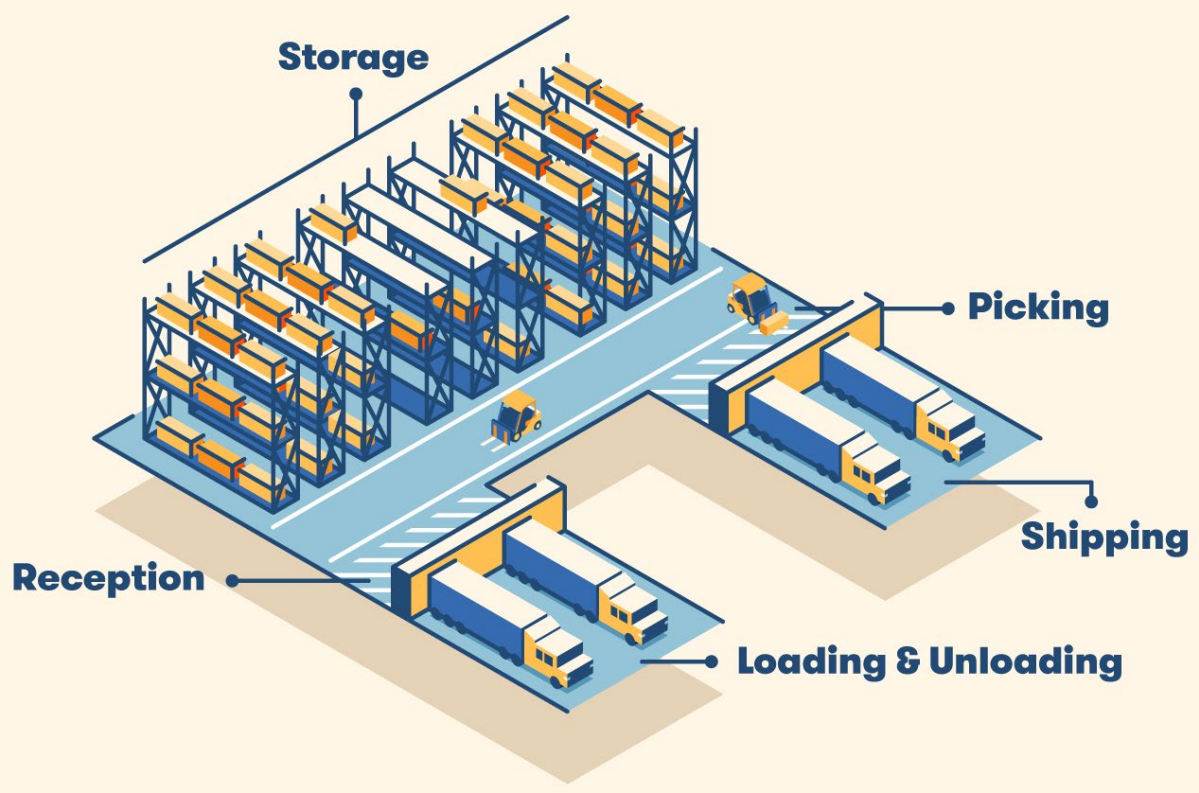

Figure 1 – Layout plan of a typical warehouse.

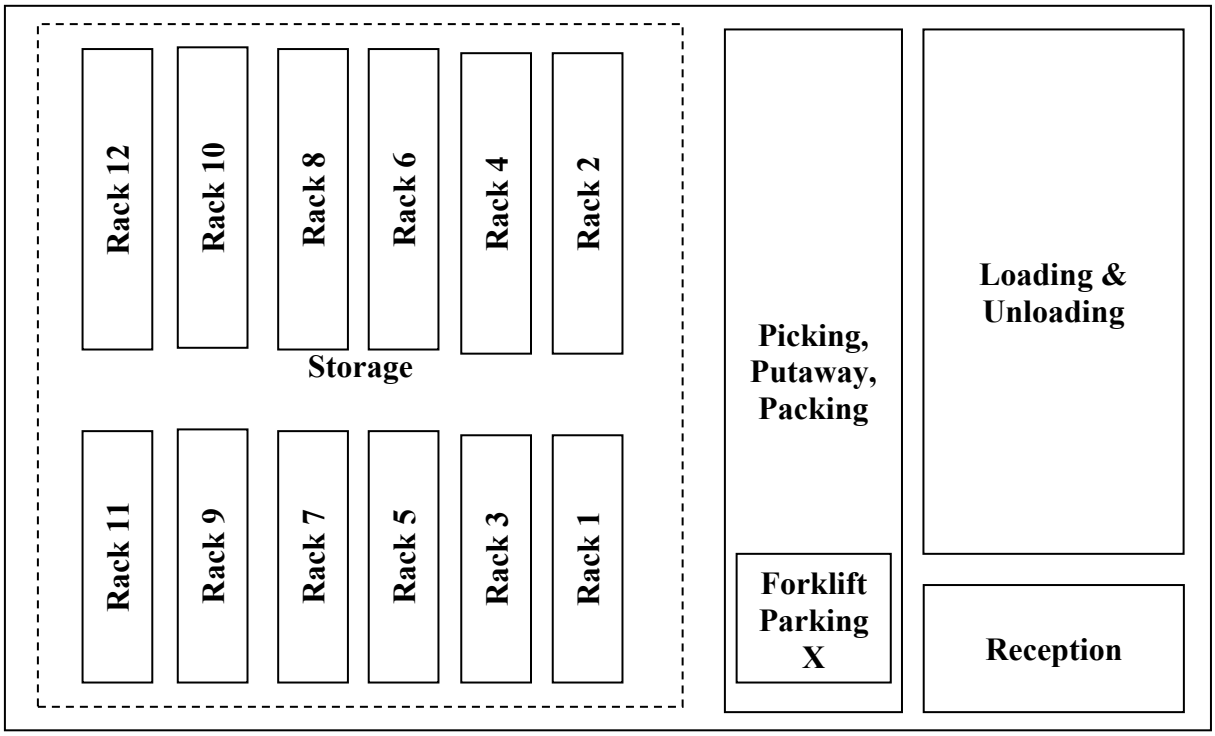

Figure 2 – Hypothetical layout plan of PyWarehouse.

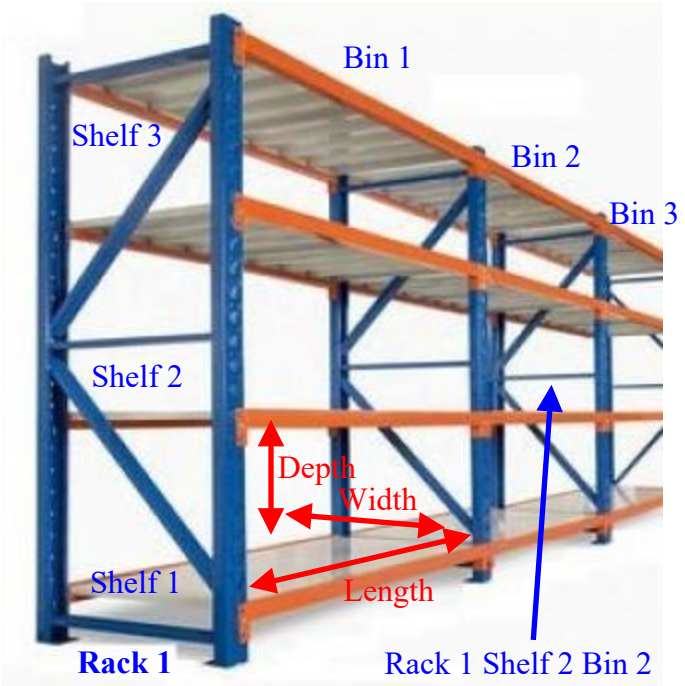

Figure 3 – Three-dimensional view of a typical storage rack (with numbering of rack, shelves and bins).

In general, the load capacity of a storage bin will remain the same or decrease as the shelf level increases in order to maintain the overall stability of the storage rack. In other words, heavier items should be stored on lower storage shelves. If a storage bin is shared among multiple items, it is necessary to check that the odd-shaped volumetric space remaining is sufficient and a heavier item is not stacked on top of a lighter item. See the two examples depicted in Figure 4.

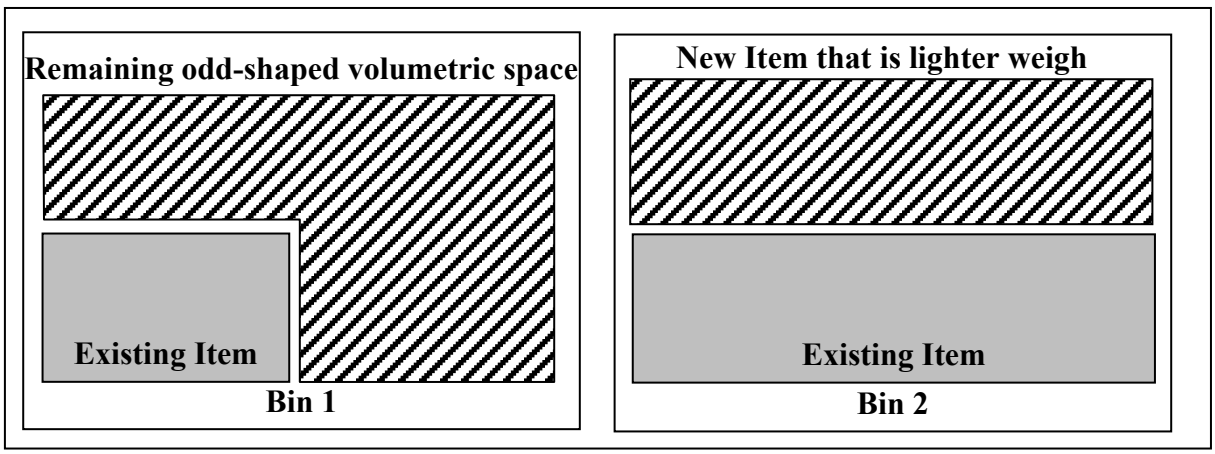

Figure 4 – Sharing of storage bins by multiple items.

Design and Implementation of Classes

Design a set of suitable classes together with the necessary inheritance and/or association relationships to represent the warehouse shown in Figure 2 that PyWarehouse would be managing. You do not need to draw a UML class diagram as part of the deliverable submission. You are only required to define the classes in a Python module and import this module for use in the actual PyWarehouse program.

You need to take into consideration that each storage rack consists of multiple storage shelves and each shelf is further divided into multiple storage bins as shown in Figure 3 and 4.

In addition, the classes also need to be able to represent inventory items that are stored in the warehouse. For simplicity, PyWarehouse is only required to track inventory items in broad lot quantity that are packaged in carton boxes or pallets. In other words, each unit of an inventory item manifest as a single carton box or pallet stored at a particular storage bin. Other than the physical dimensions, weight and quantity, you only need to keep track of the stock keeping unit (SKU) code of an item together with its name. Here is an example of an inventory item record with reference to the storage rack depicted in Figure 3:

- SKU Code – MASK
- Name – COVID-19 3-Ply Face Mask (100 Boxes)
- Dimensions (LxWxD) – 100 cm x 100 cm x 100 cm
- Volumetric Space – 1,000,000 cm^3
- Weight – 20 kg
- Quantity – 10:
    - 1 – Rack 1 Shelf 1 Bin 1
    - 2 – Rack 1 Shelf 1 Bin 1
    - 3 – Rack 1 Shelf 1 Bin 2
    - 4 – Rack 1 Shelf 1 Bin 2
    - 5 – Rack 1 Shelf 1 Bin 3
    - 6 – Rack 1 Shelf 1 Bin 3
    - 7 – Rack 1 Shelf 2 Bin 1
    - 8 – Rack 1 Shelf 2 Bin 1
    - 9 – Rack 1 Shelf 2 Bin 2
    - 10 – Rack 1 Shelf 2 Bin 2

It is mandatory to apply the object-oriented programming paradigm to this assignment. Otherwise, you would only be awarded 50% of the marks for the use cases.

Use Cases

Use the classes that you have defined to implement the following use cases for PyWarehouse without the use of any Python modules. In other words, your program should **NOT** contain any `import` statement unless you are importing your own user-defined module(s).

S/N	Use Case	Use Case Description/Business Rules

1	Create Storage Rack
	(2 marks)
	
	总共的usecase分为
	1. 创建库存架
	2. 创建库存货物
	3.增加货物到库存
	4.从库存中删除货物
	5. 查询货物
		• Create a new storage rack in the warehouse.
		• You may assume that all storage bins in a rack has the same parameters.
		
		Sample Input:
		• Rack Number – 1
		• Number of shelves – 3
		• Number of bins per shelf – 3
		• Bin Length – 200 cm
		• Bin Width – 200 cm
		• Bin Depth – 200 cm
		• Computed Volumetric Space for each Bin – 8,000,000 cm^3
		• Bin Load Capacity – 50 kg
		
		Sample Output:
		• Created the following storage bins:
		  ○ Rack 1 Shelf 1 Bin 1
		  ○ Rack 1 Shelf 1 Bin 2
		  ○ Rack 1 Shelf 1 Bin 3
		  ○ …
		  ○ Rack 1 Shelf 3 Bin 3

2	Create Inventory Item (1 marks)	• Create a new inventory item record. Sample Input: • SKU Code – MASK • Name – COVID-19 3-Ply Face Mask (100 Boxes) • Length – 100 cm • Width – 100 cm • Depth – 100 cm • Computed Volumetric Space for each Unit – 1,000,000 cm^3 • Weight – 20 kg

S/N	Use Case	Use Case Description/Business Rules
3	Add Inventory (4 marks)	• Add one or more units of an inventory item. • For each unit of inventory item, the program should search the warehouse for all available storage bins and recommend them to the user for selection. • The user should only select one storage bin for each unit. • If no suitable storage bin is available, print an error message and stop processing the current and subsequent units. • It is important to perform a check on whether a storage bin is able to accommodate the item based on the dimensions, volumetric space and weight. Sample Input: • SKU Code – MASK • Quantity – 2 • Unit 1: o Choose storage bin: ▪ Rack 1 Shelf 1 Bin 1 ▪ Rack 1 Shelf 1 Bin 2 ▪ Rack 1 Shelf 1 Bin 3 ▪ … ▪ Rack 1 Shelf 3 Bin 3 o User selects – Rack 1 Shelf 1 Bin 1 • Unit 2: o Choose storage bin: ▪ Rack 1 Shelf 1 Bin 1 ▪ Rack 1 Shelf 1 Bin 2 ▪ Rack 1 Shelf 1 Bin 3 ▪ … ▪ Rack 1 Shelf 3 Bin 3 o User selects – Rack 1 Shelf 1 Bin 1 Note that for MASK, each storage bin in Rack 1 can store 8 units based on the volumetric space. But due to the load capacity constraint (50 kg), we can only allocate two units of MASK (2 x 20 kg) to each bin.

S/N	Use Case	Use Case Description/Business Rules

4	Remove Inventory
	(1 mark)
		• Remove one or more units of an inventory item.
		• The program should display the list of all available units and the corresponding storage bin to the user for selection.
		• The user should select the required units by entering the corresponding storage bin number.
		
		Sample Input:
		• SKU Code – MASK
		• Quantity – 1
		• Choose unit:
		  ○ 1 – Rack 1 Shelf 1 Bin 1
		  ○ 2 – Rack 1 Shelf 1 Bin 1
		• User select 1
		
		Note that in use case 3, we added 2 units of MASK to Rack 1 Shelf 1 Bin 1.

5	Query Inventory Item (1 mark)	• Display information about an inventory item. • The available inventory units and their corresponding storage bin should also be displayed. Sample Input: • SKU Code – MASK Sample Output: • SKU Code – MASK • Name – COVID-19 3-Ply Face Mask (100 Boxes) • Dimensions (LxWxD) – 100 cm x 100 cm x 100 cm • Computed Volumetric Space for each Unit – 1,000,000 cm^3 • Weight – 20 kg • Quantity On Hand – 1 ○ 1 – Rack 1 Shelf 1 Bin 1 Note that in use case 3, we added 2 units of MASK to Rack 1 Shelf 1 Bin 1 and in use case 4 we removed 1 unit of MASK.

S/N	Use Case	Use Case Description/Business Rules
6	Query Storage Bin (1 mark)	• Display information about a storage bin. • The available inventory items currently in the storage bin should also be displayed. Sample Input: • Rack – 1 • Shelf – 1 • Bin – 1 Sample Output: • Dimensions (LxWxD) – 200 cm x 200 cm x 200 cm • Computed Volumetric Space – 8,000,000 cm^3 • Load Capacity – 50 kg • Inventory Items: ○ 1 – MASK Note that in use case 3, we added 2 units of MASK to Rack 1 Shelf 1 Bin 1 and in use case 4 we removed 1 unit of MASK.

Deliverable Submission

The assignment deliverable to be submitted to the Canvas Assignment tool are to be placed in a single zip archive file with the following folders structure:

- **source** subfolder containing:
  - All Python source files that constitute your program.
  - The main source file containing the program entry point should be named as pywarehouse.py, i.e., your program should be runnable with the command `python pywarehouse.py`

Upload this zip archive file to the designated Canvas Assignment: Individual Assignment 2.

Your deliverables must be submitted latest by **Sunday, 26 November 2023, 11:59 pm**. No assignment will be accepted for assessment after this date/time and you will be awarded **0** marks.

-- End of Assignment Specification --