**Project Report**

1. **Introduction**

This Project has several warehouses located all around warehouses, each warehouse has different capacity.

However, there is no system connected all the warehouse for the smart system. We decided to design the program to make the warehouse system an automatic storage system by following the requirement design a program such that.

- When received an order, a robot will pick up an item from a warehouse and transfer.

- When received an order, the belt will output 1 item at a time.

- When received a command, a robot will store an item at a specific location.

Based on these requirements, we build a lot of storage space. It will support product relocation between warehouses. It is packed into the belt to move. And these program can store products from the outside to the warehouse or to remove items from the warehouse easily. So you can find the exact position of the product that you want to.

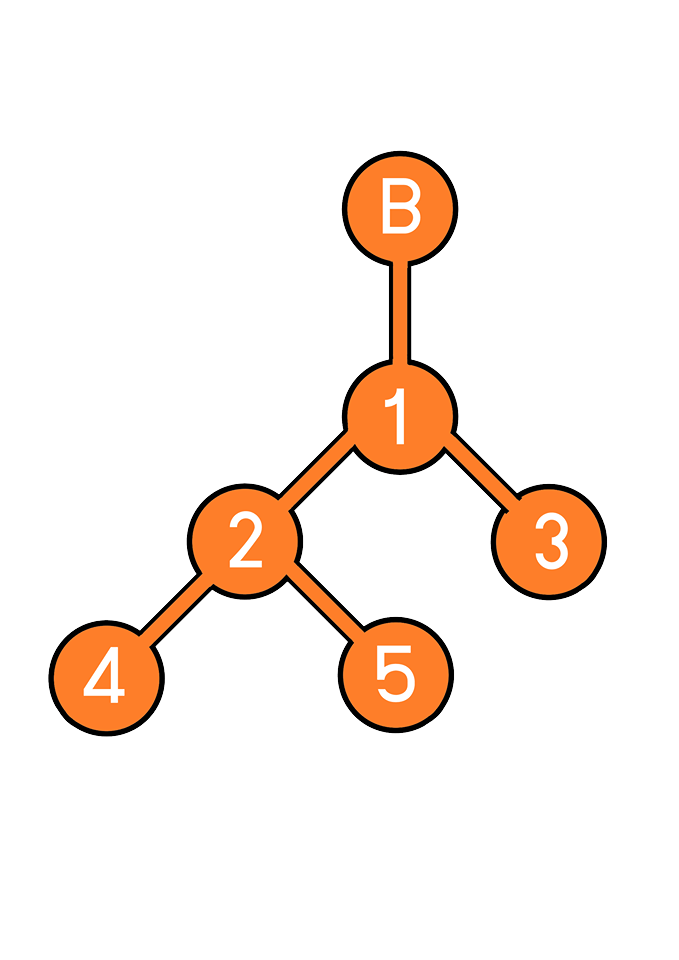
1. **Requirement Analysis**

**Functional Requirement and specification**

From the overview we will be divided into three parts: belt, warehouses, input & output Command

1. **Warehouse Specifications requirement**

A row is a 2-dimensional grid, each space in a grid is used to store an item, each warehouse has a robot to pick up and store items. There are 5 warehouses, warehouse1 connects with a **conveyor belt**, Warehouse 2 and Warehouse 3, Warehouse 2 connects with Warehouse 4 and 5, and There is a robot running around to transfer items from a warehouse to a conveyor belt. The conveyor belt can hold up to 10 items. All warehouses can storage 9675 products. Warehouse 1, 2, and 3 have 5 rows of 10x10 grid. Can storage 500x3 = 1500 products. Warehouse 4 has 7 rows of 5x5 grid. Can storage 175 products. Warehouse 5 has 20 rows of 20x20 grid. Can storage 8000 products. - When received an order, a robot will pick up an item from a warehouse and transfer. - When received an order, the belt will output 1 item at a time. - When received a command, a robot will store an item at a specific location.

1. **Belt**

**2.2 Product ID requirement**

Each product has a unique id in a form of 4 characters: x y z

* x represents a type of the item. It has a value of A to Y.
* y represents a row number of the item. It has a value of 1 to 5.
* z represents a row number of the item. It has a value of 00 to 99.

**2.3 Input Command requirement**

There are several commands we can give to the system. The commands have following formats.

* 0XXXX Retrieve a product id XXXX
* 1XXXX Store a product id XXXX
* 2XY00 Sort Warehouse X at row Y
* 30000 Retrieve a product from the conveyor belt
* 40000 Output information of all warehouses
* 5XXXX Search for a product ID XXXX
* 9XXXXYYYY Manually put a product id XXXX at position YYYY

**Non-Functional Requirement**

**Output command requirement - Retrieving command: 0XXXX**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Retrieving Successfully.

**If belt is full, the system should print out following statements:**

* Belt is full. Cannot retrieve the product

**If slot is empty, the system should print out following statements:**

* Slot is empty. Cannot retrieve the product.

If the product is still in the belt, the product can’t be retrieved. The system will display that.

* now product XXXX is on belt.

**- Storing command: 1XXXX**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Moving from Belt to A
* Moving from A to C
* Storing a product id XXXX in warehouse C: row y slot z
* Moving from C to A
* Moving from A to Start
* Storing Successfully!

**If slot is unavailable, the system should print out following statements in this order:**

* Slot is occupied. Can’t store the product.

If the product is already stored and still in the warehouse, the system will display.

o product has been stored.

If the product is still in the belt, it can’t be stored.

o now product XXXX is on belt.

**- Sorting command: 2XY00**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Sorting process for warehouse A is complete.

**- Retrieving from the belt command: 30000**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Retrieve a product with id XXXX from the belt.
* The belt now has x products on the line.

**If there is nothing on the belt, the system should print out following statements in this order:**

* The belt is empty. Cannot retrieve the product from the belt.

**- Warehouse Information: 40000**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Warehouse A
* Numbers of Rows: 5
* Numbers of total products: 8
* Product in row 1: id A100, C108, E111
* Product in row 2: id –
* Product in row 3: id L355
* Product in row 4: id Q450
* Product in row 5: id U500, W501,

**- Searching for a product: 5XXXX**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Found the product at XXXX.

**If the system cannot find the product, the system should print out following statements in this order:**

* Product not found.

**- Manually moving the product: 9XXXXYYYY**

**If the system can operate successfully, the system should print out the following statements in this order:**

* Move product XXXX to YYYY

**If the slot is occupied, the system should print out following statements in this order:**

* Slot is occupied. Failed to move.

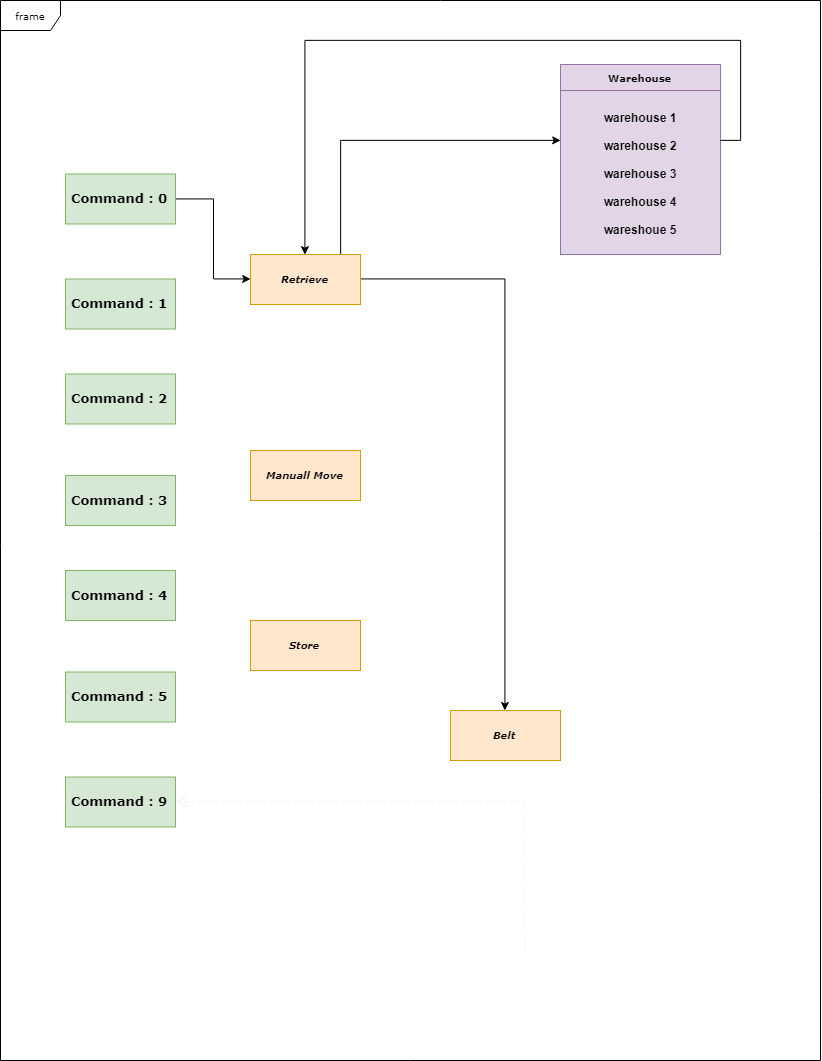
If the product is still in the belt, the product can’t be moved.

* o now product XXXX is on belt.

**If function 9XXXXYYYY only has XXXXX but not has YYYY it will not work.**

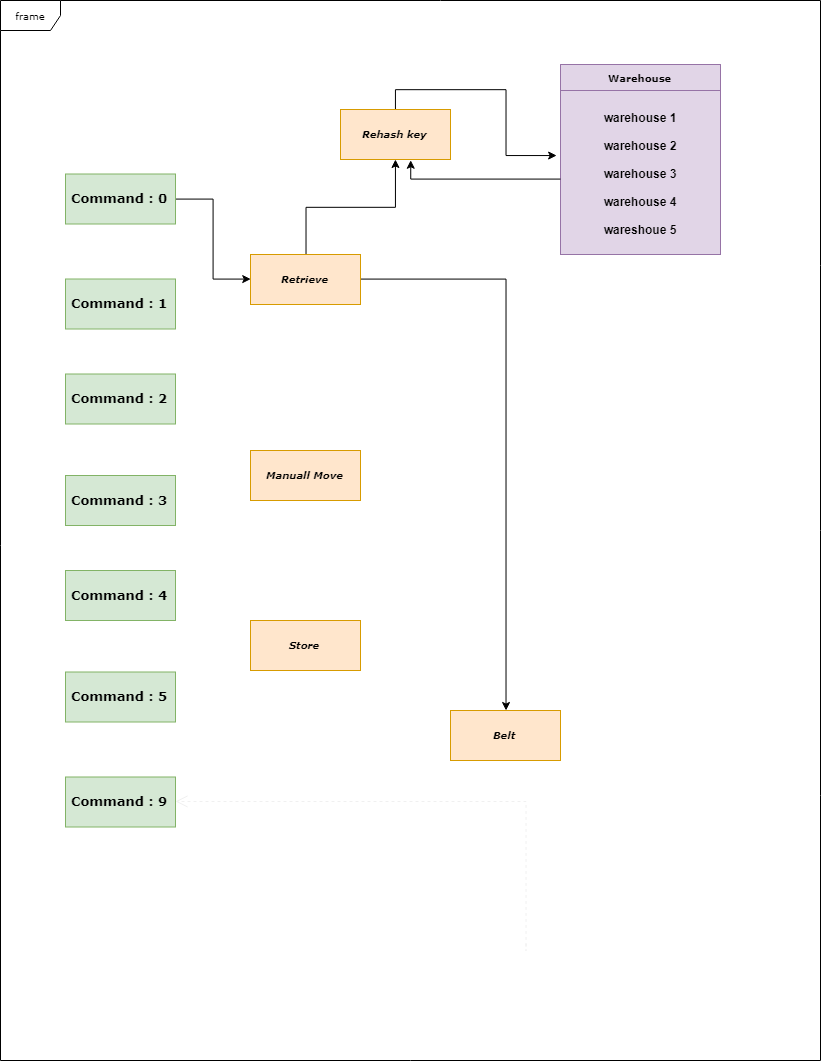
* **Error: please input positon YYYY**

**Rehash key**



(Normal way)

Normally when we need find some of product. We will need to find every warehouse by warehouse to find it.



(Rehash)

(Concept how to use rehash to retrieving product and storage)

It’s key to solve how to searching, re-check, retrieving, storage, Manually

but if we use Rehash we build key shortcut to find all the warehouse by searching in key shortcut in easily and faster than normally way.

Key shortcut it will store the current product id + position. each warehouse is a grid map to find it more convenient to search where it is, (order of one)

4. Offers

-Lower case If the product name is given by a to y, the system will change to A to Y.

It is required that characters 2 and 6 must be English characters. The rest must be numbers ,by follow product id requirement .

               - If the command entered is not equal to 5 or 9, the system will be warned.

"Your order is not clear please input again"

-Function hashing\_key in the class Robot is a function to enter and decode. The product ID A100-Y599 is changed to 32600-45099, then the product id should be located in the warehouse, whichever slot, and also find out what position to store product id and store it. dictionary, because it wants to search product id as search as O (1)

**Extra Code:**

**InputComman: 60000**

Show all Product key in all warehouse

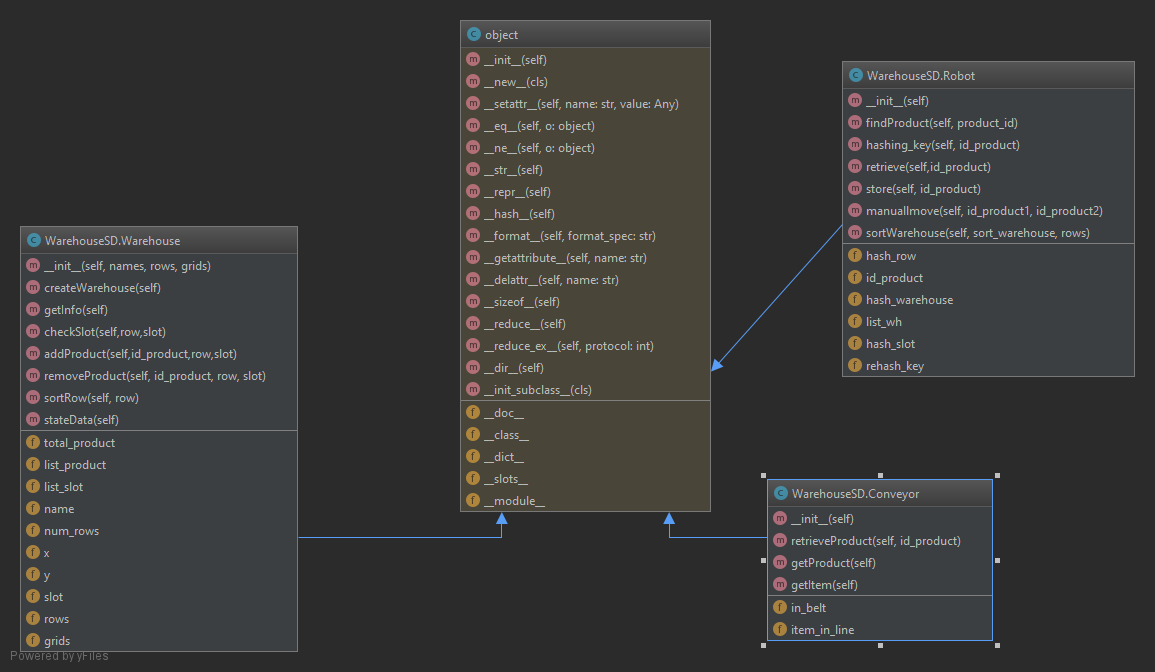
**InputComman: 70000**

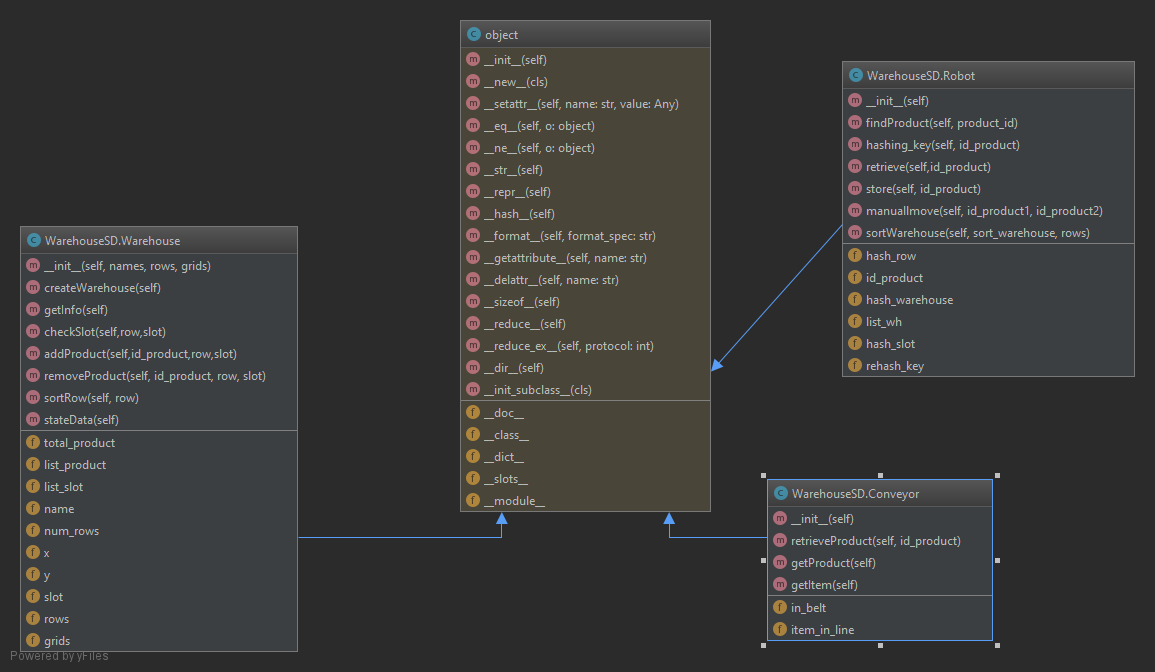
Show in side of all warehouse

**Design**

**Design class warehouse**

Relation between class\_warehouse





Class Conveyor

Warehouse.createWarehouse ()

    confirm statement to create the warehouse.

Warehouse.getInfo ()

         View the warehouse information.

Warehouse.checkSlot (row, slot)

    View the location in the warehouse to see if the slot in the row is empty.

 Warehouse.addProduct (id\_product, row, slot)

    Put the product into the warehouse as a row slot.

 Warehouse.removeProduct (id\_product, row, slot)

         Remove the product from the warehouse as a row slot.

Warehouse.sortRow (row)

 Take the product in the row and sort it out so that the product in the row goes back to where it should be.

Warehouse.stateData ()

         Show all the warehouse

Class Robot

Robot.findProduct (product\_id)

 Is a function where the product id is located.

Robot.hashing\_key (id\_product)

 A function that encodes and decodes to find the position or product id.

Robot.retrieve (id\_product)

 Function to remove product id from the warehouse.

Robot.store (id\_product)

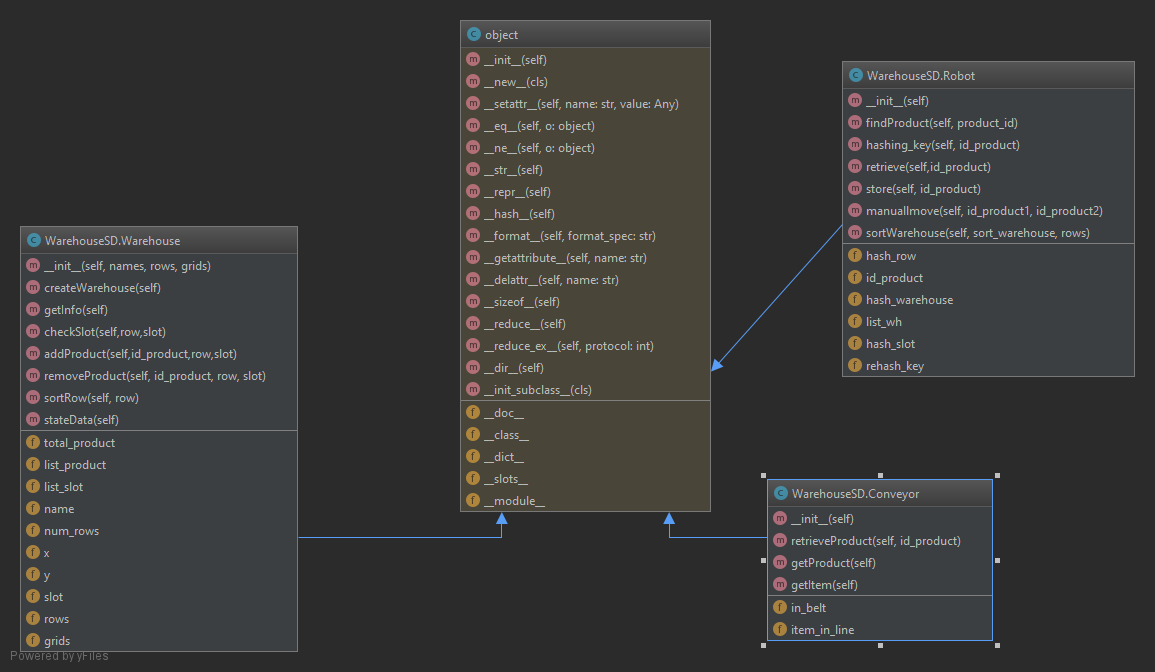
 It's a function for putting a product into the warehouse where it should be.

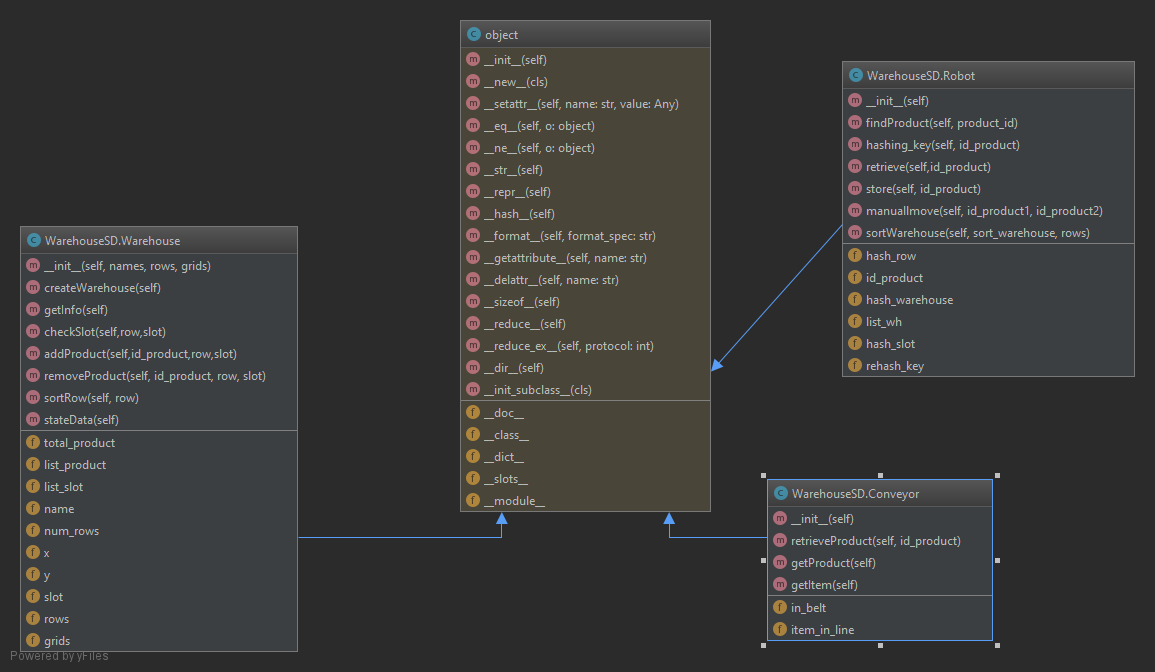
Robot.manuallmove

 It is a function to move a product that exists in the warehouse. If it is not, it will be added to it.

Robot.sortWarehouse

 The function is to sort the required warehouse by specifying the row that you want to sort by the product in the row, through the sortRow of the Warehous class.





Class Conveyor

Conveyor.retrieveProduct (id\_product)

     Bring product id back to the belt.

Conveyor.getProduct ()

  Take the product out of the belt as a first come first serve basis.

Conveyor.getItem ()

         Browse product number in belt

1XXXX Store product

Check hashing\_key[id\_product]. Check product on belt. If belt has product it will print “product XXXX is on belt”. If product has position it will print “now: product has been stored”. Check slot in warehouse. If row and slot in warehouse is occupied it will print “Slot is occupied” but if row and slot is not in warehouse it will add product by Warehouse.add() function and rehash id product.

1XXXX

hashing\_key(id\_product)

warehouse

row

slot

If product on belt

print “now: product XXXX is on belt.”

If rehash\_key(id\_product) != “ ”

print “product has been stored”

0XXXX Retrive product

Check slot in warehouse

Warehouse.checkSlot(raw,slot)

If != “ ”:

print “slot is occupied”

else:

warehouse.add Product (id\_product,row,slot)

rehash\_key [id\_product]=[id\_product]

Check rehash\_key[id\_product]. If empty print “slot is empty. Cannot retrieve the product” else hashing\_key(from position\_new of product) because we don’t know where is it now remove product from warehouse to belt. New product on the belt (use function belt.retrieve(product\_id)).

|  |  |
| --- | --- |
| rehashkey[Id\_product] | position\_new |
| A103 | Y595 |
| A104 | “ ” |
| … | … |
| Y595 | A104 |

0XXXX

Y595

Robot take product to belt

rehash\_key[id\_product] = “Belt”

belt.retrieve(id\_product)

Warehouse.removeProduct(id\_product,row,slot)

hashing\_key (positon\_new) or (product\_id)

warehouse

row

slot

30000

Check product in belt if count of product in belt = 0 print “The belt is empty” else print “Retrievefirst product from belt” rehash\_key[first product in belt] = “” . Delete first product in belt. delete 1 count of product and print “count of product in line”

if self.item\_in\_line == 0:

print (The belt is empty)

else:

print ("Retrieve” +self.in\_belt[0])

r.rehash\_key[self.in\_belt[0]] = " "  
 del self.in\_belt[0]  
 self.item\_in\_line -= 1

print (“the belt now has”+ str(self.item\_in\_line))

3000

5XXXX search of product

Check if r.rehash\_key[product\_id] not equal empty print “position of product” else print **“not found”**

if r.rehash\_key[product\_id1] != " ":  
 print("Found the product at " +r.rehash\_key[product\_id1])  
else:  
 print("Product not found")

5XXXX

**2XYY00**

X is the warehouse you want to sort with names A, B, C, D, E.

YY is the row that needs to be sorted by a value between 1 and the number of rows in the warehouse, given by two numbers. If the sort row in the warehouse is successful, the system will display "Sorting process for warehouse A is complete." If one of the rows does not return to the slot, the product will not be returned to the original slot. "Slot is occupied. Fail to sort."

X will change from A, B, C, D, E to

1,2,3,4,5, respectively.

YY is the row that needs to be sorted by a value between 1 and the number of rows in the warehouse, given by two numbers.

sortWarehouse(sort\_warehouse, rows)

**2XYY00**

Use Warehouse.sortRow (rows) to retrieve the list of products in the row, in the form sorted by product id.

If the sort row in the warehouse is successful, the system will display "Sorting process for warehouse A is complete."

If one of the rows is not returned to the channel, the product will not be returned to the original position. "Slot is occupied. Fail to sort."

9XXXXYYYY

This command enters the function manuallmove (id\_product1, id\_product2), where id\_product1 is the product id that needs to be moved. The id\_product2 part is the position to move. The procedure is to put the position into the hashing\_key, then check to see if the position is in the warehouse row slot empty, and then put the product id into the hashing\_key function to see if the product id has been inserted. And not in the belt.

If the product can be moved to the desired position, the system will display "Move product XXXX to YYYY".

If the position is moved, the system will display "Slot is occupied. Failed to move."

If the product is still in the belt, it will not be able to move the product. The system will display "now product XXXX is on belt."

If the product is not inserted, it can’t be moved because there is no product. The system displays "Slot is empty. Can’t retrieve the product.

If the product is still in the belt, it will not be able to move the product. The system will display "now product XXXX is on belt."

If the product is not inserted, it can’t be moved because there is no product. The system displays "Slot is empty. Can’t retrieve the product.

self.hashing\_key(id\_product2)

manuallmove(self, id\_product1,product2)

This command enters the function manuallmove (id\_product1, id\_product2), where id\_product1 is the product id that needs to be moved. The id\_product2 part is the position to move

9XXXXYYYY

**40000**   
The system will display the data in every warehouse. It will display the names of the number of rows, the number of items in each row, through the function getInfoAll () to display all the warehouse data.

getInfoAll()

40000

Warehouse B  
 Numbers of Rows: 5   
 Numbers of total products: 0   
 Product in row 1: id -   
 Product in row 2: id -  
 Product in row 3: id -  
 Product in row 4: id -  
 Product in row 5: id -

Warehouse C  
 Numbers of Rows: 5   
 Numbers of total products: 0   
 Product in row 1: id -   
 Product in row 2: id -  
 Product in row 3: id -  
 Product in row 4: id -  
 Product in row 5: id -

Warehouse E  
 Numbers of Rows: 20   
 Numbers of total products: 0   
 Product in row 1: id -   
 Product in row 2: id -  
 Product in row 3: id -  
 Product in row 4: id -  
 Product in row 5: id -   
 Product in row 6: id -  
 Product in row 7: id -  
 Product in row 8: id -   
 Product in row 9: id -  
 Product in row 10: id -  
 Product in row 11: id -   
 Product in row 12: id -  
 Product in row 13: id -  
 Product in row 14: id -  
 Product in row 15: id -   
 Product in row 16: id -  
 Product in row 17: id -  
 Product in row 18: id -   
 Product in row 19: id -  
 Product in row 20: id -  
 Product in row 7: id -

Warehouse D  
 Numbers of Rows: 7   
 Numbers of total products: 0   
 Product in row 1: id -   
 Product in row 2: id -  
 Product in row 3: id -  
 Product in row 4: id -  
 Product in row 5: id -   
 Product in row 6: id -  
 Product in row 7: id -

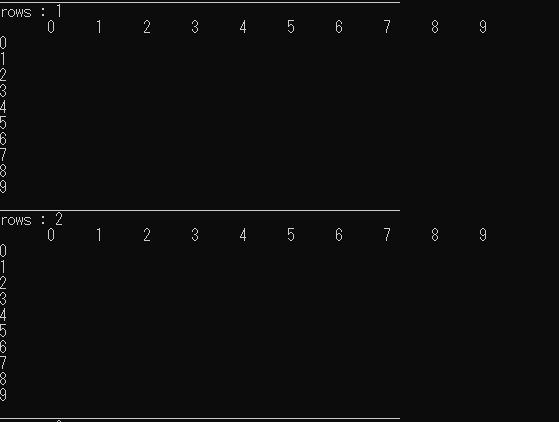
Warehouse A  
 Numbers of Rows: 5   
 Numbers of total products: 0   
 Product in row 1: id -   
 Product in row 2: id -  
 Product in row 3: id -  
 Product in row 4: id -  
 Product in row 5: id -

**Design in Warehouse**

All warehouses can storage 9675 products. (Example extra code 70000)

**Empty warehouse**







**Sort product to warehouse**

