**Source Code:**

**Store.py:**

*"""  
ITMD 513 : Assignment-Polymorphism - Bhanu kureti  
Summary : This program will create Stores abstract class with attributes Name, Address, Status and Sales\_tax\_percent  
"""*# Define abstract class store  
class Store:  
  
 # constructor  
 def \_\_init\_\_(self, s\_name, s\_address, s\_status, s\_sales\_tax\_percentage):  
 self.s\_name = s\_name  
 self.s\_address = s\_address  
 self.s\_status = s\_status  
 self.s\_sales\_tax\_percent = s\_sales\_tax\_percentage  
  
 # mutators of store class  
 def set\_s\_name(self, s\_name):  
 self.s\_name = s\_name  
  
 def set\_s\_address(self, s\_address):  
 self.s\_address = s\_address  
  
 def set\_s\_status(self, s\_status):  
 self.s\_status = s\_status  
  
 def set\_s\_sales\_tax\_percentage(self, s\_sales\_tax\_percentage):  
 self.s\_sales\_tax\_percentage = s\_sales\_tax\_percentage  
  
 # accessors of store class  
 def get\_s\_name(self):  
 return self.s\_name  
  
 def get\_s\_address(self):  
 return self.s\_address  
  
 def get\_s\_status(self):  
 return self.s\_status  
  
 def get\_s\_sales\_tax\_percentage(self):  
 return self.s\_sales\_tax\_percent  
  
 # returns true if store is open else false  
 def is\_store\_open(self):  
 if (status.lower() == "open"):  
 return True  
  
 return False  
  
 # abstract function to calculate total sales tax  
 def calculate\_total\_sales\_tax(self):  
 pass  
  
 # abstract function to calculate total sales  
 def calculate\_total\_sales(self):  
 pass

**Restaurant.py:**

*"""  
ITMD 513 : Assignment- Polymorphism - Bhanu kureti  
Summary : This program will create Restaurant class with attributes current\_occupancy,  
maximum occupancy, price\_per\_person  
"""*# Importing Store class from Store.py program  
from Store import Store  
  
  
class Restaurant(Store):  
  
 # constructor  
 def \_\_init\_\_(self, s\_name, s\_address, s\_status, s\_sales\_tax\_percentage, r\_max\_occupancy, r\_price\_per\_person):  
  
 super().\_\_init\_\_(s\_name, s\_address, s\_status, s\_sales\_tax\_percentage)  
 self.r\_max\_occupancy = r\_max\_occupancy  
 self.r\_total\_num\_served = 0  
 self.r\_current\_occupancy = 0  
 self.r\_price\_per\_person = r\_price\_per\_person  
  
 # function will take number of people to be seated as input and returns true if the people can be seated else false  
 def seat\_patrons(self, num\_of\_people):  
 # if current occupancy + num of people less that maximum occupancy, then it returns True  
 if ((self.r\_current\_occupancy + num\_of\_people) <= self.r\_max\_occupancy):  
 print("Welcome to %s" % (self.get\_s\_name()))  
 self.r\_current\_occupancy += num\_of\_people  
 return True  
 else: # if num of people cannot be seated, return false  
 print("We are at capacity, we appreciate your patience")  
 return False  
  
 # function will take number of people to be serve as input and returns the number of people served  
 def serve\_patrons(self, num\_of\_people):  
 self.r\_total\_num\_served += num\_of\_people # add num\_people to num\_served and return the updated value  
 return self.r\_total\_num\_served  
  
 # function will take number of people leaving the restaurant and returns the current\_occupancy  
 def checkout\_patrons(self, num\_of\_people):  
 self.r\_current\_occupancy -= num\_of\_people # remove the num\_people from current\_occupancy  
 return self.r\_current\_occupancy # return the updated current\_occupancy  
  
 # mutator of restaurant class  
 def set\_r\_price\_per\_person(self, r\_price\_per\_person):  
 self.r\_price\_per\_person = r\_price\_per\_person  
  
 # accessor of restaurant class  
 def get\_r\_price\_per\_person(self):  
 return self.r\_price\_per\_person  
  
 # calculate and return the total sales tax  
 def calculate\_total\_sales\_tax(self):  
 return float(self.r\_total\_num\_served \* self.r\_price\_per\_person \* self.get\_s\_sales\_tax\_percentage()) / 100  
  
 # calculate and return the total sales  
 def calculate\_total\_sales(self):  
 return float(self.r\_total\_num\_served \* self.r\_price\_per\_person)

**Grocery\_Store.py:**

*"""  
ITMD 513 : Assignment- Polymorphism - Bhanu kureti  
Summary : This program will create Grocery store class with attributes total revenue, grocery store type  
"""*# Import Store class from Store,py program  
from Store import Store  
  
  
# define GroceryStore class  
class GroceryStore(Store):  
  
 # constructor  
 def \_\_init\_\_(self, s\_name, s\_address, s\_status, s\_sales\_tax\_percentage, g\_store\_type):  
 super().\_\_init\_\_(s\_name, s\_address, s\_status, s\_sales\_tax\_percentage)  
 self.total\_revenue = 0  
 self.g\_store\_type = g\_store\_type  
  
 # function that takes the quantity and price of item to sell as input and returns the total revenue after selling  
 def sell\_item(self, quantity, price\_item):  
 self.total\_revenue += float(quantity) \* price\_item  
 return self.total\_revenue # return the updated total\_revenue  
  
 # mutator of grocery store class  
 def set\_g\_store\_type(self, g\_store\_type):  
 self.g\_store\_type = g\_store\_type  
  
 # accessor of grocery store class  
 def get\_store\_type(self):  
 return self.store\_type  
  
 # calculate and return total sales revenue  
 def calculate\_total\_sales(self):  
 return float(self.total\_revenue)  
  
 # calculate and return total sales tax  
 def calculate\_total\_sales\_tax(self):  
 return float(self.total\_revenue \* self.get\_s\_sales\_tax\_percentage()) / 100

**Shopping.py:**

*"""  
ITMD 513 : Assignment- Polymorphism - Bhanu kureti  
Summary : This shopping.py program will be used to test the restaurant and grocery stores program  
"""*# Importing the Restaurant and GroceryStore classes from restaurant and grocery\_store programs  
from restaurant import Restaurant  
from grocery\_store import GroceryStore  
  
  
# main function where execution starts  
def main():  
 print("\n------Restaurant class Testing------")  
 storeR = Restaurant("CumIn Restaurant", "28, State Street", "open", 6.25, 150, 14.99)  
 print("%s is %s" % (storeR.get\_s\_name(), storeR.get\_s\_status()))  
 print(storeR.seat\_patrons(75))  
 print("Number of people served: %d " % storeR.serve\_patrons(25))  
 print("Current occupancy: %d" % storeR.checkout\_patrons(30))  
 print("Total sales : $%.2f" % (float(storeR.calculate\_total\_sales())))  
 print("Total sales tax: $%.2f" % (float(storeR.calculate\_total\_sales\_tax())))  
 print("\n----Checking when input value more than capacity----")  
 print(storeR.seat\_patrons(185))  
 print("Testing Restaurant class is done\n")  
 print("--------------------------------------------")  
 print("GroceryStore class Testing")  
 storeG = GroceryStore("AllInOne Grocery Store", "2801 S king drive", "open", 7.45, "independent")  
 print("Grocery Store %s is %s" % (storeG.get\_s\_name(), storeG.get\_s\_status()))  
 print("Total revenue of store after selling 4 rice bags @14.75 = $%.2f " % (storeG.sell\_item(4, 14.75)))  
 print("Total revenue of store after selling 6 milk @2.75 = $%.2f" % (storeG.sell\_item(6, 2.75)))  
 print("Total revenue of store after selling 2 Bread @1.75 = $%.2f" % (storeG.sell\_item(2, 1.75)))  
 print("\nTotal sales tax: $%.2f" % (storeG.calculate\_total\_sales\_tax()))  
 print("Total cost of sales with tax: $%.2f" % (storeG.calculate\_total\_sales\_tax() + storeG.calculate\_total\_sales()))  
 print("\n----checking the Status attribute of store ----")  
 storeG.set\_s\_status("closed")  
 print("Grocery Store %s is %s" % (storeG.get\_s\_name(), storeG.get\_s\_status()))  
 print("End of testing GroceryStore class")  
  
  
# call the main function  
main()

**OUTPUT:**

/Users/bhanuchanderkureti/PycharmProjects/python\_projects/venv/bin/python /Users/bhanuchanderkureti/PycharmProjects/python\_projects/shopping.py

**------Restaurant class Testing------**

CumIn Restaurant is open

Welcome to CumIn Restaurant

True

Number of people served: 25

Current occupancy: 45

Total sales: $374.75

Total sales tax: $23.42

**----Checking when input value more than capacity----**

We are at capacity, we appreciate your patience

False

Testing Restaurant class is done

--------------------------------------------

**GroceryStore class Testing**

Grocery Store AllInOne Grocery Store is open

Total revenue of store after selling 4 rice bags @14.75 = $59.00

Total revenue of store after selling 6 milk @2.75 = $75.50

Total revenue of store after selling 2 Bread @1.75 = $79.00

Total sales tax: $5.89

Total cost of sales with tax: $84.89

**----checking the Status attribute of store ----**

Grocery Store AllInOne Grocery Store is closed

End of testing GroceryStore class

Process finished with exit code 0