**EXERCISE NO: 13**  **REGISTER NO:** 1518102099

**DATE:** 23.11.2020

**AIM:**

An apparel shop wants to manage the items which it sells 25 min .Write a python program to implement the class diagram given below.

**Class Description:**

**Apparel class:**

Initialize static variable counter to 100

In the constructor, auto-generate item\_id starting from 101 prefixed by "C" for cotton apparels and "S" for silk apparels. Example – C101, S102, S103, C104 etc.

**calculate\_price():** Add 5% service tax on the price of the apparel and update attribute, price with the new value

**Cotton class:**

While invoking parent constructor from child constructor, pass "Cotton" as item\_type

**calculate\_price():** Update attribute, price of Apparel class based on rules given below

Add service tax on price by invoking appropriate method of Apparel class

Apply discount on price

Add 5% VAT on final price

**Silk class:**

While invoking parent constructor from child constructor, pass "Silk" as item\_type

**calculate\_price():**Update attribute, price of Apparel class based on rules given below

Add service tax on price by invoking appropriate method of Apparel class

**Identify points earned based on rules given below:**

Silk apparels with price more than Rs. 10000, earn 10 points and anything less than or equal to that earn 3 points

Initialize attribute, points with the identified points

Add 10% VAT on price

**Note:** Perform case sensitive string comparison   
 **For testing:**

Create objects of Cotton class and Silk class

Invoke calculate\_price() on Cotton objects and Silk objects

**PROGRAM:**

class Apparel:

counter=100

def \_\_init\_\_(self,price,item\_type):

Apparel.counter+=1

self.\_\_item\_id=item\_type[0]+str(Apparel.counter)

self.\_\_price=price

self.\_\_item\_type=item\_type

def calculate\_price(self):

self.\_\_price+=self.\_\_price\*0.05

def get\_item\_id(self):

return self.\_\_item\_id

def get\_price(self):

return self.\_\_price

def get\_item\_type(self):

return self.\_\_item\_type

def set\_price(self,price):

self.\_\_price=price

return self.\_\_price

class Cotton(Apparel):

def \_\_init\_\_(self,price,discount):

super().\_\_init\_\_(price,'Cotton')

self.\_\_discount=discount

def calculate\_price(self):

super().calculate\_price()

price=self.get\_price()

price-=price\*(self.\_\_discount/100)

price+=price\*0.05

self.set\_price(price)

return price

def get\_discount(self):

return self.\_\_discount

class Silk(Apparel):

def \_\_init\_\_(self,price):

super().\_\_init\_\_(price,"Silk")

self.\_\_points=None

def calculate\_price(self):

super().calculate\_price()

if (self.get\_price()>10000):

self.\_\_points=10

else:

self.\_\_points=3

return self.set\_price(self.get\_price()+(self.get\_price()\*0.1))

def get\_points(self):

return self.\_\_points

silk=int(input())

cotton=int(input())

discount=int(input())

a=Silk(silk)

print(a.calculate\_price())

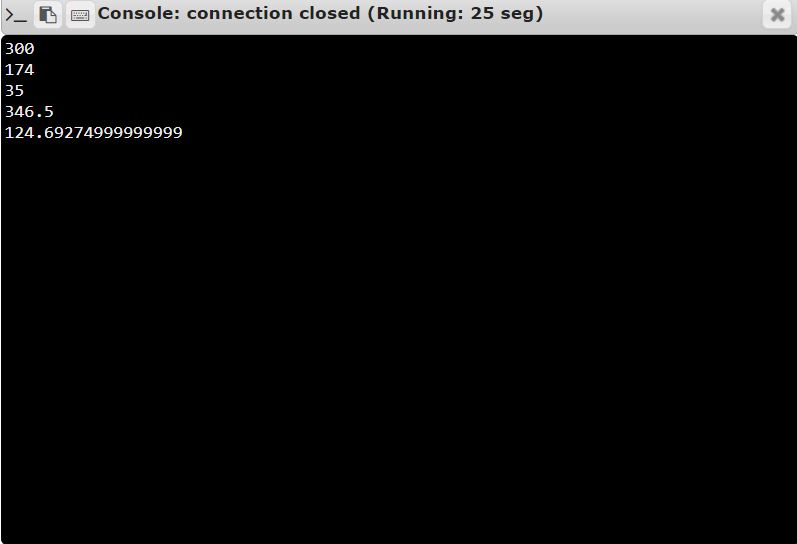
b=Cotton(cotton,discount)

print(b.calculate\_price())

**LINK:**

<http://103.53.53.18/mod/vpl/forms/submissionview.php?id=328&userid=1704>

**OUTPUT:**

****

**RESULT:**

Thus, the code has been successfully executed and output is verified. ­­­­­