**EXPERIMENT NO - 03**

CLASSES AND OBJECTS :-

====

**=3.1=**

==== class novel:

name="UNTOLD STORY" author="CHETAN BHAGAT" price=2000 def getname(self):

print("NAME : ",self.name) @staticmethod def getprice():

print("PRICE : ",novel.price) @classmethod def getauthor(cls):

print("AUTHOR : ",cls.author) class moredetails:

publication="MUMBAI PUBLICATIONS" def getpublication(self):

print("PUBLICATIONS : ",self.publication)

n=novel() n.getname() n.getprice() n.getauthor() x=n.moredetails() x.getpublication()

OUTPUT :

**=3.2= ====**

==== #importing class

import math class square():

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

self.side=side def area(self):

return (self.side\*\*2) def perimeter(self): return 4\*self.side r=int(input("ENTER SIDE OF A SQUARE : ")) obj=square(r) print("AREA OF SQUARE IS :",obj.area()) print("PERIMETER OF SQUARE IS :",obj.perimeter())

OUTPUT :

**=3.3= ====**

==== class A:

def explore(self):

print("EXPLORE() METHOD CALLED") def search(self):

print("POLYMORPHISM")

class B:

def search(self):

print("SEARCH() METHOD CALLED")

class C:

def discover(self):

print("DISCOVER() METHOD CALLED") def discover(self):

print("METHOD OVERLOADING")

class D(A, B, C):

def test(self):

print("TEST() METHOD CALLED") def search(self):

print("METHOD OVERRIDING")

d\_obj = D() obj=A() d\_obj.explore() d\_obj.search() obj.search() d\_obj.discover() d\_obj.test() d\_obj.search()

OUTPUT :

=3.4= =====

==== #ASSERTION ERROR

try:

a=int(input("ENTER THE NUMBER BETWEEN 10 AND 50 : ")) assert a>10 and a<=50 print("ENTER YOUR NUMBER : ",a) except AssertionError:

print("THIS DOES NOT SATISFY THE CONDITION")

OUTPUT :

#NORMAL EXCEPTION ERROR print("INDEX OUT OF BOUND ERROR") try:

b=str("HELLO EVERYONE..") print(b[10]) except LookupError:

print("INDEX OUT OF BOUND ERROR") else:

print("THE WORD IS : ",b[10])

print(" ")

print("ARITHMETIC ERROR") try:

a = 5/0 print(a) except ArithmeticError:

print("THIS DOES NOT SATISFY THE CONDITION") else:

print("ERROR HAVEN'T OCCURED")

print(" ")

print("VALUE ERROR") try:

x=float(input("ENTER A NUMBER : ")) except ValueError:

print("THIS IS VALUE ERROR EXCEPTION")

print(" ")

print("INPUT OUTPUT ERROR") try:

name=input("ENTER NAME OF FILE : ") f= open(name, 'r') except IOError:

print("FILE NOT FOUND : ",name) else:

n= len(f.readlines()) print(name,'HAS',n,'LINES') F.closes

OUTPUT :

#USER DEFINED EXCEPTION class Error(Exception):

"""Base class for other exceptions""" pass

class ValueTooSmallError(Error):

"""Raised when the input value is too small""" pass

class ValueTooLargeError(Error):

"""Raised when the input value is too large"""

pass

num=10

while True:

try:

i\_num = int(input("ENTER A NUMBER : ")) if i\_num < num:

raise ValueTooSmallError elif i\_num > num:

raise ValueTooLargeError break except ValueTooSmallError:

print("THIS VALUE IS TOO SMALL!") except ValueTooLargeError:

print("HIS VALUE IS TOO LARGE!") print("CONGRATULATIONS!!!THIS VALUE IS CORRECT.")

OUTPUT :