**IMPLEMENTATION – EX 6 :**

**Question 1:**

from abc import ABC

from abc import abstractmethod

class Loan(ABC):

def \_\_init\_\_(self, loan\_amount, account\_type, location, borrower\_details):

self.borrower\_details = borrower\_details

self.loan\_amount = loan\_amount

self.account\_type = account\_type

self.location = location

@abstractmethod

def calculate\_interest(self):

...

@abstractmethod

def DisplayDetails(self):

...

@abstractmethod

def MonthlyPaymentInterest(self):

...

@abstractmethod

def MonthlyPaymentTotal(self):

...

class EducationLoan(Loan):

def \_\_init\_\_(self, loan\_amount, account\_type, location, course\_fee, borrower\_details):

super().\_\_init\_\_(loan\_amount, account\_type, location, borrower\_details)

self.course\_fee = course\_fee

def calculate\_interest(self):

if self.location == "urban":

location\_factor = 1

else:

location\_factor = 0.95

if self.account\_type == "Savings":

account\_type\_factor = 1.05

else:

account\_type\_factor = 1

base\_interest\_rate = 0.08

return self.loan\_amount \* base\_interest\_rate \* location\_factor \* account\_type\_factor

def DisplayDetails(self):

print(f"Borrower name is {self.borrower\_details[0]}")

print(f"Borrower age is {self.borrower\_details[1]}")

print(f"Borrower martial status is {self.borrower\_details[2]}")

def MonthlyPaymentInterest(self,years):

interest = self.calculate\_interest()

return interest / (years \* 12)

def MonthlyPaymentTotal(self,years):

interest\_per\_month = self.MonthlyPaymentInterest(years)

loan\_amt\_per\_month = self.loan\_amount / (years \* 12)

return loan\_amt\_per\_month + interest\_per\_month

class HomeLoan(Loan):

def \_\_init\_\_(self, loan\_amount, account\_type, location, borrower\_details):

super().\_\_init\_\_(loan\_amount, account\_type, location, borrower\_details)

def calculate\_interest(self):

base\_interest\_rate = 0.06

location\_factor = 1.02 if self.location == "urban" else 1.0

account\_type\_factor = 1.05 if self.account\_type == "Savings" else 1.0

return self.loan\_amount \* base\_interest\_rate \* location\_factor \* account\_type\_factor

def DisplayDetails(self):

print(f"Borrower name is {self.borrower\_details[0]}")

print(f"Borrower age is {self.borrower\_details[1]}")

print(f"Borrower martial status is {self.borrower\_details[2]}")

def MonthlyPaymentInterest(self,years):

interest = self.calculate\_interest()

return interest / (years \* 12)

def MonthlyPaymentTotal(self,years):

interest\_per\_month = self.MonthlyPaymentInterest(years)

loan\_amt\_per\_month = self.loan\_amount / (years \* 12)

return loan\_amt\_per\_month + interest\_per\_month

class PersonalLoan(Loan):

def \_\_init\_\_(self, loan\_amount, account\_type, location, borrower\_details):

super().\_\_init\_\_(loan\_amount, account\_type, location, borrower\_details)

def calculate\_interest(self):

base\_interest\_rate = 0.1

location\_factor = 1.05 if self.location == "urban" else 0.98

account\_type\_factor = 1.08 if self.account\_type == "Savings" else 1.0

return self.loan\_amount \* base\_interest\_rate \* location\_factor \* account\_type\_factor

def DisplayDetails(self):

print(f"Borrower name is {self.borrower\_details[0]}")

print(f"Borrower age is {self.borrower\_details[1]}")

print(f"Borrower martial status is {self.borrower\_details[2]}")

def MonthlyPaymentInterest(self,years):

interest = self.calculate\_interest()

return interest / (years \* 12)

def MonthlyPaymentTotal(self,years):

interest\_per\_month = self.MonthlyPaymentInterest(years)

loan\_amt\_per\_month = self.loan\_amount / (years \* 12)

return loan\_amt\_per\_month + interest\_per\_month

#driver code

if \_\_name\_\_ == '\_\_main\_\_':

#The code provided here will not be executed when imported

try:

education\_loan = EducationLoan(100000, "Savings", "urban", 8000, ["Ram",19,"Unmarried"])

home\_loan = HomeLoan(500000, "Current", "rural",["Vivek",50,"Married"])

personal\_loan = PersonalLoan(200000, "Savings", "urban",["Nikhil",28,"Married"])

education\_loan.DisplayDetails()

print()

home\_loan.DisplayDetails()

print()

personal\_loan.DisplayDetails()

print()

print(f"Education Loan Interest:{education\_loan.calculate\_interest()}")

print()

print(f"Home Loan Interest:{home\_loan.calculate\_interest()}")

print()

print(f"Personal Loan Interest:{personal\_loan.calculate\_interest()}")

print()

print(f"Education Loan Payment interest per month for 5 years : {education\_loan.MonthlyPaymentInterest(5)}")

print()

print(f"Home Loan Payment interest per month for 5 years : {home\_loan.MonthlyPaymentInterest(5)}")

print()

print(f"Personal Loan Interest per month for 5 years : {personal\_loan.MonthlyPaymentInterest(5)}")

print()

print(f"Education Loan total payment per month for 5 years : {education\_loan.MonthlyPaymentTotal(5)}")

print()

print(f"Home Loan total payment per month for 5 years : {home\_loan.MonthlyPaymentTotal(5)}")

print()

print(f"Personal Loan total payment per month for 5 years : {personal\_loan.MonthlyPaymentTotal(5)}")

print()

except Exception as e:

print("Error:", str(e))

**OUTPUT:**

**Borrower name is Ram**

**Borrower age is 19**

**Borrower martial status is Unmarried**

**Borrower name is Vivek**

**Borrower age is 50**

**Borrower martial status is Married**

**Borrower name is Nikhil**

**Borrower age is 28**

**Borrower martial status is Married**

**Education Loan Interest:8400.0**

**Home Loan Interest:30000.0**

**Personal Loan Interest:22680.0**

**Education Loan Payment interest per month for 5 years : 140.0**

**Home Loan Payment interest per month for 5 years : 500.0**

**Personal Loan Interest per month for 5 years : 378.0**

**Education Loan total payment per month for 5 years : 1806.6666666666667**

**Home Loan total payment per month for 5 years : 8833.333333333334**

**Personal Loan total payment per month for 5 years : 3711.3333333333335**

**Question 2:**

import os

def SecureFileReader(fname,fpath):

try:

found = False

for dir\_path, dir\_names, file\_names in os.walk(fpath):

if fname in file\_names:

found = True

file=open(fname.txt, "r")

a = file.readlines()

for i in a:

print(i)

if not found:

raise FileNotFoundError("File does not exist.")

except FileNotFoundError:

print("File not found")

except PermissionError:

print("Required permissions not met")

except Exception as e:

print("An error occurred while reading the file.")

else:

print("File contents have been successfully printed.\n")

#driver code

SecureFileReader("Movie.py",r"Z:\Programming and Design Patterns")

print()

SecureFileReader("RandomFileDoesNotExist.txt",r"Z:\Programming and Design Patterns")

print()

SecureFileReader("textfile.txt",r"Z:\Programming and Design Patterns")

print()

**OUTPUT:**

**Required permissions not met**

**File not found**

**HI**

**HOW**

**ARE YOU**

**Question 3:**

class Calculator:

def add(self,a,b):

c=a+b

return f"Addition of {a} and {b} is {c}"

def subract(self,a,b):

d=a-b

return f"Subraction of {a} and {b} is {d}"

def multiply(self,a,b):

e=a\*b

return f"Multiplication of {a} and {b} is {e}"

def divide(self,a,b):

if b==0:

raise ZeroDivisionError("Division by Zero is not Allowed")

else:

f=a/b

return f"Division of {a} and {b} is {f}"

c=Calculator()

a=input("Enter Number1: ")

b=input("Enter Number2: ")

try:

a=float(a)

b=float(b)

operation=input("Enter Operation: (Add, Sub, Mul, Div): ")

if not ((isinstance(a ,int) or isinstance(b ,float)) and (isinstance(a ,float) or isinstance(b ,int))):

raise TypeError("Both a and b must be Integer or Float")

if operation=="Add":

print(c.add(a,b))

elif operation=="Sub":

print(c.subract(a,b))

elif operation=="Mul":

print(c.multiply(a,b))

elif operation=="Div":

print(c.divide(a,b))

else:

raise ValueError("Invalid Operation, Please Enter Valid Operation like (Add, Sub, Mul, Div)")

except (ZeroDivisionError,ValueError,TypeError) as error:

print(error)

**OUTPUT:**

**Enter Number1: 5**

**Enter Number2: 0**

**Enter Operation: (Add, Sub, Mul, Div): Div**

**Division by Zero is not Allowed**

**Enter Number1: 3**

**Enter Number2: g**

**could not convert string to float: 'g'**

**Enter Number1: 7**

**Enter Number2: 4**

**Enter Operation: (Add, Sub, Mul, Div): square root**

**Invalid Operation, Please Enter Valid Operation like (Add, Sub, Mul, Div)**