#

class Employee: #parent class

def \_\_init\_\_(self,name,salary):

self.name=name

self.salary=salary

def displayDetails(self):

print(f"Name: {self.name}\nSalary: {self.salary}")

class Manager(Employee): #child class

def \_\_init\_\_(self,name,salary,dept):

super().\_\_init\_\_(name,salary)

self.dept=dept

def printDetails(self):

self.displayDetails()

print(f"Department: {self.dept}")

m=Manager('Divya',70000,'AI') #object creation

m.printDetails()

class Libraryitem:

def \_\_init\_\_(self,title,author,publication\_year):

self.title=title

self.author=author

self.publication\_year=publication\_year

def display(self):

print(f"Title: {self.title}\nAuthor: {self.author}\nPublished Year: {self.publication\_year}")

class book(Libraryitem):

def \_\_init\_\_(self,title,author,publication\_year,genre):

super().\_\_init\_\_(title,author,publication\_year)

self.genre=genre

def displayDetails(self):

self.display()

print(f"Genre: {self.genre}")

b=book('To Kill a Mockingbird','Harper Lee',1960,'Fiction')

b.displayDetails()

#

class Bankaccount:

def \_\_init\_\_(self,balance=0):

self.balance=balance

def deposit(self,amount):

if amount>0:

self.balance+=amount

print(f"Amount Deposited: {amount}")

else:

print("Amount to deposit should be in positive")

def withdraw(self,amount):

if amount>self.balance:

print("Insufficient Funds.......")

elif amount<0:

print("Withdraw amount should be positive")

else:

self.balance-=amount

print(f"Amount Withdraw: {amount}")

def check\_balance(self):

print(f"Balance: {self.balance}")

my\_account=Bankaccount()

my\_account.deposit(50000)

my\_account.withdraw(3000)

my\_account.check\_balance()