**Problem Statement**​

Company ABC has recently hired a new employee named Alex Bob. As the Salesforce developer, you are tasked with printing the details for Alex.​

​

**Requirements**​

* Employee Name​
* Employee Salary​
* Employee Number​
* Employee Joining Date​
* Is Employee being Experienced?

public class Employee {

public String employeeName;

public Double employeeSalary;

public String employeeNumber;

public Date employeeJoiningDate;

public Boolean isExperienced;

public Employee(String name, Decimal salary, String empNumber, Date joinDate, Boolean experienced) {

this.employeeName = name;

this.employeeSalary = salary;

this.employeeNumber = empNumber;

this.employeeJoiningDate = joinDate;

this.isExperienced = experienced;

}

public void printEmployeeDetails() {

System.debug('Employee Name: ' + employeeName);

System.debug('Employee Salary: ' + employeeSalary);

System.debug('Employee Number: ' + employeeNumber);

System.debug('Employee Joining Date: ' + employeeJoiningDate);

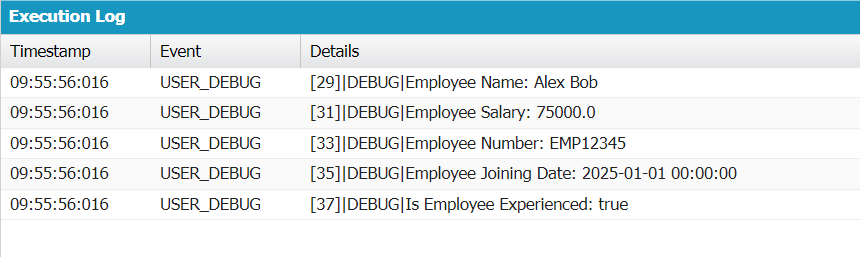
System.debug('Is Employee Experienced: ' + isExperienced);

}

}

Employee alexBob = new Employee('Alex Bob', 75000.00, 'EMP12345', Date.newInstance(2025, 1, 1), true);

alexBob.printEmployeeDetails();



**Problem Statement**​

You are tasked with designing an inventory management system for a small retail store. The store sells various products, including electronics, clothing, and accessories. The current system lacks organization, and the store owner wants to improve efficiency.​

​

**Requirements**​

You need to complete the below job as part of the given task:​

* Add new products to the catalogue (List).​
* Add categories (ex: Electronics, Clothing) (Set).​
* Adding new products with initial quantities (Map).​

public class Inventary {

public void catalogue() {

list<string> products = new list<string>();

products.add('Headphones');

products.add('Speakers');

products.add('Jacket');

products.add('Dress');

system.debug(products);

}

public void categories() {

set<string> st = new set<string>();

st.add('Electronics');

st.add('Clothing');

st.add('Accessories');

st.add('Home');

system.debug(st);

}

public void quantities() {

map<string, integer> mp = new map<string, integer>();

mp.put('Electronics', 40);

mp.put('Clothing', 100);

mp.put('Accessories', 60);

mp.put('Home', 40);

system.debug(mp);

}

}

Inventary i = new Inventary();  
i.catalogue();  
i.categories();  
i.quantities();

A screenshot of a computer

Description automatically generated

**Problem Statement**​

Create a CustomerOrder class that encapsulates order-related information, such as items, number of items and total amount.​

Implement methods to add items, number of items and to calculate the total.​

​

**Properties:**​

* items: A private string representing the name of the ordered item.​
* totalItems: A private integer representing the quantity of the ordered item.​
* totalAmount: A private decimal value representing the total amount for the order.​
* ​

**Methods:**​

* setItem(String itemName, Integer quantity, Decimal pricePerUnit).​
* getCalculateTotal(): A public method that retrieves the total amount for the order. It returns the calculated total amount (decimal).​

public class CustomerOrder {

private String items;

private Integer totalItems;

private Decimal totalAmount;

public CustomerOrder() {

this.totalItems = 0;

this.totalAmount = 0.00;

this.items = '';

}

public void setItem(String itemName, Integer quantity, Decimal pricePerUnit) {

this.items = itemName;

this.totalItems = quantity;

this.totalAmount = quantity \* pricePerUnit;

}

public Decimal getCalculateTotal() {

return this.totalAmount;

}

public Integer getTotalItems() {

return this.totalItems;

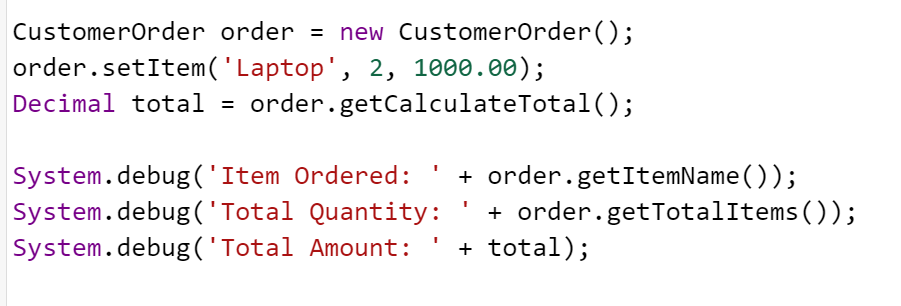
}

public String getItemName() {

return this.items;

}

}



A screenshot of a computer

Description automatically generated

​

**Problem Statement**​

Design a simple banking application that models bank accounts and savings accounts. The system should allow users to create bank accounts, deposit money, and calculate interest for savings accounts.​

​

**Requirements:**​

BankAccount (Base Class): Set Properties as accountNumber, balance, ownerName, amount.​

* Implement a method called accountDetails(Integer accNumber, Decimal initialBalance, String owner, Decimal depositAmount).​
* Define a method called totalAmount().​

​

SavingsAccount (Subclass): ​

* Implement a method called calculateInterest(Decimal interest): This method calculates the interest earned based on the total amount and the provided interest rate.​

public virtual class BankAccount {

private Integer accountNumber;

private Decimal balance;

private String ownerName;

private Decimal amountDeposited;

public BankAccount(Integer accNumber, Decimal initialBalance, String owner, Decimal depositAmount) {

this.accountNumber = accNumber;

this.balance = initialBalance;

this.ownerName = owner;

this.amountDeposited = depositAmount;

}

public void accountDetails(Integer accNumber, Decimal initialBalance, String owner, Decimal depositAmount) {

this.accountNumber = accNumber;

this.balance = initialBalance;

this.ownerName = owner;

this.amountDeposited = depositAmount;

this.balance += depositAmount;

System.debug('Account Number: ' + accountNumber);

System.debug('Owner: ' + ownerName);

System.debug('Initial Balance: ' + initialBalance);

System.debug('Amount Deposited: ' + depositAmount);

System.debug('Total Balance: ' + balance);

}

public Decimal totalAmount() {

return this.balance;

}

public void displayAccountDetails() {

System.debug('Account Number: ' + accountNumber);

System.debug('Owner: ' + ownerName);

System.debug('Balance: ' + balance);

}

}

public class SavingsAccount extends BankAccount {

public SavingsAccount(Integer accNumber, Decimal initialBalance, String owner, Decimal depositAmount) {

super(accNumber, initialBalance, owner, depositAmount); // Calls the parent class constructor

}

public Decimal calculateInterest(Decimal interestRate) {

Decimal interest = this.totalAmount() \* (interestRate / 100);

Decimal newBalance = this.totalAmount() + interest;

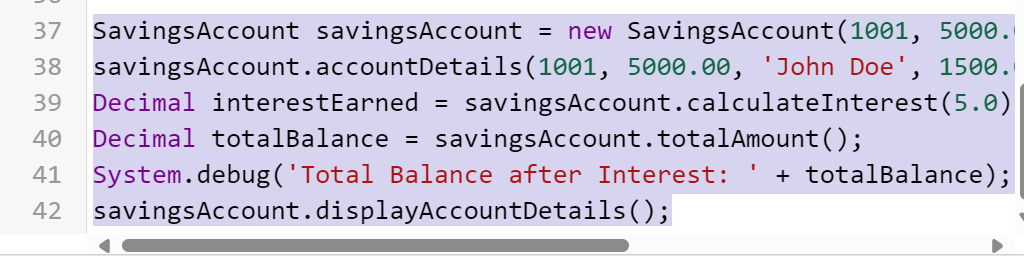
System.debug('Interest earned: ' + interest);

System.debug('New Balance after interest: ' + newBalance);

return interest;

}

}



A screenshot of a computer

Description automatically generated

​

**Problem Statement**​

Create a class called **Shape** with a method called **getShapeArea()**.​

Now, create a subclass called **RectangleShape** that overrides the getShapeArea() method to calculate the area of a rectangle.​

public virtual class Shape {

public virtual Decimal getShapeArea() {

return 0;

}

}

public class RectangleShape extends Shape {

private Decimal length;

private Decimal width;

public RectangleShape(Decimal length, Decimal width) {

this.length = length;

this.width = width;

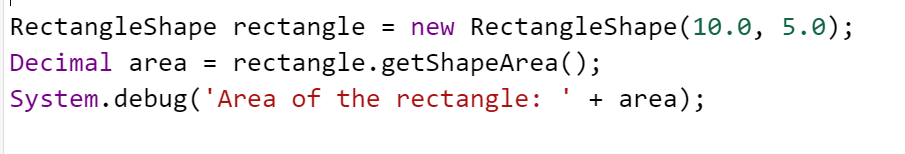
}

public override Decimal getShapeArea() {

return length \* width;

}

}



A blue rectangle with black text

Description automatically generated

​**Problem Statement**​

Create an interface called **DiscountCalculator** that defines a method for calculating discounts. Implement a class that uses this interface as **CustomerDiscount**.​

​

**Requirements:**​

* DiscountCalculator Interface: It includes a calculateDiscount method that takes an input parameter as originalAmount, which represents the original price of an item.​
* ​
* CustomerDiscount Class: The CustomerDiscount class implements the DiscountCalculator interface. It provides an implementation for the calculateDiscount method. ​

public interface DiscountCalculator {

Decimal calculateDiscount(Decimal originalAmount);

}

public class CustomerDiscount implements DiscountCalculator {

// discount rate 10%

private Decimal discountRate;

public CustomerDiscount(Decimal rate) {

this.discountRate = rate;

}

public Decimal calculateDiscount(Decimal originalAmount) {

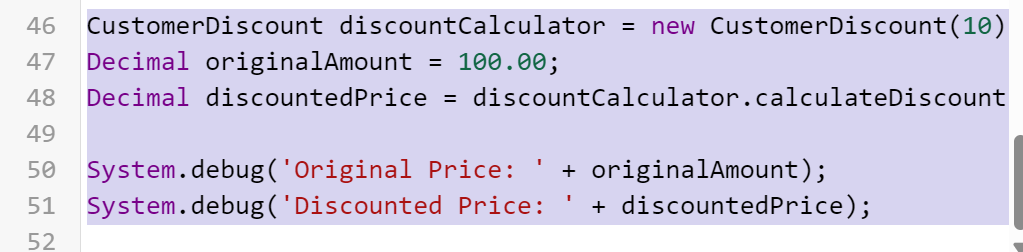
Decimal discount = originalAmount \* discountRate / 100;

Decimal discountedPrice = originalAmount - discount;

return discountedPrice;

}

}



A screenshot of a computer

Description automatically generated

​

​