01

17 December 2024 16:22

Employee Information System.

You've been tasked with developing a Java program for managing employee information. The program should allow users to input details for a employee, including their name, age and salary. Ensure that the age is a non-negative integer and the salary is a non-negative double. Once the details are entered, the program should display the employee's information.

Input Format:
The first line consists of a String that represents the name.
The second line consist of int which represents age.
The third line consists of a double that represents the salary.

Output Format:
The output should display the student's deatials.

Sample Input 1:
Rakesh Sharma
45
25000.00

Sample Output 1:
Employee details:
Name: Rakesh Sharma
Age : 45
Salary: 25000.0

Sample Input 2:
Rakesh Sharma
-25
45000.0

Sample Output 2:
Age must be a non-negative integer.
Employee details:
Name: Rakesh Sharma
Age: 0
Salary: 45000.00

Sample Input 3:
Rakesh Sharma
45
-1000

Sample Output 3:
Salary must be a non-negative double.
Employee details:
Name: Rakesh Sharma
Age: 45

Salary: 0.00

Sample Input 4:
Rakesh Sharma
-45
-1000

Sample Output 4:
Age must be a non-negative integer.
Salary must be a non-negative double.
Employee details:
Name: Rakesh Sharma
Age: 0
Salary: 0.00

```
class Employee
{
    private String name;
    private int age;
    private double salary;

    public Employee()
    {
    }
    public Employee(String name, int age, double salary)
    {
        this.name = name;
        this.age = age;
        this.salary = salary;
    }
    =
}
class Main
{
    public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        String name = scan.nextLine();
        int age = scan.nextInt();
```

```
double sal = scan.nextDouble();
if(age < 0)
{
    S.O.P("Age must be non-negative integer.");
    age = 0;
}
if(sal < 0.00)
{
    S.O.P("Salary must be non-negative double.");
    sal = 0;
}
Employee e = new Employee(name, age, sal);
S.O.P("Employee details:");
S.O.P("Name: " + e.getName());
S.O.P("Age: " + e.getAge());
S.O.printf("Salary: %.2f", e.getSalary());
    }
}
```

02

17 December 2024 16:41

Book billing system using HashMap

You are tasked with developing a simple Book bill system for a Sapna Book store. The system should allow the user to input the book name and book price they want to purchase. After the data is collected, the system should ask for a total bill price threshold. If the total price of books purchased exceeds the threshold, the customer will receive a 25% discount on the total bill. The program will ensure that only positive price values are accepted and will display the details of the purchased books, the total bill before discount, and the total bill after applying the discount if applicable.

Input Format
The first line of input consists of an integer representing the number of books the customer wants to purchase.
The next two lines consists of the user inputs for each book:
. Book name (String)
. Book price (double, must be a positive number)
After entering the book details, the last line of input consists of the total price threshold.

Output Format:
The output displays the book names and prices of the items (2 decimal values) purchased.
The total bill (2 decimal values) before any discount.
If the total bill exceeds the threshold, the output displays the total bill(2 decimal values) after a 25% discount. If not then "No discount applied as the total bill does not exceed the threshold".
If no books are purchased, the output prints a message stating that "No books were purchased.

Sample Input 1:
2
Java Complete Reference
100.00
C#
200.00
500.00

Sample Output 2:
Books purchased:
Book name: Java Complete Reference, Price: 100.00
Book name: C#, Price: 200.00
Total bill before discount: 300.00
No discount applied as the total bill does not exceed the threshold.

Sample Input 2:
2
Java Complete Reference
800.00
C#
200.00
200.00

Sample Output 2:
Books purchased:
Book name: Java Complete Reference, Price: 100.00
Book name: C#, Price: 200.00
Total bill before discount: 1000.00
Total bill after 25% discount: 750.00

```
class Main
{
    public static void main(String [] args)
    {
        Scanner scan = new Scanner(System.in);
        HashMap<String, Double> map = new HashMap<>();
        int n = scan.nextInt();
        for(int i=1; i<=n; i++)
        {
            scan.nextLine();
            String bookTitle = scan.nextLine();
            double price = scan.nextDouble();
            map.put(bookTitle, price);
        }
        double threshold = scan.nextDouble();
        S.O.P(" Books Purchased:");
```

```
double totalPrice = 0.00;

for(Map.Entry<String, Double> e : map.entrySet())
{
    S.O.printf(" Book Name :" + e.getKey() + ", Price : %.2f", e.getValue());
    totalPrice = totalPrice + e.getValue();
}
S.O.printf("\n Total bill before discount : %.2f", totalPrice);
if(totalPrice > threshold)
{
    totalPrice = totalPrice * 0.75;
    S.O.printf("\n Total bill after 25 \% discount : %.2f", totalPrice);
}
else
{
    S.O.P(" No discount applied as the total price does not exceed the threshold");
}
}
}
```

Electronic Store Management System using HashMap

You are assigned to develop a Electronic Store Management System in Java using HashMap. The system should enable users to efficiently manage the Store collection of Electronic products.

Functionalities:

Add Product: Users should be able to add new Product to the system by providing the product name and quantity.

Search Product: Users should be able to search for a product by providing the product name. The system should display the corresponding quantity if the product exists; otherwise, it should notify the user that the product does not exist.

Display All Products: Users should have the option to view all products stored in the system, including their product name and quantities.

Total Number of Product: The system should display the total number of products stored in the system.

Clear All Products: Users should be able to clear all products from the library in a single operation.

Input format :
The first line consists of the size of the HashMap.

The next lines consist of values representing products in the following format: "Product name, Product Quantity".

The last input consists of a String that represents the product name to search.

Output format :
The output should contain search result, number of products and product details.

Sample Input 1 :
4
HP ZenBook, 8
Dell Mouse, 5
Dell Precision, 10
MacBook Air, 2
Dell Precision

Sample Output 1 :

Dell Precision Quantity: 10
Total number of products: 4
All Products:
MacBook Air, Quantity=2
Dell Mouse, Quantity=5
HP ZenBook, Quantity=8
Dell Precision, Quantity=10
All Products Cleared

Sample Input 2 :

4
HP ZenBook, 8
Dell Mouse, 5
Dell Precision, 10
MacBook Air, 2
MacBook Pro

Sample Output 2 :

MacBook Pro does not exist
Total number of products: 4

All Products:
MacBook Air, Quantity=2
Dell Mouse, Quantity=5
HP ZenBook, Quantity=8
Dell Precision, Quantity=10
All Products Cleared

```
class Main
{   public static void main(String[] args)
    {
        Scanner scan = new Scanner(System.in);
        HashMap<String, Integer> map = new HashMap<>();
        int n = scan.nextInt();
        scan.nextLine();
        for(int i=1; i<=n; i++)
        {
            String line = scan.nextLine();
            String[] arr = line.split(",");
            String name = arr[0];
            int qty = Integer.parseInt(arr[1].trim());
            map.put(name, qty);
        }
        String searchProduct = scan.nextLine();
        if(map.containsKey(searchProduct))
        {
            S.O.P(searchProduct + " Quantity:" + map.get(searchProduct));
        }
        else
        {
            S.O.P(searchProduct + " does not exist");
        }
        S.O.P("Total number of products:" + map.size());
        S.O.P("All Products:");
        for(Map.Entry<String, Integer> e: map.entrySet())
        {
            S.O.P(e.getKey() + ", Quantity=" + e.getValue());
        }
        map.clear();
```

```
        }
        map.clear();
        S.O.P("All Products cleared");
    }
}
```

04

17 December 2024 17:51

Employee Salary Management using HashMap

You are tasked with developing a simple employee salary management system for a company. The system should allow the HR manager to input the names and salaries of employees. After the data is collected, the system should ask for a salary thresho ld and then display the names of all employees whose salaries are above that threshold. The program will ensure that only positi ve salary values are accepted and will display a message if no employees meet the salary criteria.

Input Format
The first line of input consists of an integer representing the number of employees.
The next two lines consist of the user inputs for each employee:
- Employee name (String)
- Employee salary (double, must be a positive number)
After entering employee details, the last line of input consists of the salary threshold.

Output Format
The output displays the names and salaries of employees whose salaries are greater than the provided threshold.

Please note that only up to two decimal places are allowed while printing the salary.

If no employees meet the criteria, the output prints a message stating that "No employees have a salary greater than the threshold".

Sample Input 1

4
Ajay
23500.50
Kiran
30000.00
Sunil
20000.00
Ravi
40000.00
25000.00

Sample Output 1

Employee: Kiran, Salary: 30000.00
Employee: Ravi, Salary: 40000.00

Sample Input 2
4
Ajay
23500.50
Kiran
30000.00
Sunil
20000.00
Ravi
40000.00
55000.00

Sample Output 2
No employees have a salary greater than the threshold

05

17 December 2024 20:09

06

17 December 2024 20:09

07

17 December 2024 20:09

08

17 December 2024 20:09