

Test Question 5(24.7.24)

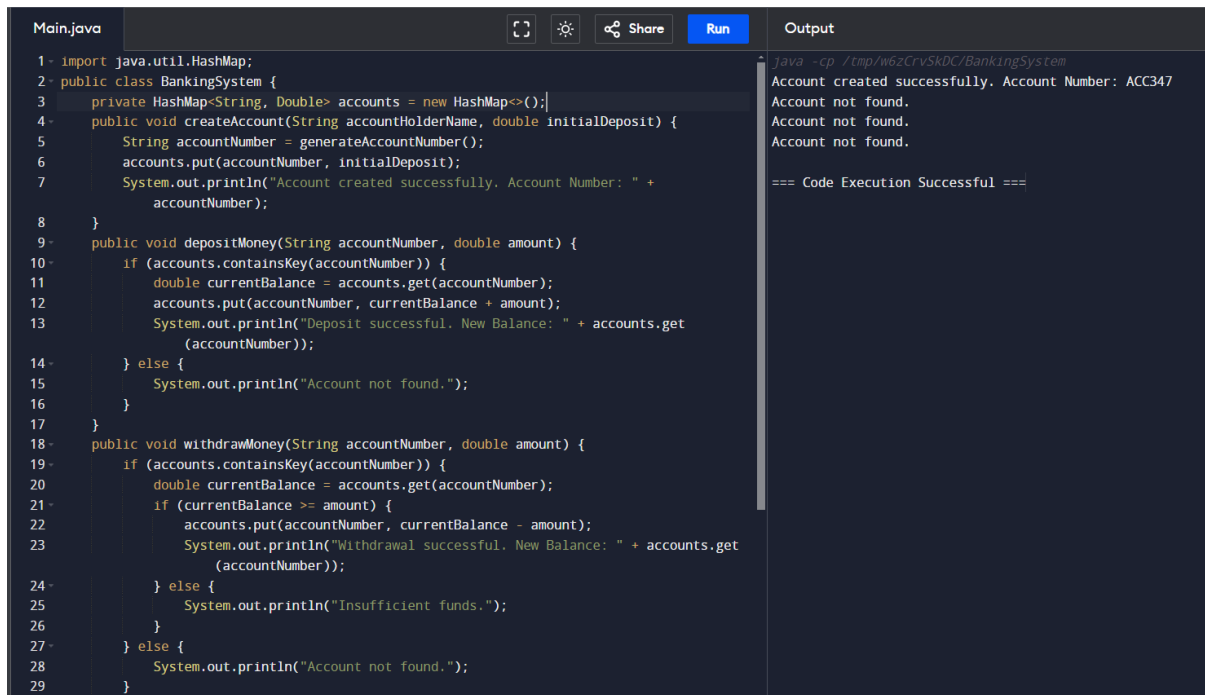
SET 1

1. Develop a simple banking system that allows users to create accounts, deposit money, withdraw money, and check balance. Implement methods for account creation, deposit, withdrawal, and balance inquiry.

Methods:

- createAccount(String accountHolderName, double initialDeposit)
- depositMoney(String accountNumber, double amount)
- withdrawMoney(String accountNumber, double amount)
- checkBalance(String accountNumber)

PROGRAM :-



The screenshot shows a Java IDE with a file named 'Main.java'. The code defines a 'BankingSystem' class with a private 'HashMap' for accounts. It includes methods for creating accounts, depositing money, and withdrawing money. The output window shows the results of running the program, including successful account creation and deposit, and failed attempts to find non-existent accounts.

```
1- import java.util.HashMap;
2- public class BankingSystem {
3-     private HashMap<String, Double> accounts = new HashMap<>();
4-     public void createAccount(String accountHolderName, double initialDeposit) {
5-         String accountNumber = generateAccountNumber();
6-         accounts.put(accountNumber, initialDeposit);
7-         System.out.println("Account created successfully. Account Number: " +
8-             accountNumber);
9-     }
10-    public void depositMoney(String accountNumber, double amount) {
11-        if (accounts.containsKey(accountNumber)) {
12-            double currentBalance = accounts.get(accountNumber);
13-            accounts.put(accountNumber, currentBalance + amount);
14-            System.out.println("Deposit successful. New Balance: " + accounts.get
15-                (accountNumber));
16-        } else {
17-            System.out.println("Account not found.");
18-        }
19-    }
20-    public void withdrawMoney(String accountNumber, double amount) {
21-        if (accounts.containsKey(accountNumber)) {
22-            double currentBalance = accounts.get(accountNumber);
23-            if (currentBalance >= amount) {
24-                accounts.put(accountNumber, currentBalance - amount);
25-                System.out.println("Withdrawal successful. New Balance: " + accounts.get
26-                    (accountNumber));
27-            } else {
28-                System.out.println("Insufficient funds.");
29-            }
30-        } else {
31-            System.out.println("Account not found.");
32-        }
33-    }
34-}
```

Output

```
java -cp /tmp/w6zCrvSkDC/BankingSystem
Account created successfully. Account Number: ACC347
Account not found.
Account not found.
Account not found.

=== Code Execution Successful ===
```

Main.java	Output
<pre> 19- if (accounts.containsKey(accountNumber)) { 20- double currentBalance = accounts.get(accountNumber); 21- if (currentBalance >= amount) { 22- accounts.put(accountNumber, currentBalance - amount); 23- System.out.println("Withdrawal successful. New Balance: " + accounts.get (accountNumber)); 24- } else { 25- System.out.println("Insufficient funds."); 26- } 27- } else { 28- System.out.println("Account not found."); 29- } 30- } 31- public void checkBalance(String accountNumber) { 32- if (accounts.containsKey(accountNumber)) { 33- System.out.println("Current Balance: " + accounts.get(accountNumber)); 34- } else { 35- System.out.println("Account not found."); 36- } 37- } 38- private String generateAccountNumber() { 39- return "ACC" + (int) (Math.random() * 1000); 40- } 41- public static void main(String[] args) { 42- BankingSystem bankingSystem = new BankingSystem(); 43- bankingSystem.createAccount("John Doe", 1000); 44- bankingSystem.depositMoney("ACC123", 500); 45- bankingSystem.withdrawMoney("ACC123", 200); 46- bankingSystem.checkBalance("ACC123"); 47- } 48- } 49- </pre>	<pre> java -cp /tmp/w6zCrvSkDC/BankingSystem Account created successfully. Account Number: ACC347 Account not found. Account not found. Account not found. === Code Execution Successful === </pre>

2. Create an expense tracker that allows users to add expenses, categorize them, and view a summary report. Implement methods to add expenses, categorize expenses, and generate reports.

Methods:

- addExpense(String description, double amount, String category)
- viewExpensesByCategory(String category)
- generateExpenseReport()

PROGRAM :-

Main.java	Output
<pre> 1- import java.util.ArrayList; 2- import java.util.HashMap; 3- import java.util.List; 4- import java.util.Map; 5- public class ExpenseTracker { 6- private List<Expense> expenses; 7- public ExpenseTracker() { 8- this.expenses = new ArrayList<>(); 9- } 10- public void addExpense(String description, double amount, String category) { 11- expenses.add(new Expense(description, amount, category)); 12- } 13- public void viewExpensesByCategory(String category) { 14- for (Expense expense : expenses) { 15- if (expense.getCategory().equals(category)) { 16- System.out.println(expense); 17- } 18- } 19- } 20- public void generateExpenseReport() { 21- Map<String, Double> categoryTotal = new HashMap<>(); 22- for (Expense expense : expenses) { 23- String category = expense.getCategory(); 24- double amount = expense.getAmount(); 25- 26- categoryTotal.put(category, categoryTotal.getOrDefault(category, 0.0) + amount); 27- } 28- for (Map.Entry<String, Double> entry : categoryTotal.entrySet()) { 29- System.out.println("Category: " + entry.getKey() + ", Total Amount: " + entry.getValue()); 30- } 31- } 32- } </pre>	<pre> java -cp /tmp/inJNWoYy2/ExpenseTracker Category: Utilities, Total Amount: 60.0 Category: Food, Total Amount: 80.0 Description: Groceries, Amount: 50.0, Category: Food Description: Dinner, Amount: 30.0, Category: Food === Code Execution Successful === </pre>

Main.java	<div><div><div></div><div></div><div></div></div><div>Share</div><div>Run</div></div>	Output
	<pre>31 } 32 public static void main(String[] args) { 33 ExpenseTracker tracker = new ExpenseTracker(); 34 tracker.addExpense("Groceries", 50.0, "Food"); 35 tracker.addExpense("Internet Bill", 60.0, "Utilities"); 36 tracker.addExpense("Dinner", 30.0, "Food"); 37 tracker.generateExpenseReport(); 38 tracker.viewExpensesByCategory("Food"); 39 } 40 } 41 class Expense { 42 private String description; 43 private double amount; 44 private String category; 45 public Expense(String description, double amount, String category) { 46 this.description = description; 47 this.amount = amount; 48 this.category = category; 49 } 50 public String getDescription() { 51 return description; 52 } 53 public double getAmount() { 54 return amount; 55 } 56 public String getCategory() { 57 return category; 58 } 59 public String toString() { 60 return "Description: "+description+",Amount: "+amount+",Category:"+category; 61 } 62 }</pre>	<pre>java -cp /tmp/lnJNWoYVy2/ExpenseTracker Category: Utilities, Total Amount: 60.0 Category: Food, Total Amount: 80.0 Description: Groceries, Amount: 50.0, Category: Food Description: Dinner, Amount: 30.0, Category: Food === Code Execution Successful ===</pre>