**Exercise 1: Control Structures**

**Scenario 1:** The bank wants to apply a discount to loan interest rates for customers above 60 years old.

* Question: Write a PL/SQL block that loops through all customers, checks their age, and if they are above 60, apply a 1% discount to their current loan interest rates.

**Scenario 2:** A customer can be promoted to VIP status based on their balance.

* + Question: Write a PL/SQL block that iterates through all customers and sets a flag IsVIP to TRUE for those with a balance over $10,000.

**Scenario 3:** The bank wants to send reminders to customers whose loans are due within the next 30 days.

* + Question: Write a PL/SQL block that fetches all loans due in the next 30 days and prints a reminder message for each customer.

public class Customer {

int id;

String name;

int age;

double balance;

boolean isVIP;

public Customer(int id, String name, int age, double balance) {

this.id = id;

this.name = name;

this.age = age;

this.balance = balance;

this.isVIP = false;

}

}

public class Loan {  
 int loanId;  
 int customerId;  
 double interestRate;  
 LocalDate dueDate;  
  
 public Loan(int loanId, int customerId, double interestRate, LocalDate dueDate) {  
 this.loanId = loanId;  
 this.customerId = customerId;  
 this.interestRate = interestRate;  
 this.dueDate = dueDate;  
 }  
}

**Scenario 1:**

import java.util.List;

public class Scenario1\_Discount {

public static void applyDiscount(List<Customer> customers, List<Loan> loans) {

for (Loan loan : loans) {

for (Customer cust : customers) {

if (loan.customerId == cust.id && cust.age > 60) {

loan.interestRate -= 1.0;

}

}

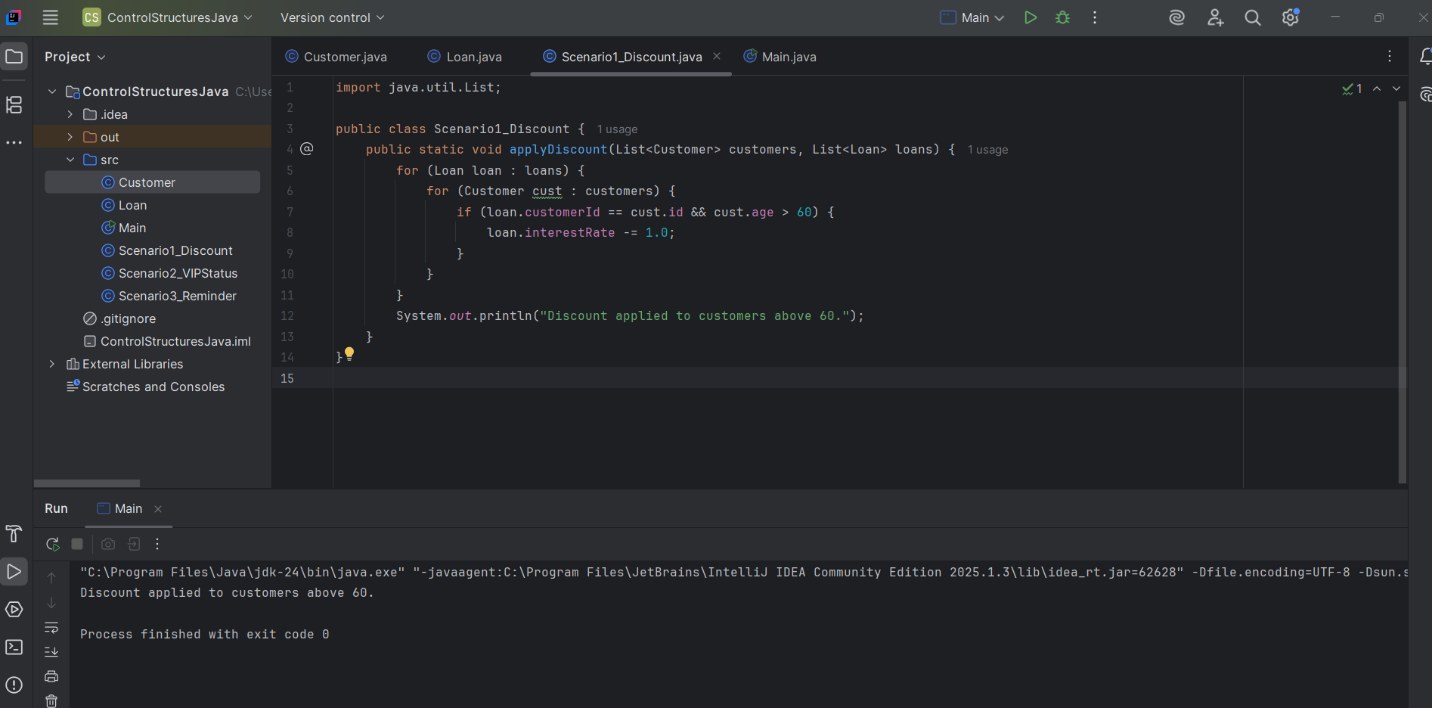
}

System.out.println("Discount applied to customers above 60.");

}

}

**Output:**

****

**Scenario 2:**

import java.util.List;

public class Scenario2\_VIPStatus {

public static void updateVIPStatus(List<Customer> customers) {

for (Customer cust : customers) {

if (cust.balance > 10000) {

cust.isVIP = true;

}

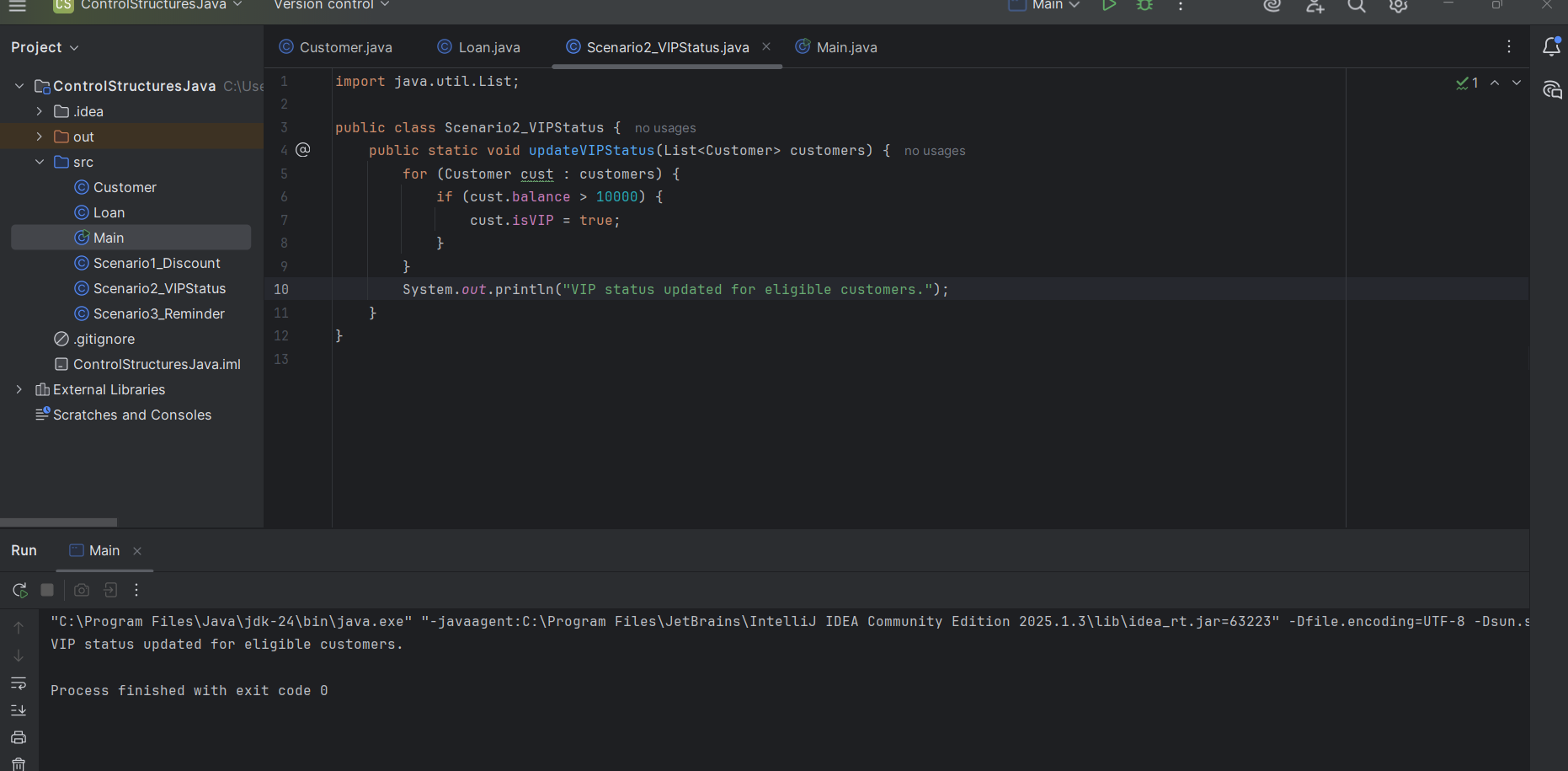
}

System.out.println("VIP status updated for eligible customers.");

}

}

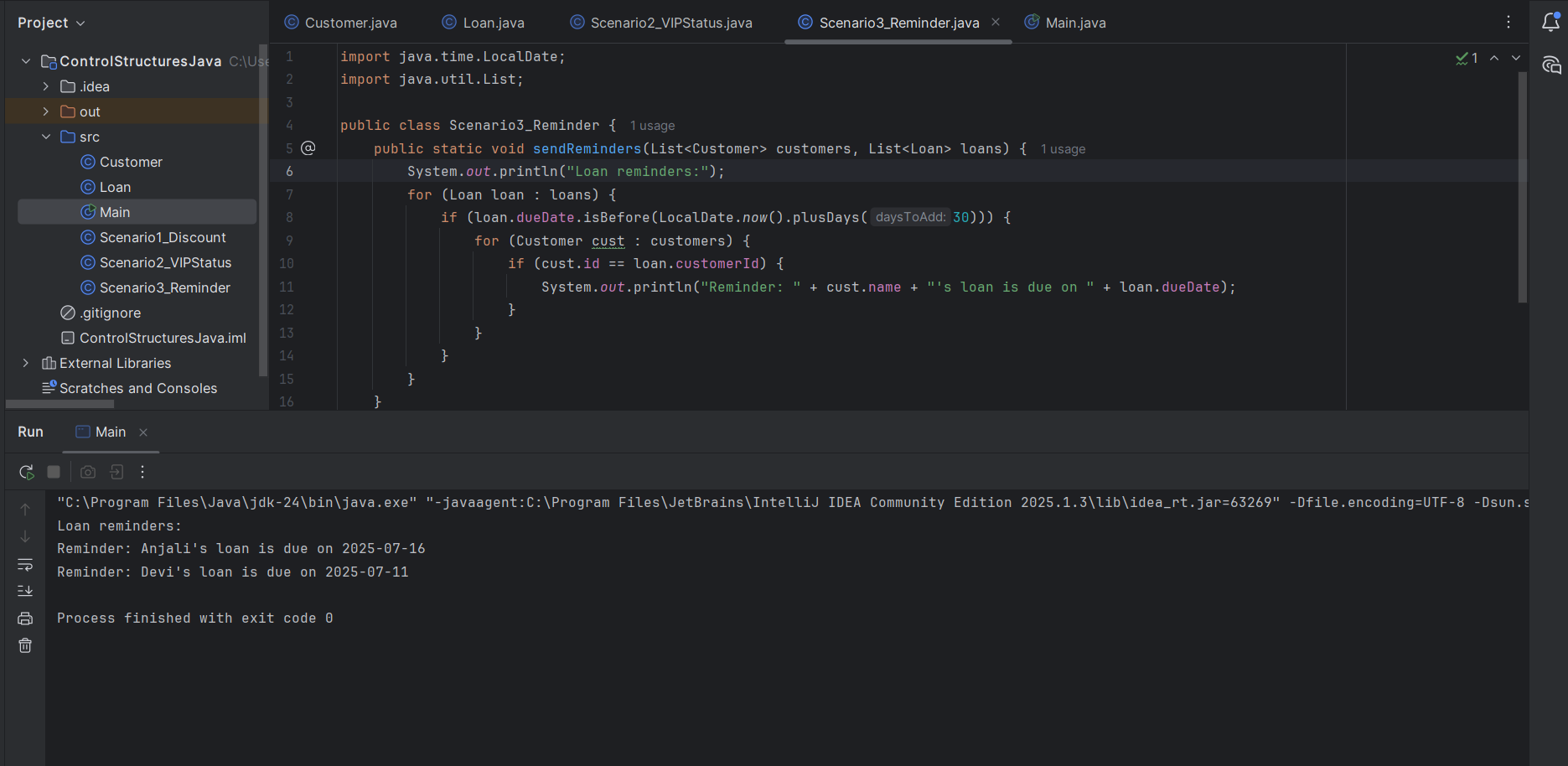
**Output:**

****

**Scenario 3:**

import java.time.LocalDate;  
import java.util.List;  
  
public class Scenario3\_Reminder {  
 public static void sendReminders(List<Customer> customers, List<Loan> loans) {  
 System.*out*.println("Loan reminders:");  
 for (Loan loan : loans) {  
 if (loan.dueDate.isBefore(LocalDate.*now*().plusDays(30))) {  
 for (Customer cust : customers) {  
 if (cust.id == loan.customerId) {  
 System.*out*.println("Reminder: " + cust.name + "'s loan is due on " + loan.dueDate);  
 }  
 }  
 }  
 }  
 }  
}

**Output:**

****

import java.time.LocalDate;  
import java.util.Arrays;  
import java.util.List;  
  
public class Main {  
 public static void main(String[] args) {  
  
 List<Customer> customers = Arrays.*asList*(  
 new Customer(1, "Anjali", 61, 12000),  
 new Customer(2, "Sathi", 60, 8000),  
 new Customer(3, "Devi", 75, 15000)  
 );  
  
 List<Loan> loans = Arrays.*asList*(  
 new Loan(101, 1, 7.5, LocalDate.*now*().plusDays(15)),  
 new Loan(102, 2, 8.0, LocalDate.*now*().plusDays(45)),  
 new Loan(103, 3, 6.8, LocalDate.*now*().plusDays(10))  
 );  
 Scenario1\_Discount.*applyDiscount*(customers, loans);  
 Scenario2\_VIPStatus.*updateVIPStatus*(customers);  
 Scenario3\_Reminder.*sendReminders*(customers, loans);  
 }  
}

**Exercise 3: Stored Procedures**

**Scenario 1:** The bank needs to process monthly interest for all savings accounts.

* + Question: Write a stored procedure ProcessMonthlyInterest that calculates and updates the balance of all savings accounts by applying an interest rate of 1% to the current balance.

**Scenario 2:** The bank wants to implement a bonus scheme for employees based on their performance.

* + Question: Write a stored procedure UpdateEmployeeBonus that updates the salary of employees in a given department by adding a bonus percentage passed as a parameter.

**Scenario 3:** Customers should be able to transfer funds between their accounts.

* + Question: Write a stored procedure TransferFunds that transfers a specified amount from one account to another, checking that the source account has sufficient balance before making the transfer.

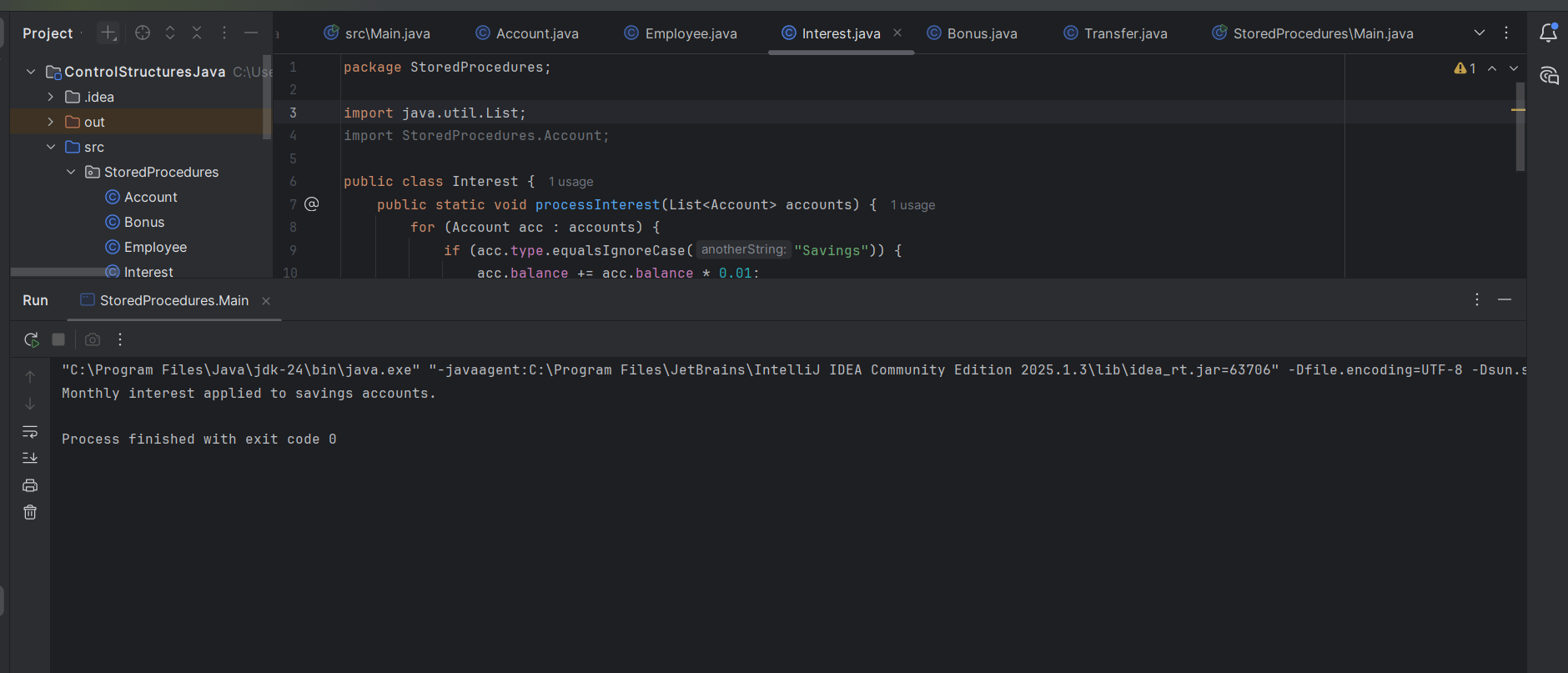
package StoredProcedures;  
  
public class Account {  
 int id;  
 String customerName;  
 String type; // "Savings" or "Checking"  
 double balance;  
  
 public Account(int id, String customerName, String type, double balance) {  
 this.id = id;  
 this.customerName = customerName;  
 this.type = type;  
 this.balance = balance;  
 }  
}

package StoredProcedures;  
  
public class Employee {  
 int id;  
 String name;  
 String department;  
 double salary;  
  
 public Employee(int id, String name, String department, double salary) {  
 this.id = id;  
 this.name = name;  
 this.department = department;  
 this.salary = salary;  
 }  
}

**Scenario 1: ProcessMonthlyInterest**

package StoredProcedures;  
  
import java.util.List;   
import StoredProcedures.Account;   
  
public class Interest {  
 public static void processInterest(List<Account> accounts) {  
 for (Account acc : accounts) {  
 if (acc.type.equalsIgnoreCase("Savings")) {  
 acc.balance += acc.balance \* 0.01;  
 }  
 }  
 System.*out*.println("Monthly interest applied to savings accounts.");  
 }  
}

**Output:**



**Scenario 2: UpdateEmployeeBonus**

package StoredProcedures;

import java.util.List;

import StoredProcedures.Employee;

public class Bonus {

public static void applyBonus(List<Employee> employees, String department, double bonusPercent) {

for (Employee emp : employees) {

if (emp.department.equalsIgnoreCase(department)) {

emp.salary += emp.salary \* (bonusPercent / 100);

}

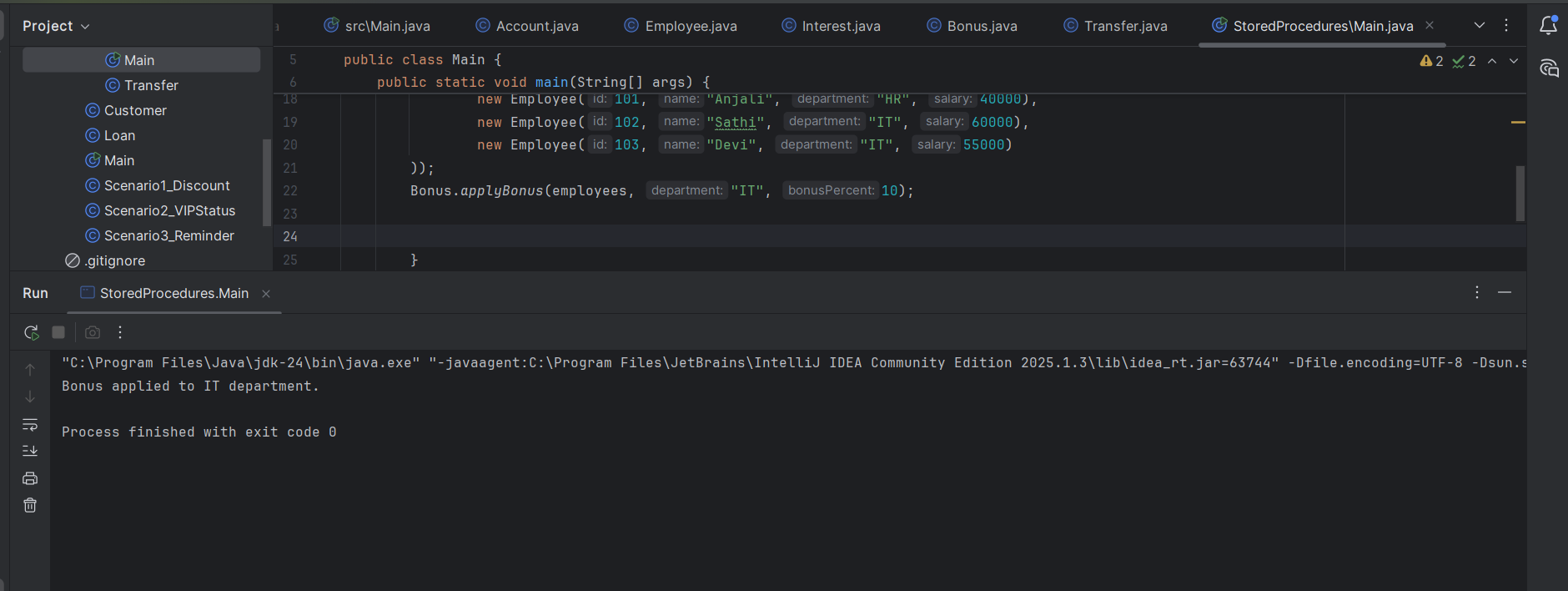
}

System.out.println("Bonus applied to " + department + " department.");

}

}

**Output:**

****

**Scenario 3: TransferFunds**

package StoredProcedures;

import java.util.List;

import StoredProcedures.Account;

public class Transfer {

public static void transferFunds(List<Account> accounts, int fromId, int toId, double amount) {

Account fromAcc = null, toAcc = null;

for (Account acc : accounts) {

if (acc.id == fromId) fromAcc = acc;

if (acc.id == toId) toAcc = acc;

}

if (fromAcc != null && toAcc != null) {

if (fromAcc.balance < amount) {

System.out.println("Transfer failed: Insufficient balance.");

} else {

fromAcc.balance -= amount;

toAcc.balance += amount;

System.out.println("Transfer successful: " + amount + " from Account " + fromId + " to Account " + toId);

}

} else {

System.out.println("Transfer failed: Account not found.");

}

}

}

package StoredProcedures;

import java.util.\*;

public class Main {

public static void main(String[] args) {

List<Account> accounts = new ArrayList<>(Arrays.asList(

new Account(1, "Anjali", "Savings", 1000),

new Account(2, "Sathi", "Savings", 2000),

new Account(3, "Devi", "Checking", 1500),

new Account(4, "Reddy", "Checking", 2500)

));

List<Employee> employees = new ArrayList<>(Arrays.asList(

new Employee(101, "Anjali", "HR", 40000),

new Employee(102, "Sathi", "IT", 60000),

new Employee(103, "Devi", "IT", 55000)

));

Interest.processInterest(accounts);

Bonus.applyBonus(employees, "IT", 10);

Transfer.transferFunds(accounts, 3, 4, 500);

System.out.println("\n--- Final Account Balances ---");

for (Account acc : accounts) {

System.out.println("ID: " + acc.id + ", Name: " + acc.customerName +

", Type: " + acc.type + ", Balance: ₹" + acc.balance);

}

System.out.println("\n--- Final Employee Salaries ---");

for (Employee emp : employees) {

System.out.println("ID: " + emp.id + ", Name: " + emp.name +

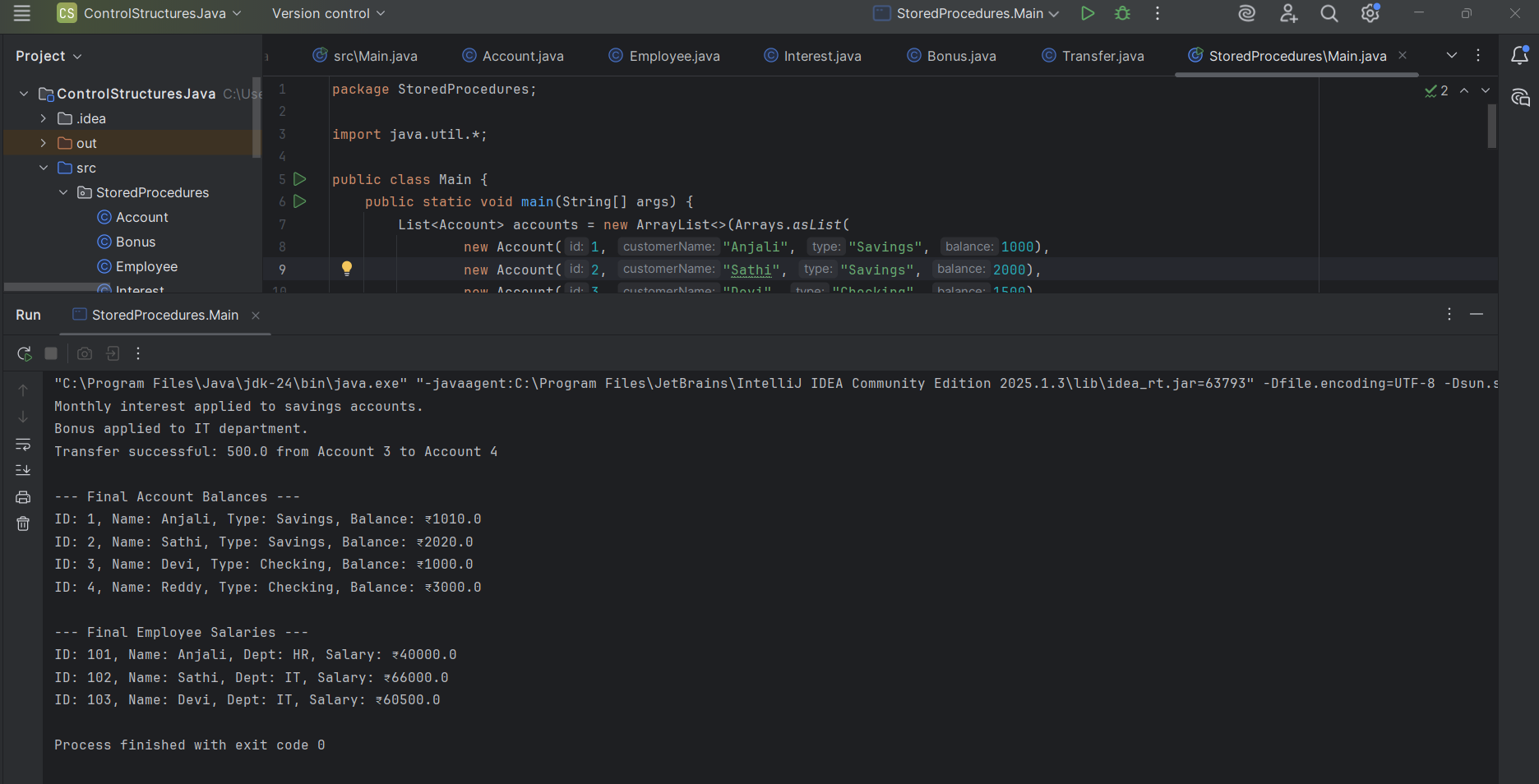
", Dept: " + emp.department + ", Salary: ₹" + emp.salary);

}

}

}

**Output:**

****