Southern University Bangladesh

Department of Computer Science and Engineering

Faculty of Science and Engineering

Object Oriented Programming Lab

Lab Report: 04

Course code : CSE 0613-108

Submitted by:

Name : Sharmistha Chowdhury

ID: 666-61-60

Submitted to :

Mohammed Arif Hasan Chowdhury, Assistant Professor

Dept. of Computer Science and Engineering

Date : 26/04/2024

	Index	
Sl	Name of Program	Remarks
1	**Class Practice**: Write a Java program that includes a superclass named Account and two subclasses named Savings and Current. The Account class should have properties for account name, account number, and balance, and a method to calculate tax at 2.5%. The Savings and Current classes should inherit these properties and have methods to calculate interest rates of 15% and 5% respectively.	
2	**Problem 01:** Write a Java program that creates a parent class with a method that prints “This is parent class”, and a child class that inherits from the parent class and has its own method that prints “This is child class” and demonstrate calling the parent class method with both the parent and child class objects, and the child class method with the child class object.	
3	**Problem 02:** Write a Java program that creates a parent class ‘Member’ with properties like name, age, phone number, address, and salary, and methods to print the salary, and two child classes ‘Employee’ and ‘Manager’ that inherit from the ‘Member’ class and have their own unique properties. Assign values to these properties and print the salaries of an employee and a manager by creating objects of these classes.	
4	**Problem 03**: Write a Java program that includes an Arithmetic class with an add method and an Adder class that inherits from Arithmetic. The add method should take two integers as parameters and return their sum.	
5		
6		
7		
8		
9		
10		
11		
12		
13		
14		
15		

Class Practice: Write a Java program that includes a superclass named Account and two subclasses named Savings and Current. The Account class should have properties for account name, account number, and balance, and a method to calculate tax at 2.5%. The Savings and Current classes should inherit these properties and have methods to calculate interest rates of 15% and 5% respectively.

Objective:

The objective of this program is to model a banking system with different types of accounts (Savings and Current), each having unique properties and behaviors, and to perform operations like calculating tax and interest.

Algorithm:

- Start the program.
- Define a superclass named Account with properties for account name, account number, and balance.
  - ✓ This class should have a constructor that initializes these properties.
  - ✓ This class should also have a method named calculateTax that calculates tax at a rate of 2.5% on the balance and returns the result.
- Define a subclass named Savings that extends Account.
  - ✓ This class should have a constructor that calls the superclass constructor to initialize the inherited properties.
  - ✓ This class should also have a method named calculateInterest that calculates interest at a rate of 15% on the balance and returns the result.
- Define another subclass named Current that also extends Account.
  - ✓ This class should have a constructor that calls the superclass constructor to initialize the inherited properties.
  - ✓ This class should also have a method named calculateInterest that calculates interest at a rate of 5% on the balance and returns the result.
- In the main method, create instances of Savings and Current with appropriate values for account name, account number, and balance.
- Call the calculateTax and calculateInterest methods on these instances and print the returned values.
- End the program.

Code:

```
package Inheritence;

public class MainAccount {
   protected String accountName;
   protected String accountNumber;
   protected double balance;

   public MainAccount(String accountName, String accountNumber, double balance) {
      this.accountName = accountName;
      this.accountNumber = accountNumber;
      this.balance = balance;
   }

   public double calculateTax() {
      return balance * 0.025; // Tax rate is 2.5%
   }

   public double calculateInterest() {
      return 0;
   }
}
```

```
package Inheritence;

public class SavingsAccount extends MainAccount {
   public SavingsAccount(String accountName, String accountNumber, double balance) {
      super(accountName, accountNumber, balance);
   }

   /**
    *
    * @return
    */
   @Override
   public double calculateInterest() {
      double balanceAfterTax = balance - calculateTax();
      return balanceAfterTax * 0.15; // Interest rate for savings account is 15%
   }
}
```

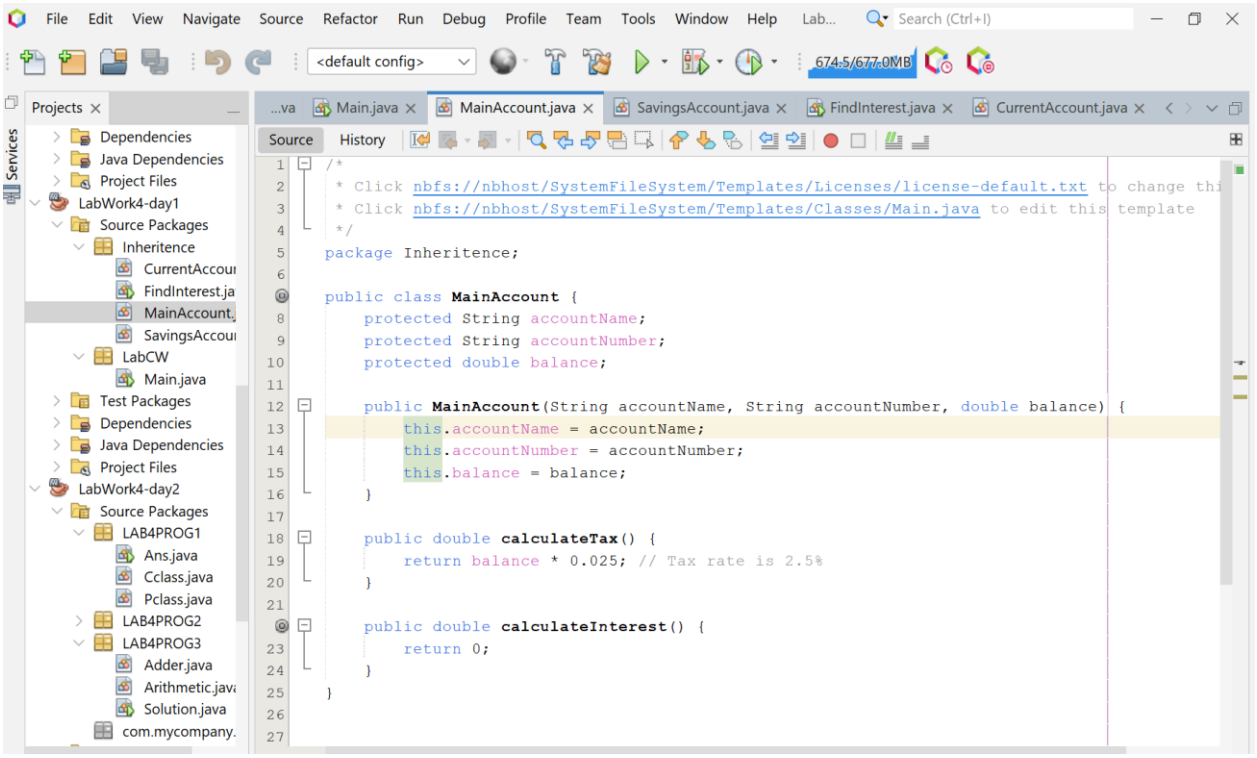
```
package Inheritence;

public class CurrentAccount extends MainAccount {
    public CurrentAccount(String accountName, String accountNumber, double balance) {
        super(accountName, accountNumber, balance);
    }

    @Override
    public double calculateInterest() {
        double balanceAfterTax = balance - calculateTax();
        return balanceAfterTax * 0.05; // Interest rate for current account is 5%
    }
}
package Inheritence;

import java.util.Scanner;

public class FindInterest {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter account name:");
        String accountName = scanner.nextLine();

        System.out.println("Enter account number for Savings account:");
        String savingsAccountNumber = scanner.nextLine();

        System.out.println("Enter balance for Savings account:");
        double savingsBalance = scanner.nextDouble();

        SavingsAccount savingsAccount = new SavingsAccount(accountName,
savingsAccountNumber, savingsBalance);
        System.out.println("Savings Account Interest: " + savingsAccount.calculateInterest());
```

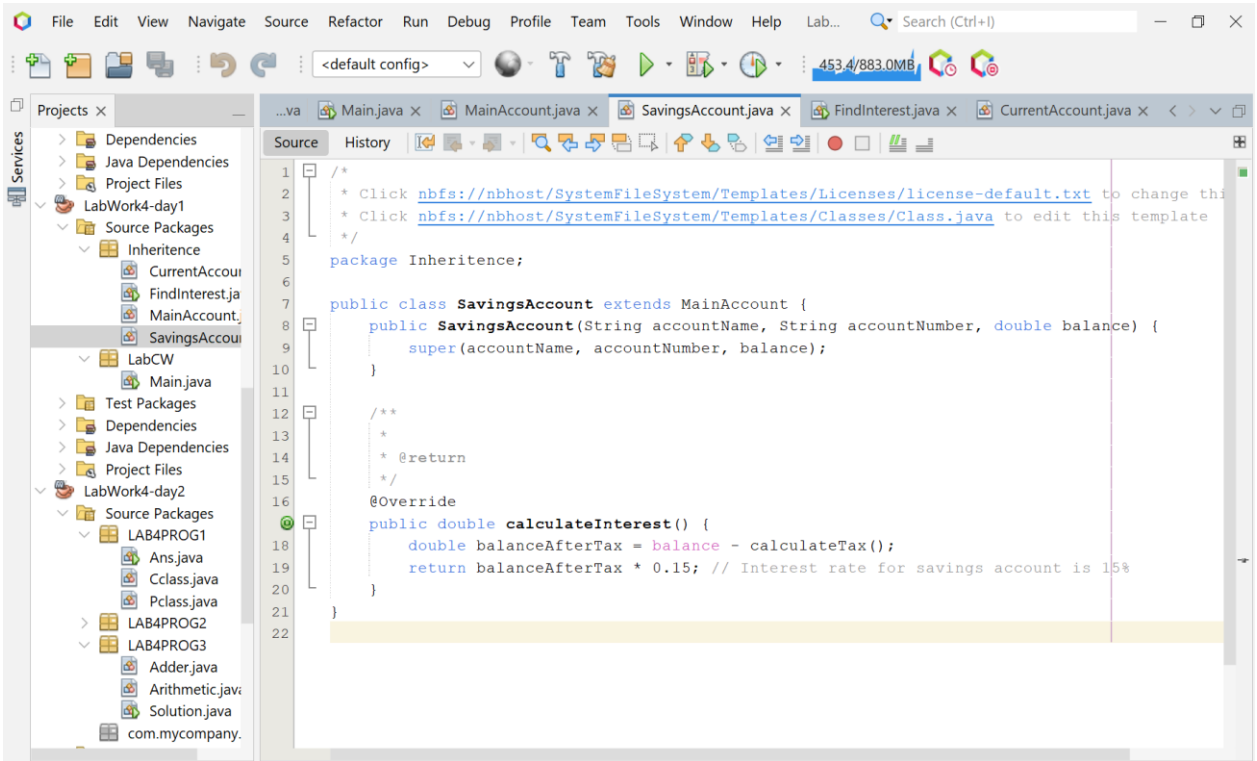
```
        scanner.nextLine(); // Consume newline left-over

        System.out.println("Enter account number for Current account:");
        String currentAccountNumber = scanner.nextLine();

        System.out.println("Enter balance for Current account:");
        double currentBalance = scanner.nextDouble();

        CurrentAccount currentAccount = new CurrentAccount(accountName,
currentAccountNumber, currentBalance);
        System.out.println("Current Account Interest: " + currentAccount.calculateInterest());
    }
}
```

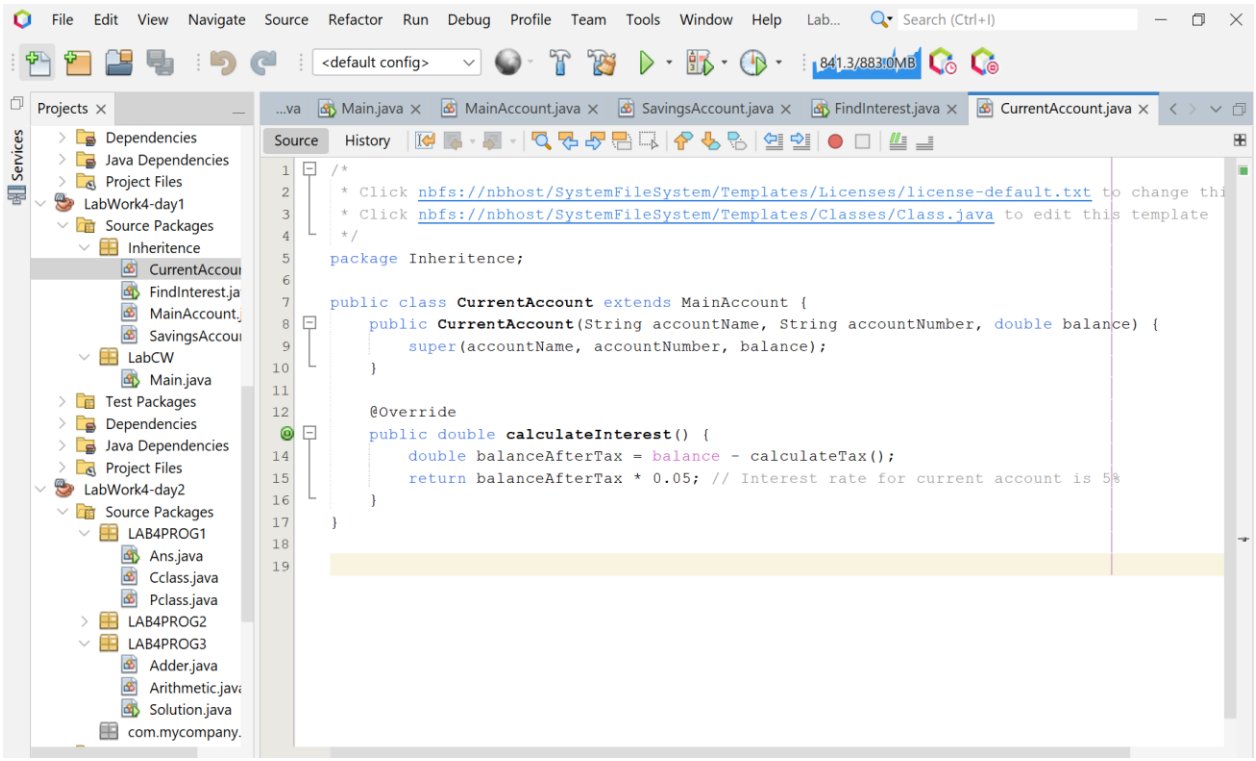
Input/ Expected Output:

```
Enter account name:
Sharmistha
Enter account number for Savings account:
666-61-60
Enter balance for Savings account:
100000
Savings Account Interest: 14625.0
Enter account number for Current account:
666-62-60
Enter balance for Current account:
500000
Current Account Interest: 24375.0
```

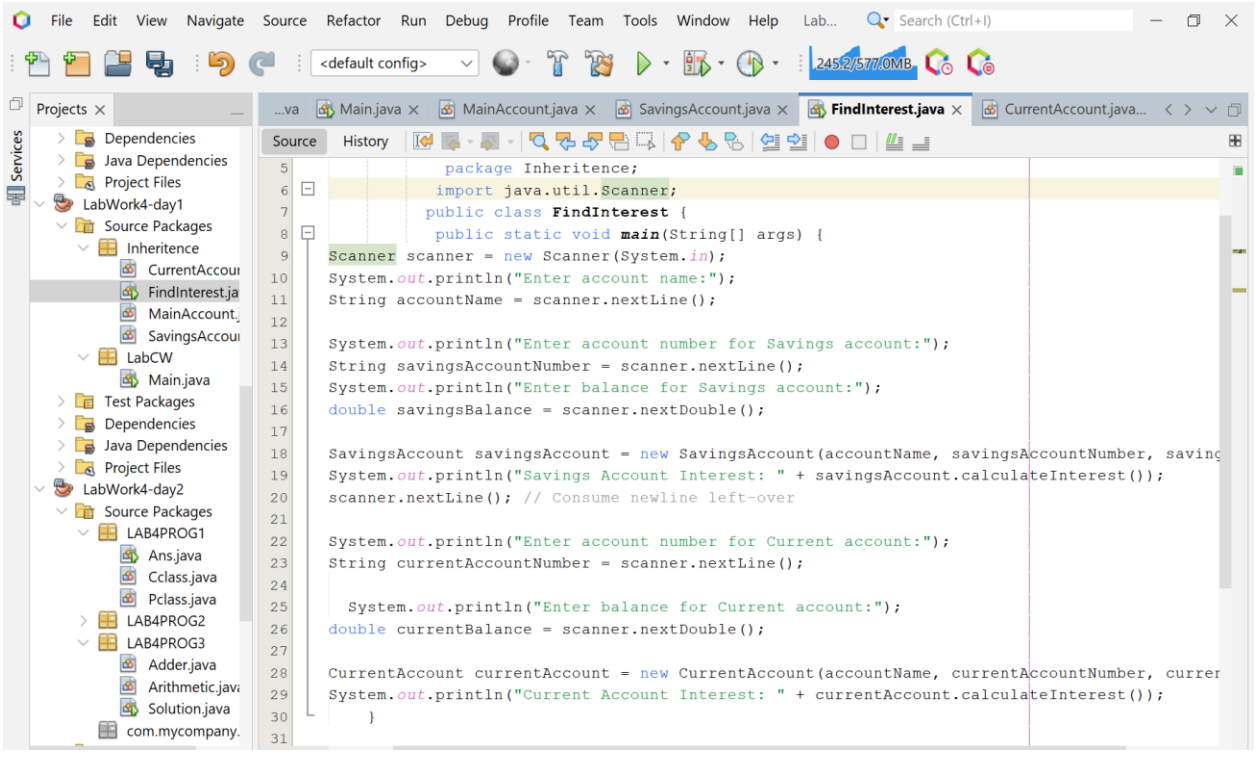
Screenshot of code edition window:





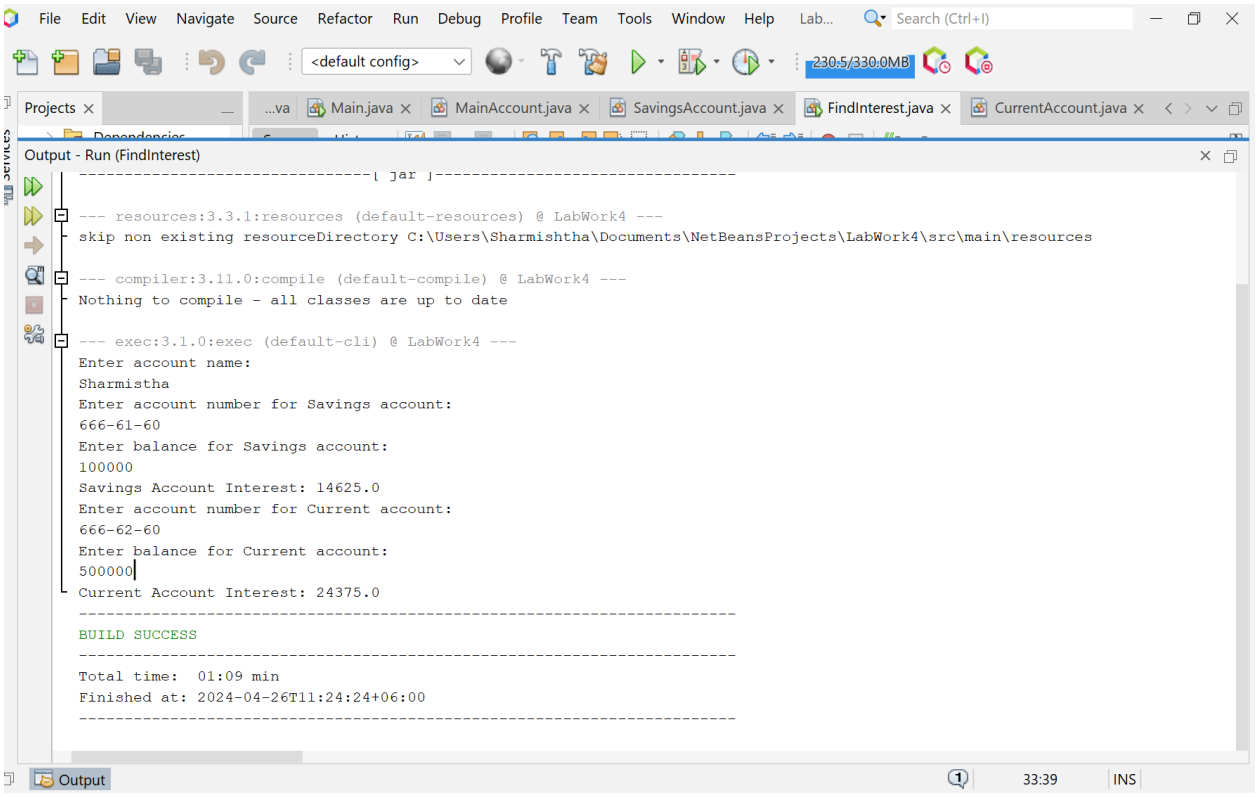


```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change thi
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package Inheritence;

public class CurrentAccount extends MainAccount {
    public CurrentAccount(String accountName, String accountNumber, double balance) {
        super(accountName, accountNumber, balance);
    }

    @Override
    public double calculateInterest() {
        double balanceAfterTax = balance - calculateTax();
        return balanceAfterTax * 0.05; // Interest rate for current account is 5%
    }
}
```



```
package Inheritence;
import java.util.Scanner;
public class FindInterest {
 public static void main(String[] args) {
Scanner scanner = new Scanner(System.in);
System.out.println("Enter account name:");
String accountName = scanner.nextLine();

System.out.println("Enter account number for Savings account:");
String savingsAccountNumber = scanner.nextLine();
System.out.println("Enter balance for Savings account:");
double savingsBalance = scanner.nextDouble();

SavingsAccount savingsAccount = new SavingsAccount(accountName, savingsAccountNumber, saving
System.out.println("Savings Account Interest: " + savingsAccount.calculateInterest());
scanner.nextLine(); // Consume newline left-over

System.out.println("Enter account number for Current account:");
String currentAccountNumber = scanner.nextLine();

  System.out.println("Enter balance for Current account:");
double currentBalance = scanner.nextDouble();

CurrentAccount currentAccount = new CurrentAccount(accountName, currentAccountNumber, currer
System.out.println("Current Account Interest: " + currentAccount.calculateInterest());
    }
```

Screenshot of Output screen/Run screen window:



Explanation:

- The program begins by defining a superclass named Account that has three properties: accountName, accountNumber, and balance. This class also includes a constructor to initialize these properties and a method named calculateTax that calculates the tax on the balance at a rate of 2.5% and returns the result.

- Next, two subclasses named Savings and Current are defined. These classes extend the Account class, inheriting its properties and behaviors. They each have a constructor that calls the superclass constructor to initialize the inherited properties. They also each have a method named calculateInterest that calculates and returns the interest on the balance. The interest rate is 15% for Savings and 5% for Current.

- In the main method, an instance of java.util.Scanner is created to read the input from the standard input (keyboard).

- The program then prompts the user to input the account name, account number, and balance for a savings account and a current account. These values are stored in appropriate variables.

- Next, the program creates instances of Savings and Current with the entered details.

- Finally, the program calls the calculateTax and calculateInterest methods on these instances and prints the returned values to the standard output. These values represent the tax and interest for each account.

Discussion

This program effectively models a banking system with different types of accounts (Savings and Current), each having unique properties and behaviors, and performs operations like calculating tax and interest. It demonstrates the concept of inheritance in Java, where subclasses inherit properties and behaviors from a superclass, and can also provide their own unique behaviors.

Program 01: Write a Java program that creates a parent class with a method that prints “This is parent class”, and a child class that inherits from the parent class and has its own method that prints “This is child class” and demonstrate calling the parent class method with both the parent and child class objects, and the child class method with the child class object.

Objective:

The objective of the program is to demonstrate the concept of inheritance in Java by creating a parent class and a child class, and showing how methods from the parent class can be accessed by both parent and child class objects.

Algorithm:

- Start the program.
- Define a class named Pclass with a method pmethod that prints “This is parent class”.
- Define a subclass named Cclass that extends Pclass.
- This class should have a method cmethod that prints “This is child class”.
- In the main method, create an object m of Pclass and an object n of Cclass.
- Call the pmethod of Pclass using the object m.
- Call the cmethod of Cclass using the object n.
- Call the pmethod of Pclass using the object n.
- End the program.

Code:

```
class Pclass {
   public void pmethod() {
      System.out.println("This is parent class");
   }
}

class Cclass extends Pclass {
   public void cmethod() {
      System.out.println("This is child class");
   }
}

public class Ans {
   public static void main(String[] args) {
      Pclass m = new Pclass();
```

```
        Cclass n = new Cclass();
        m.pmethod();
        n.cmethod();
        n.pmethod();
    }
}
```

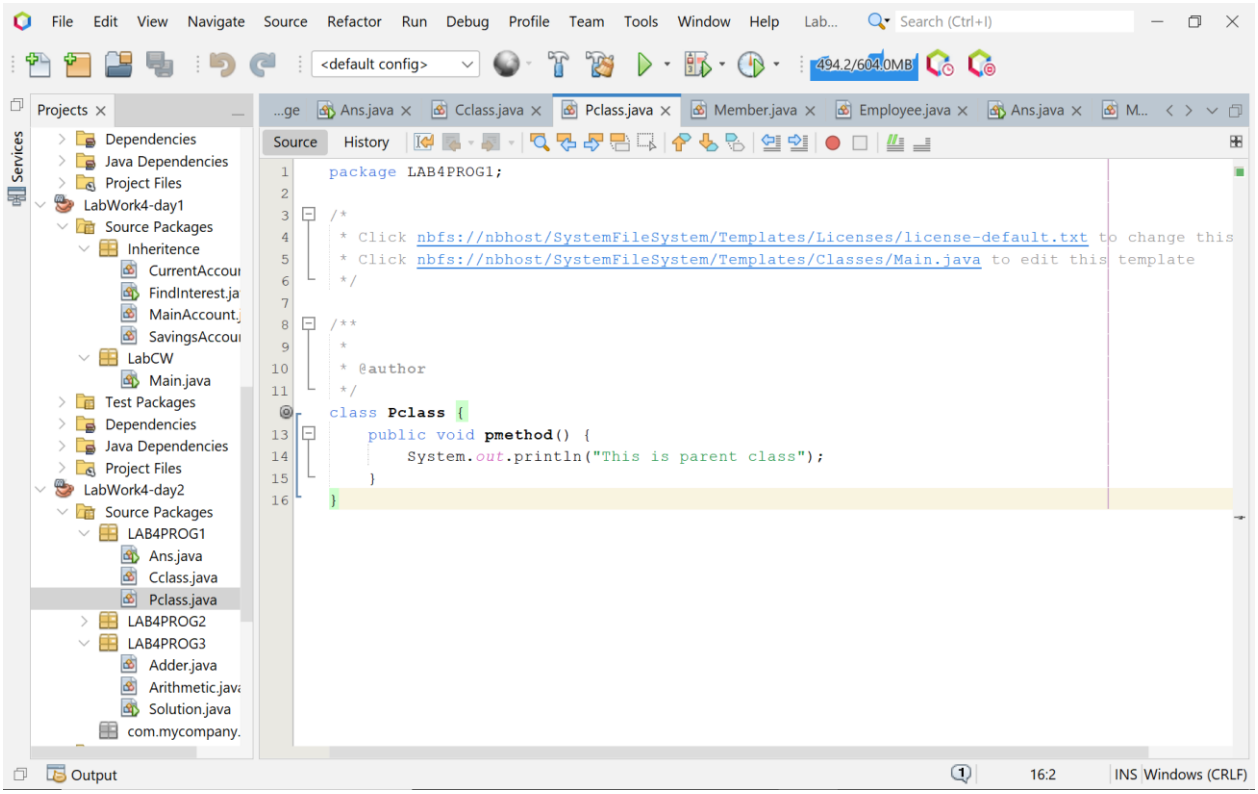
Input/ Expected Output:

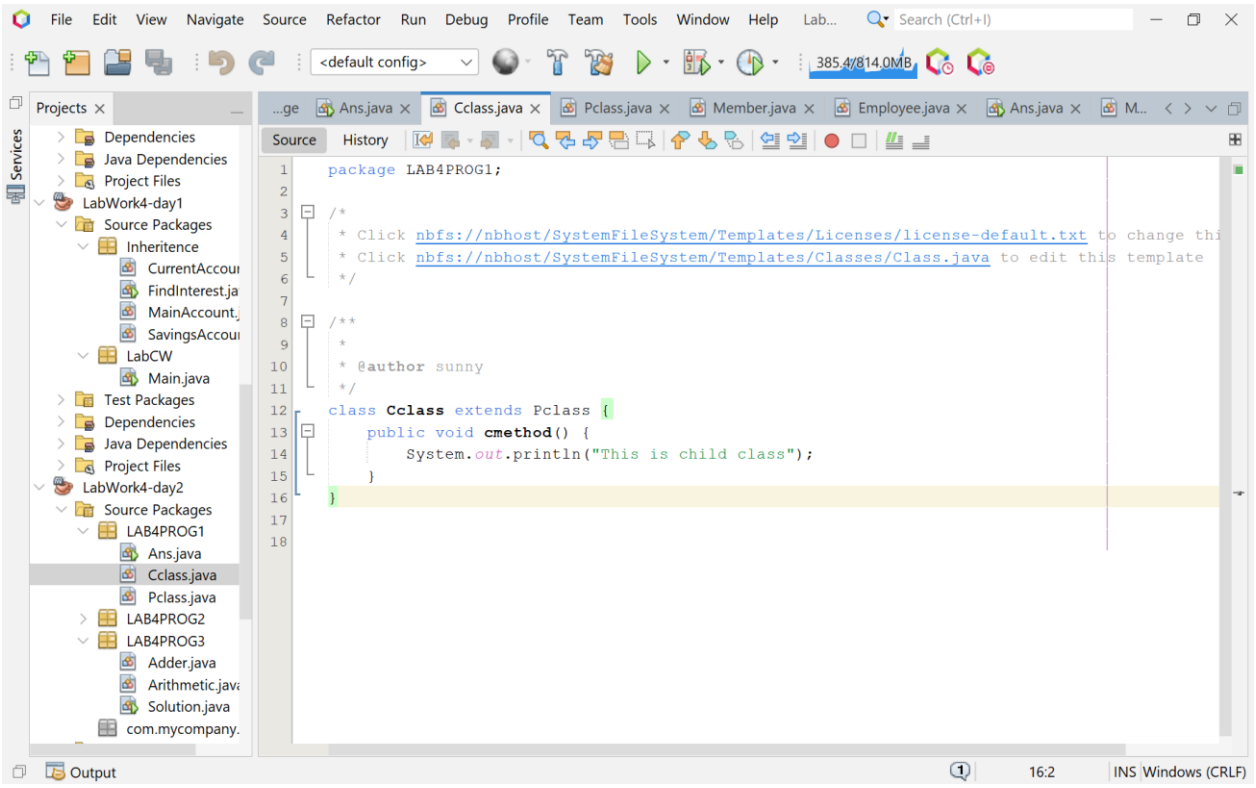
This is parent class

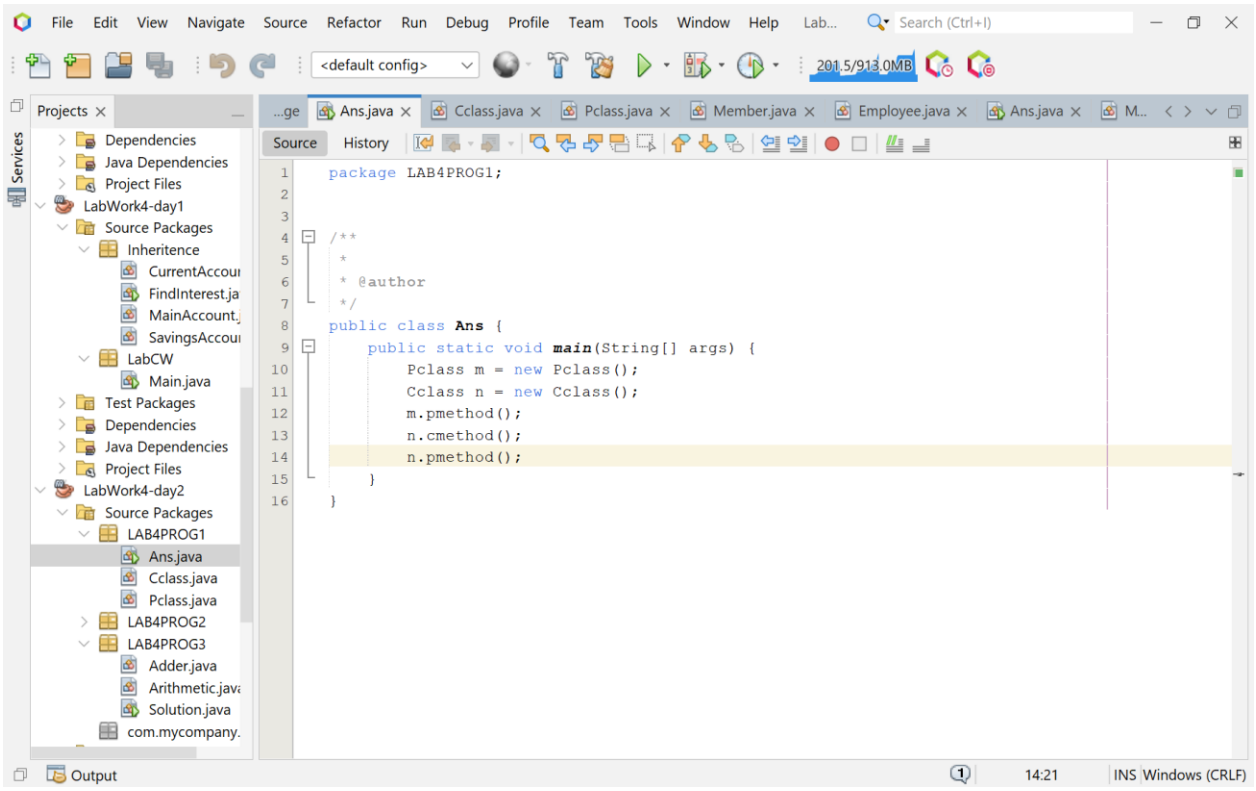
This is child class

This is parent class

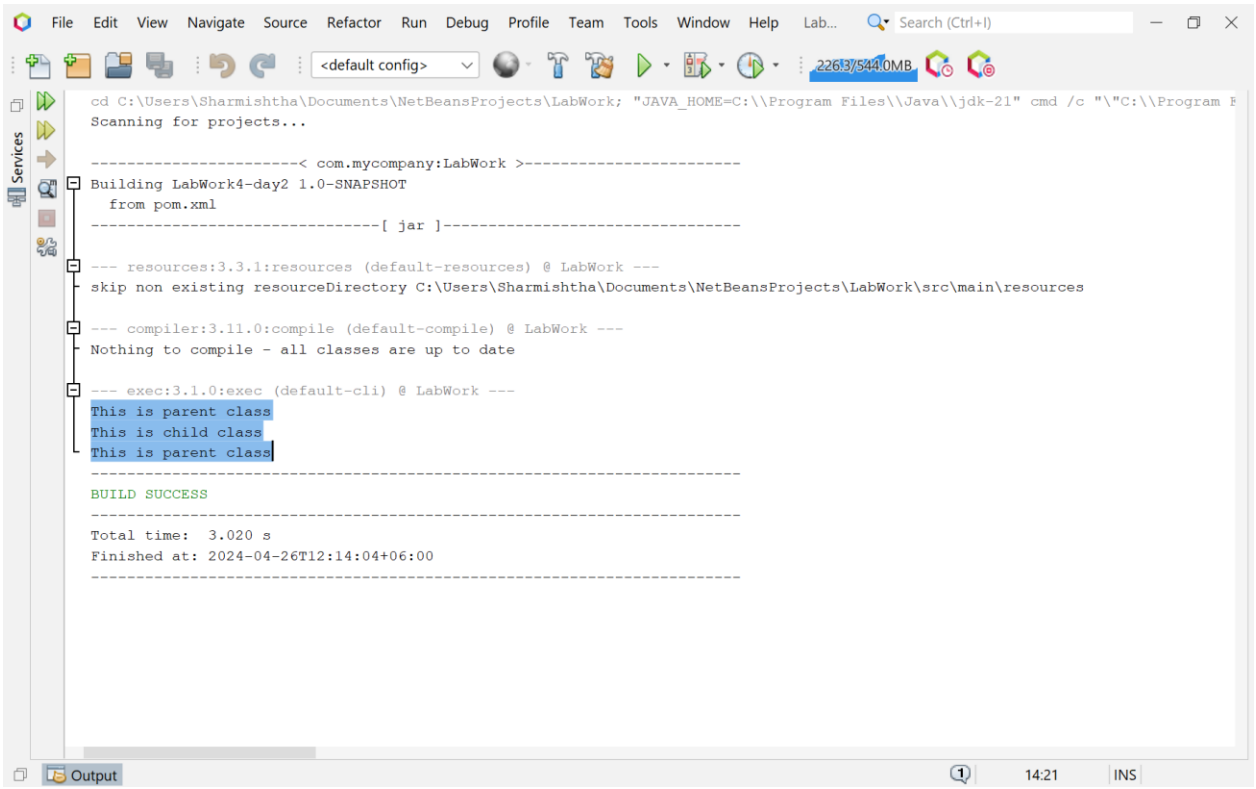
Screenshot of code edition window:




File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
385.4/814.0MB
Projects
Services
Ans.java
Cclass.java
Pclass.java
Member.java
Employee.java
Source History
package LAB4PROG1;
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change thi
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
*/
/**
*
* @author sunny
*/
class Cclass extends Pclass {
public void cmethod() {
System.out.println("This is child class");
}
}
Dependencies
Java Dependencies
Project Files
LabWork4-day1
Source Packages
Inheritence
LabCW
Main.java
Test Packages
LabWork4-day2
LAB4PROG1
LAB4PROG2
LAB4PROG3
Adder.java
Solution.java
com.mycompany.
Output
16:2
INS Windows (CRLF)


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
<default config>
201.5/913.0MB
Projects
Services
Ans.java
Cclass.java
Pclass.java
Member.java
Employee.java
Source History
package LAB4PROG1;
/**
*
* @author
*/
public class Ans {
public static void main(String[] args) {
Pclass m = new Pclass();
Cclass n = new Cclass();
m.pmethod();
n.cmethod();
n.pmethod();
}
}
Dependencies
Java Dependencies
Project Files
LabWork4-day1
Source Packages
Inheritence
LabCW
Main.java
Test Packages
LabWork4-day2
LAB4PROG1
LAB4PROG2
LAB4PROG3
Adder.java
Solution.java
com.mycompany.
Output
14:21
INS Windows (CRLF)

Screenshot of Output screen/Run screen window:



Explanation :

The program begins by defining a class named Pclass (parent class) with a method pmethod that prints “This is parent class”. Then, it defines another class named Cclass (child class) that extends Pclass. This class inherits all the properties and methods of Pclass and also defines its own method cmethod that prints “This is child class”. In the main method, an object m of Pclass and an object n of Cclass are created. Then, the pmethod of Pclass is called using the object m, the cmethod of Cclass is called using the object n, and again the pmethod of Pclass is called using the object n. This demonstrates that a child class object can access the methods of its parent class.

Discussion:

This program effectively demonstrates the basic principles of Object-Oriented Programming (OOP), specifically inheritance and polymorphism. Inheritance is a mechanism in which one class acquires the property of another class. In this program, the Cclass (child class) inherits the properties and methods of the Pclass (parent class). Polymorphism allows us to perform a single action in different ways. In this program, the pmethod of the parent class Pclass can be called by both the parent class object and the child class object. This is an example of runtime polymorphism (also known as dynamic method dispatch or late binding), where the call to an overridden method is resolved at runtime rather than at compile-time. This program is a simple yet powerful demonstration of these fundamental OOP concepts. It’s a great starting point for anyone learning Java or OOP.

Problem 02: Write a Java program that creates a parent class 'Member' with properties like name, age, phone number, address, and salary, and methods to print the salary, and two child classes 'Employee' and 'Manager' that inherit from the 'Member' class and have their own unique properties. Assign values to these properties and print the salaries of an employee and a manager by creating objects of these classes.

Objective:

The objective of the program is to demonstrate the concept of inheritance in Java by creating a parent class 'Member' and child classes 'Employee' and 'Manager', assigning values to their properties, and printing the salaries of an employee and a manager.

Algorithm:

- Start the program.
- Define a class named Member with data members name, age, number, address, and salary.
- This class should also have a method printSalary that prints the salary of the members.
- Define two subclasses named Employee and Manager that extend Member. These classes should have their own data members specialization and department respectively.
- In the main method, create an object e of Employee and an object m of Manager.
- Assign values to the data members of e and m.
- Call the printSalary method on e and m to print their salaries.
- End the program..

Code:

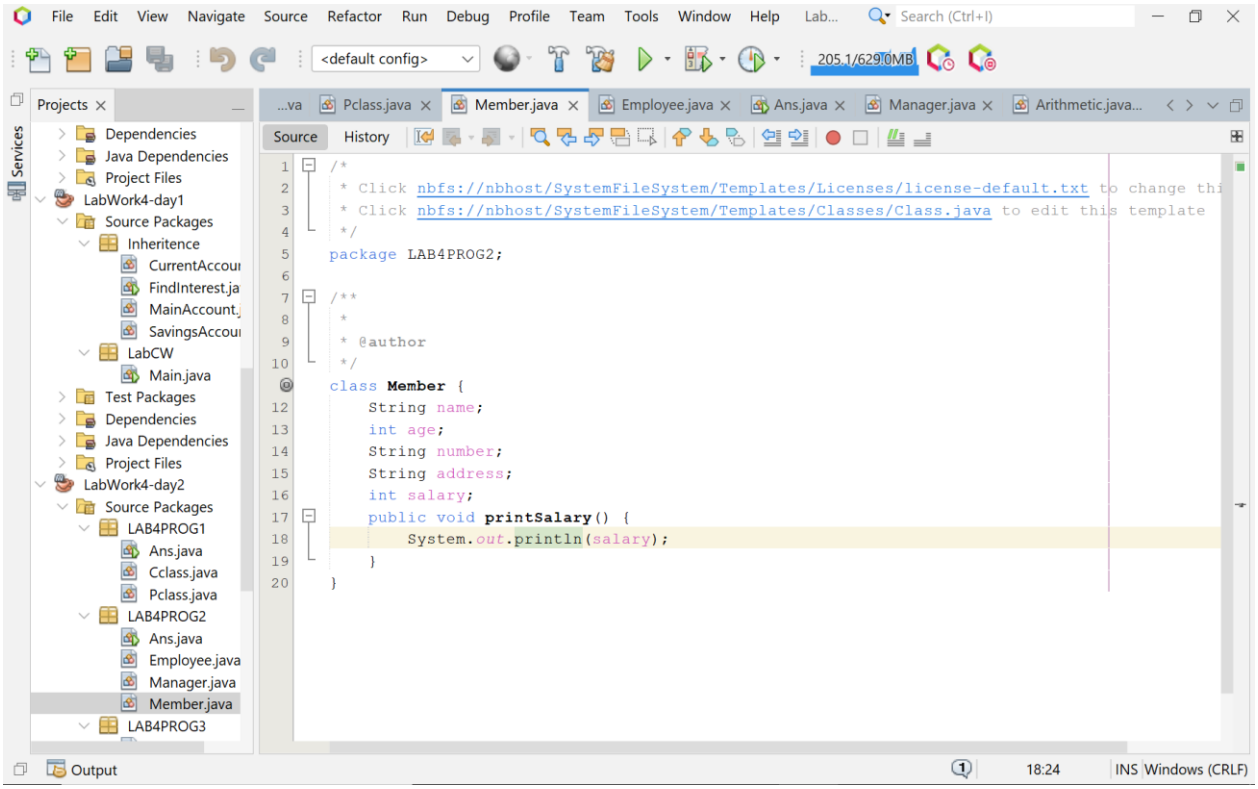
```
class Member {
   String name;
   int age;
   String number;
   String address;
   int salary;
   public void printSalary() {
      System.out.println(salary);
   }
}
   class Employee extends Member {
   String specialization;
}
```

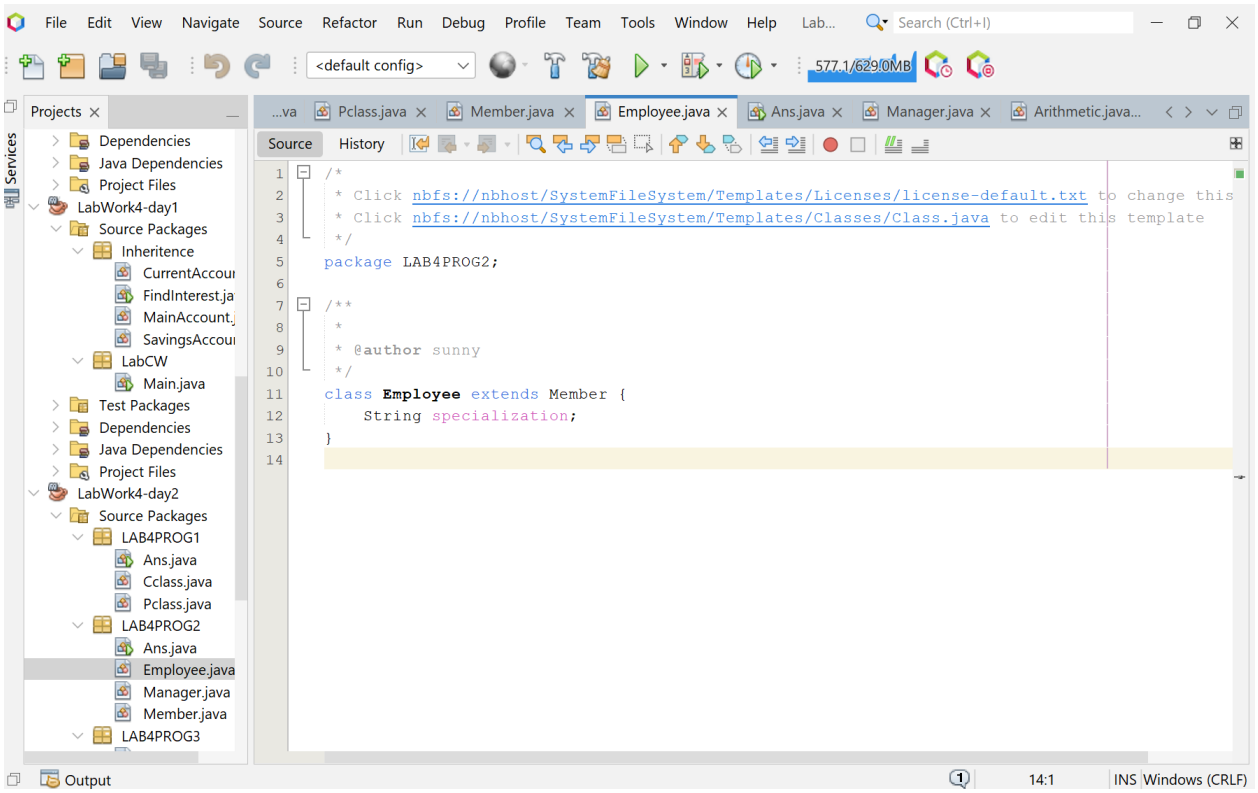
```
class Manager extends Member {
   String department;
}
public class Ans {
   public static void main(String[] args) {
      Employee e = new Employee();
      e.name = "xyz";
      e.age = 23;
      e.number = "986****";
      e.address = "xyzxyz";
      e.salary = 1231;
      e.specialization = "xyzxyz";
      e.printSalary();

      Manager m = new Manager();
      m.name = "abc";
      m.age = 45;
      m.number = "987****";
      m.address = "abcabc";
      m.salary = 4562;
      m.department = "abcabc";
      m.printSalary();
   }
}
```

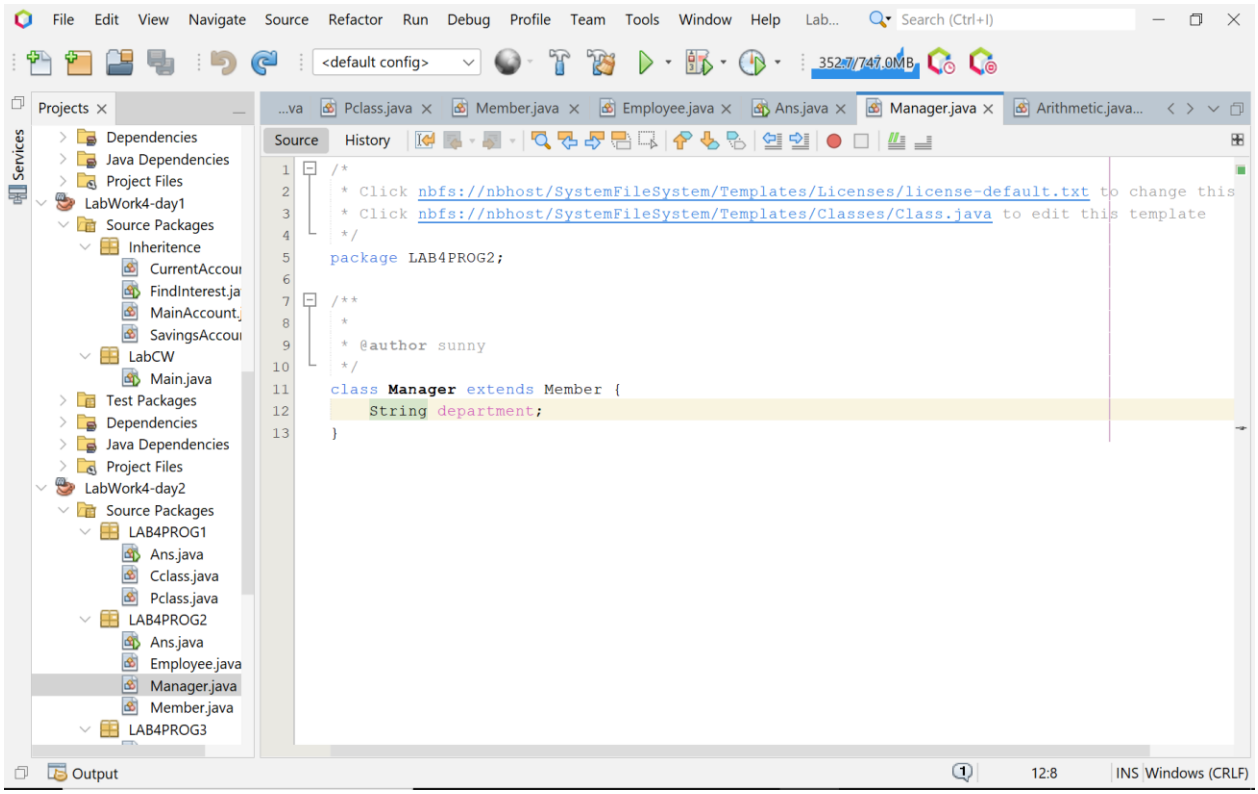
Input/ Expected Output:

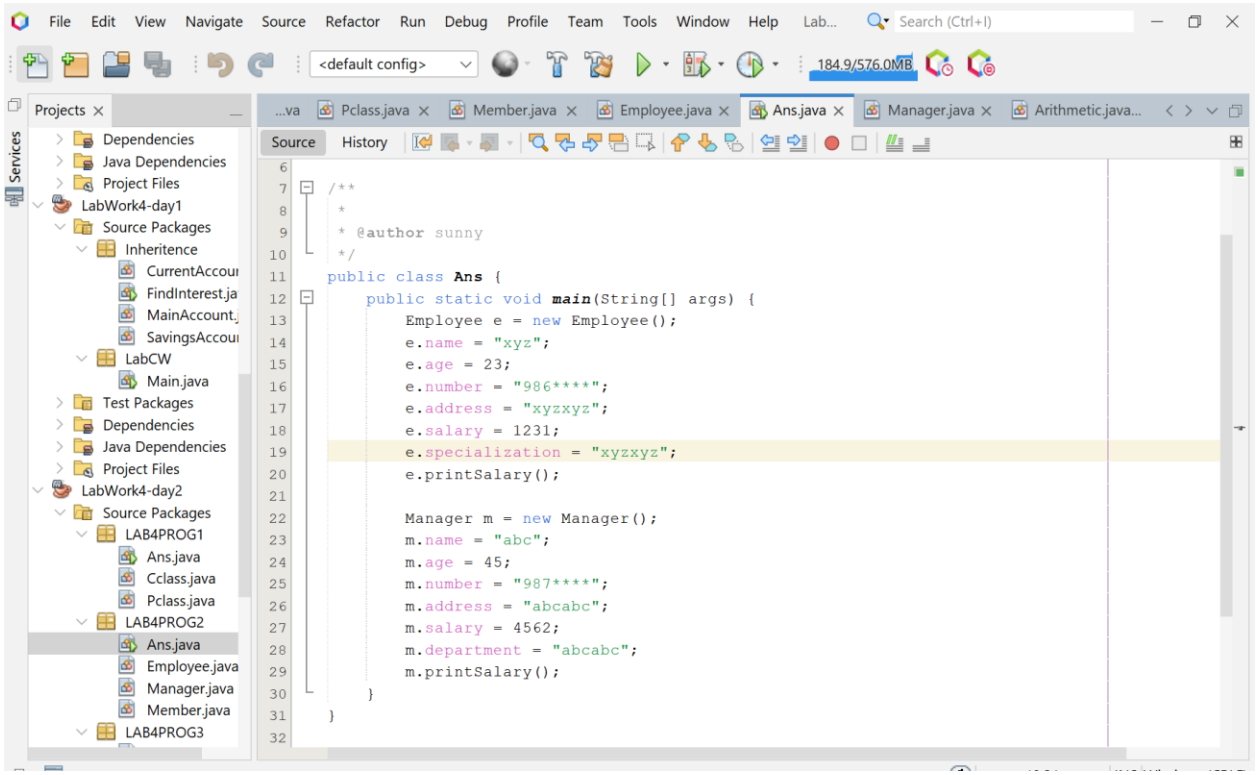
```
1231
4562
```

Screenshot of code edition window:

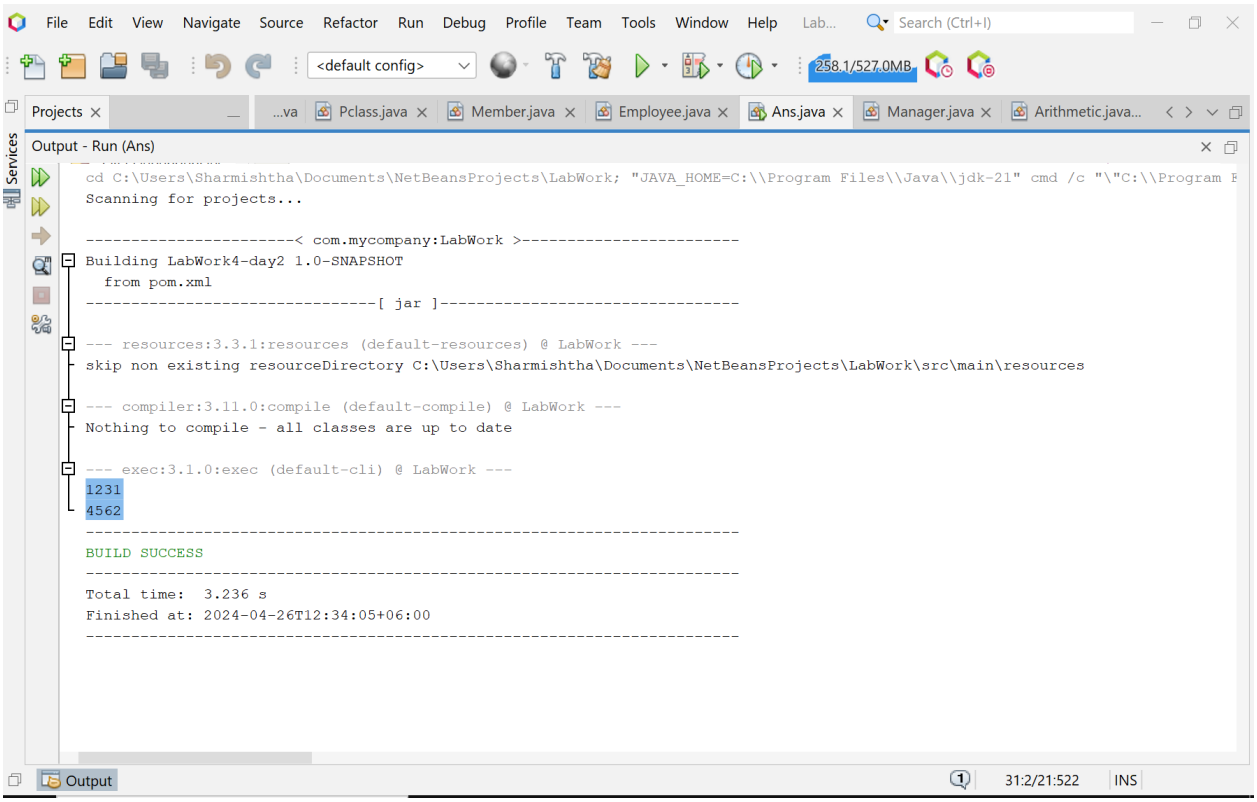





File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Dependencies
Java Dependencies
Project Files
LabWork4-day1
Source Packages
Inheritence
LabCW
Main.java
Test Packages
LabWork4-day2
LAB4PROG1
Ans.java
Cclass.java
Pclass.java
LAB4PROG2
Employee.java
Manager.java
Member.java
LAB4PROG3
Pclass.java
Member.java
Employee.java
Ans.java
Manager.java
Arithmetic.java...
Source
History
/*
* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this
* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
*/
package LAB4PROG2;
/**
*
* @author sunny
*/
class Manager extends Member {
String department;
}
Output
12:8
INS Windows (CRLF)


File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help
Projects
Dependencies
Java Dependencies
Project Files
LabWork4-day1
Source Packages
Inheritence
LabCW
Main.java
Test Packages
LabWork4-day2
LAB4PROG1
Ans.java
Cclass.java
Pclass.java
LAB4PROG2
Employee.java
Manager.java
Member.java
LAB4PROG3
Pclass.java
Member.java
Employee.java
Ans.java
Manager.java
Arithmetic.java...
Source
History
/**
*
* @author sunny
*/
public class Ans {
public static void main(String[] args) {
Employee e = new Employee();
e.name = "xyz";
e.age = 23;
e.number = "986****";
e.address = "xyzxyz";
e.salary = 1231;
e.specialization = "xyzxyz";
e.printSalary();
Manager m = new Manager();
m.name = "abc";
m.age = 45;
m.number = "987****";
m.address = "abcabc";
m.salary = 4562;
m.department = "abcabc";
m.printSalary();
}
}

Screenshot of Output screen/Run screen window:



Explanation :

The program begins by defining a class named Member (parent class) with data members name, age, number, address, and salary. It also includes a method printSalary that prints the salary of the members. Then, it defines two subclasses named Employee and Manager that extend Member. These classes inherit all the properties and methods of Member and also define their own member's specialization and department respectively. In the main method, an object e of Employee and an object m of Manager are created. Then, values are assigned to the members of e and m. Finally, the printSalary method is called on e and m to print their salaries.

Discussion:

This program effectively demonstrates the basic principles of Object-Oriented Programming (OOP), specifically inheritance, where subclasses inherit properties and behaviors from a superclass, and can also provide their own unique behaviors. The Member class represents a general member with common properties like name, age, number, address, and salary, and a common behavior printSalary. The Employee and Manager classes represent specific types of members with additional properties specialization and department respectively. By creating objects of Employee and Manager and assigning values to their properties, the program models a real-world scenario where different types of members have different properties but share some common properties and behaviors. The program then prints the salaries of an employee and a manager, demonstrating how the common behavior printSalary can be accessed by both Employee and Manager. This program is a simple yet powerful demonstration of these fundamental OOP concepts. It's a great starting point for anyone learning Java or OOP.

Problem 03: Write a Java program that includes an Arithmetic class with an add method and an Adder class that inherits from Arithmetic. The add method should take two integers as parameters and return their sum.

Objective:

The program's goal is to demonstrate the use of Java inheritance and method overriding by performing addition operations.

Algorithm:

- Start the program.
- Define the Arithmetic class:
  - This class has a method named add that takes two integer parameters a and b.
  - The add method calculates the sum of a and b and returns the result.
- Define the Adder class:
  - This class extends the Arithmetic class, inheriting its add method.
  - The Adder class also has a method named callAdd that takes two integer parameters a and b.
  - The callAdd method calls the inherited add method with a and b as arguments and returns the result.
- In the main method of the Solution class:
  - Create a new Adder object.
  - Print the name of the superclass of the Adder class.
  - Call the add method of the Adder object with the arguments 10 and 32, and print the returned value.
  - Call the add method of the Adder object with the arguments 10 and 3, and print the returned value.
  - Call the add method of the Adder object with the arguments 10 and 10, and print the returned value.
- End the program.

Code:

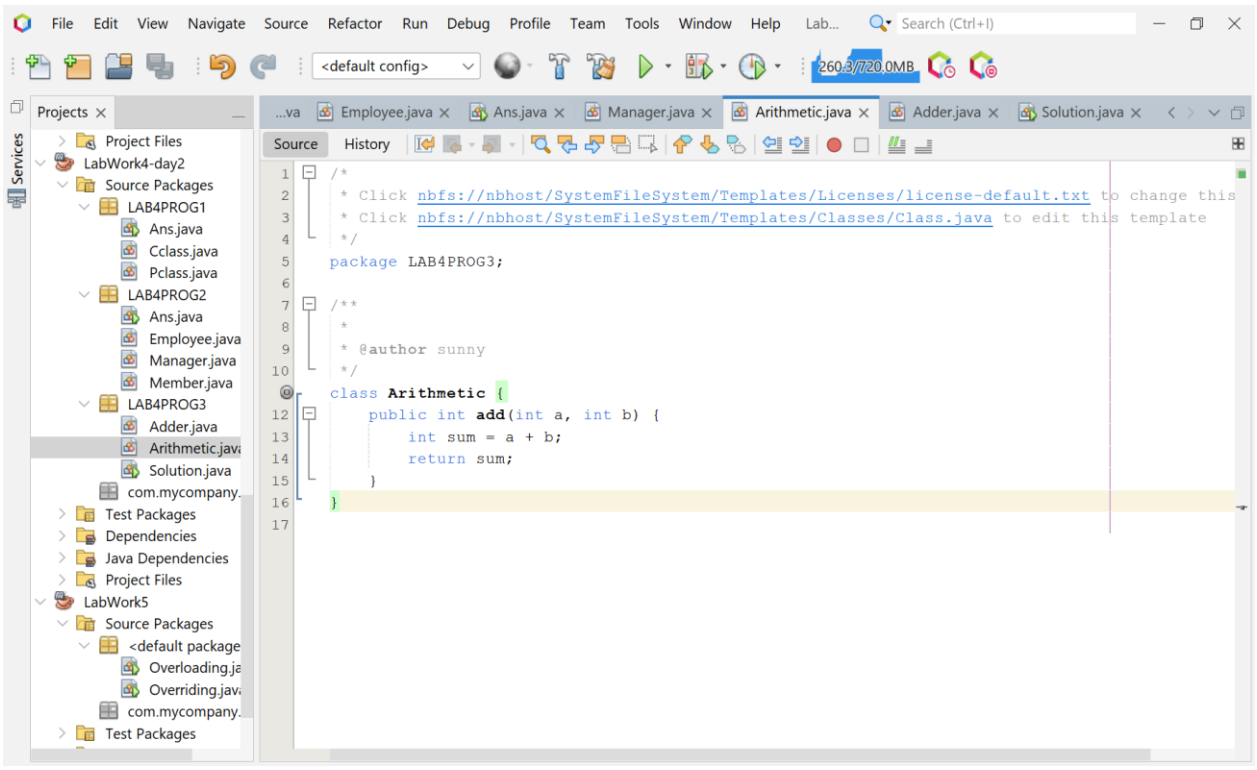
```
class Arithmetic {

   public int add(int a, int b) {

      int sum = a + b;

      return sum;

   }

}
```

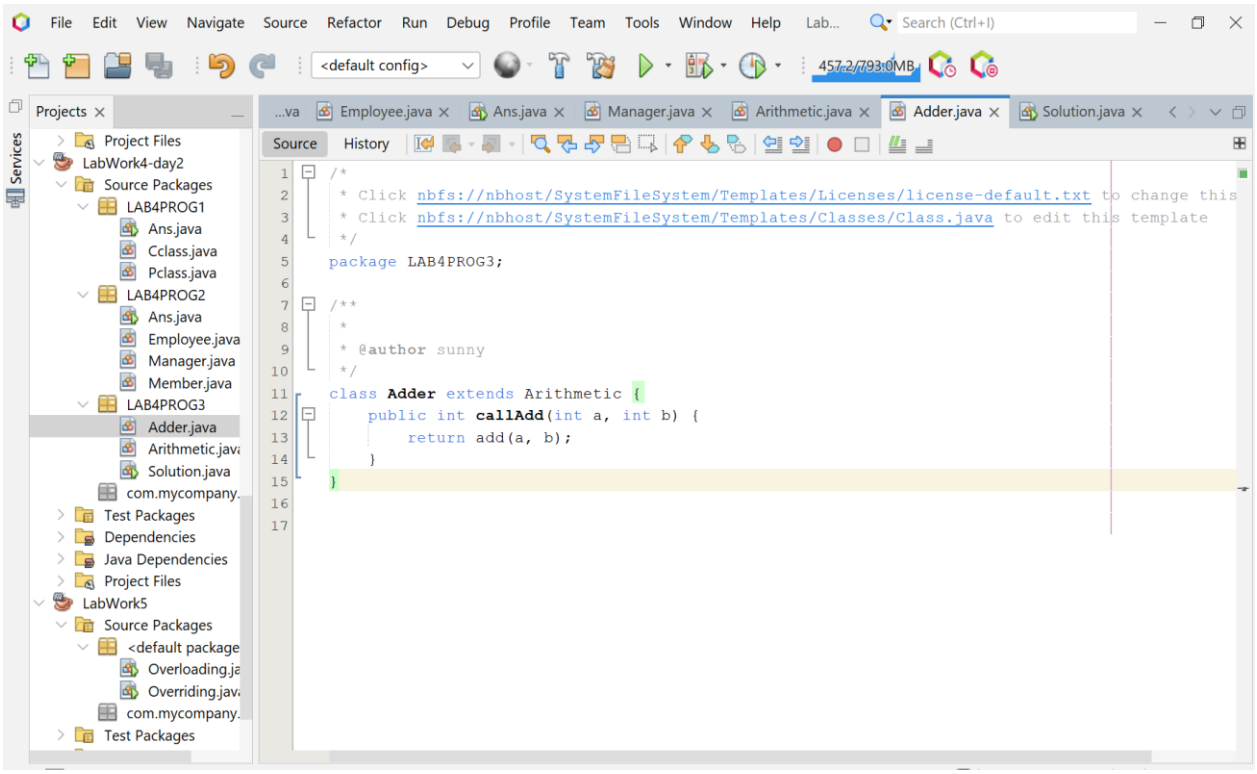
```
class Adder extends  Arithmetic {
   public int callAdd(int a, int b) {
      return add(a, b);
   }
}

public class Solution {
   public static void main(String[] args) {
      // Create a new Adder object
      Adder a = new Adder();
      System.out.println("My superclass is: " + a.getClass().getSuperclass().getSimpleName());
      // Print the result of 3 calls to Adder's `add(int,int)` method as 3 space-separated integers
      System.out.print(a.add(10, 32) + " " + a.add(10, 3) + " " + a.add(10, 10) + "\n");
   }
}
```

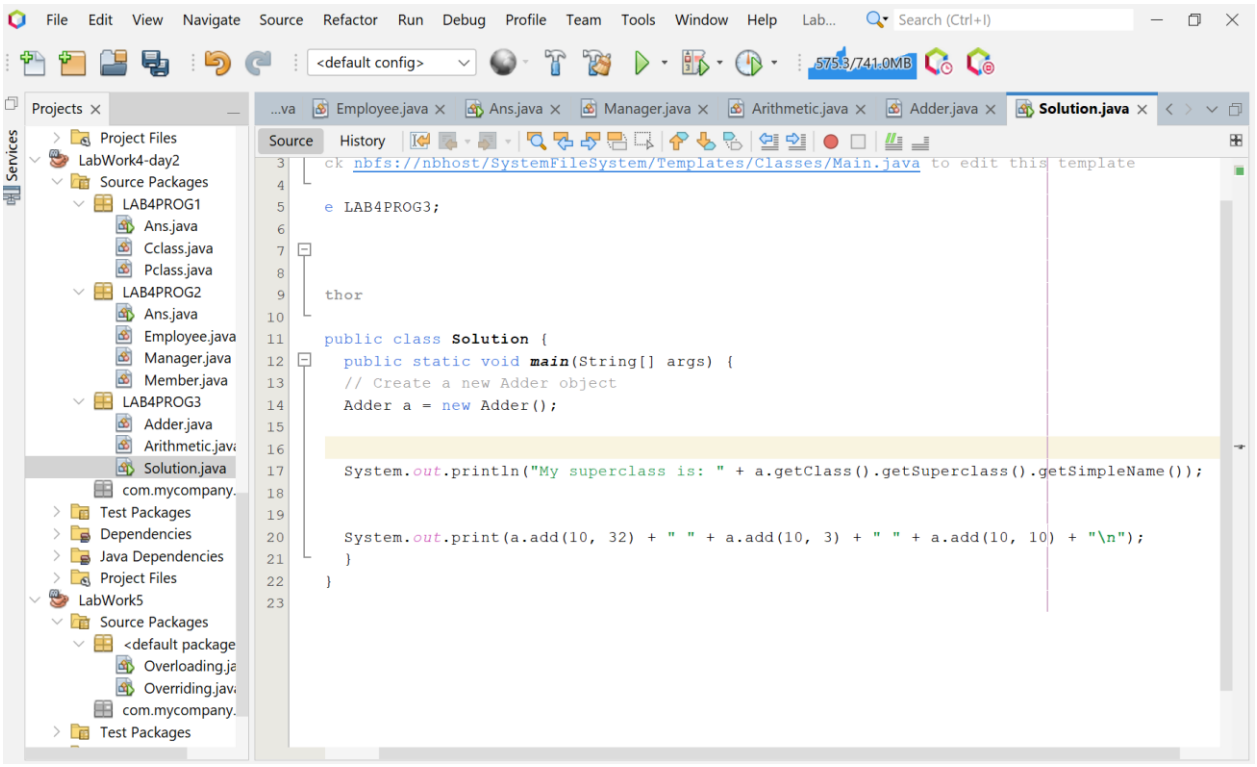
Input/ Expected Output:

```
My superclass is: Arithmetic
42 13 20
```

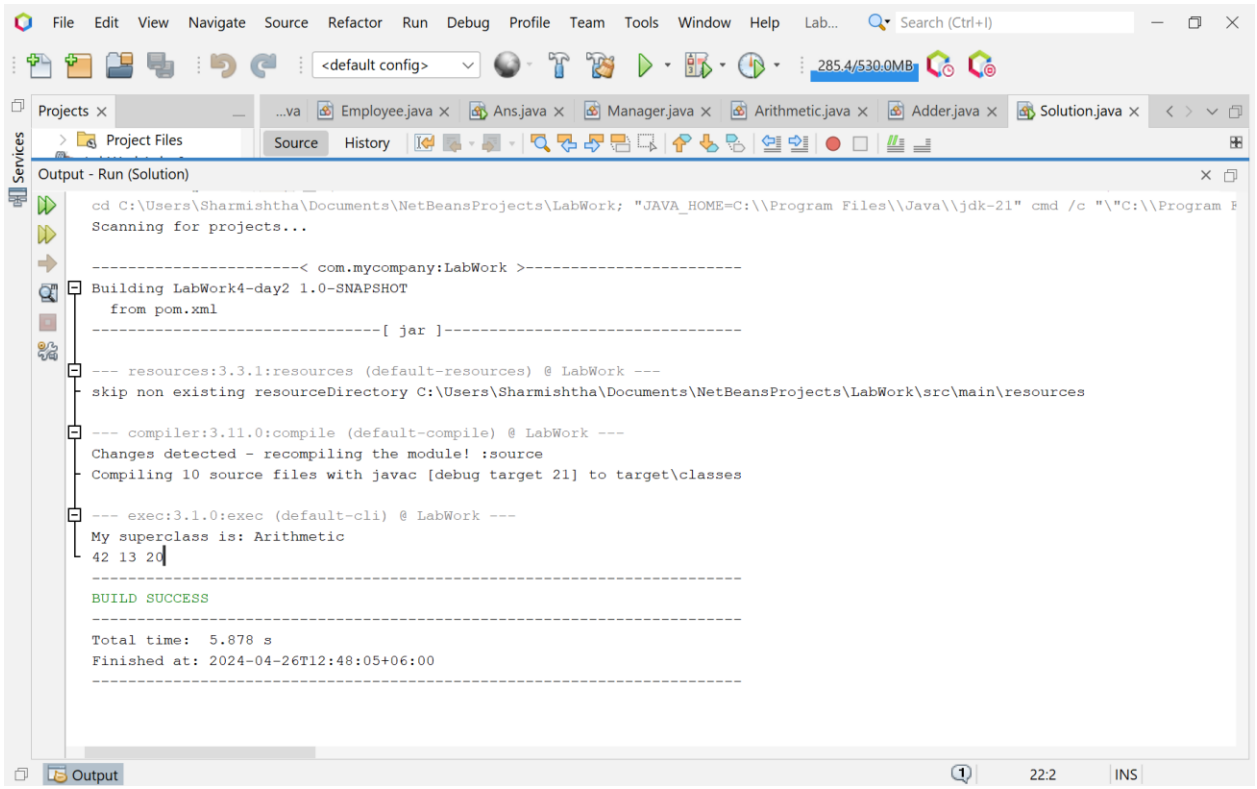
Screenshot of code edition window:







Screenshot of Output screen/Run screen window:



Explanation :

The Java program is structured around two classes: Arithmetic and Adder. The Arithmetic class has a single method, add, which takes two integers as parameters and returns their sum. The Adder class extends Arithmetic, inheriting it's add method. It also defines a method callAdd that takes two integers as parameters, calls the inherited add method with these parameters, and returns the result.
In the main method of the Solution class, an instance of Adder is created. The program then prints the name of the superclass of Adder using the getSimpleName method, which returns the simple name of the underlying class as given in the source code. Following this, the add method of the Adder object is called three times with different sets of arguments, and the results are printed to the console.

Discussion:

This program effectively demonstrates the concept of inheritance in Java. Inheritance is a fundamental principle of object-oriented programming where one class acquires the properties (methods and fields) of another. With the use of inheritance, information is managed in a hierarchical manner.
In this program, the Adder class is a subclass of the Arithmetic class, and it inherits the add method from Arithmetic. This is a clear demonstration of code reuse, one of the significant advantages of inheritance.
However, the program could be enhanced in several ways. For instance, it currently only supports addition operations. It could be extended to support other arithmetic operations like subtraction, multiplication, and division. Additionally, the program does not handle any potential errors, such as non-integer input. Incorporating exception handling could make the program more robust and user-friendly.
Overall, this program serves as a good starting point for understanding inheritance in Java, but there is room for further exploration and improvement.