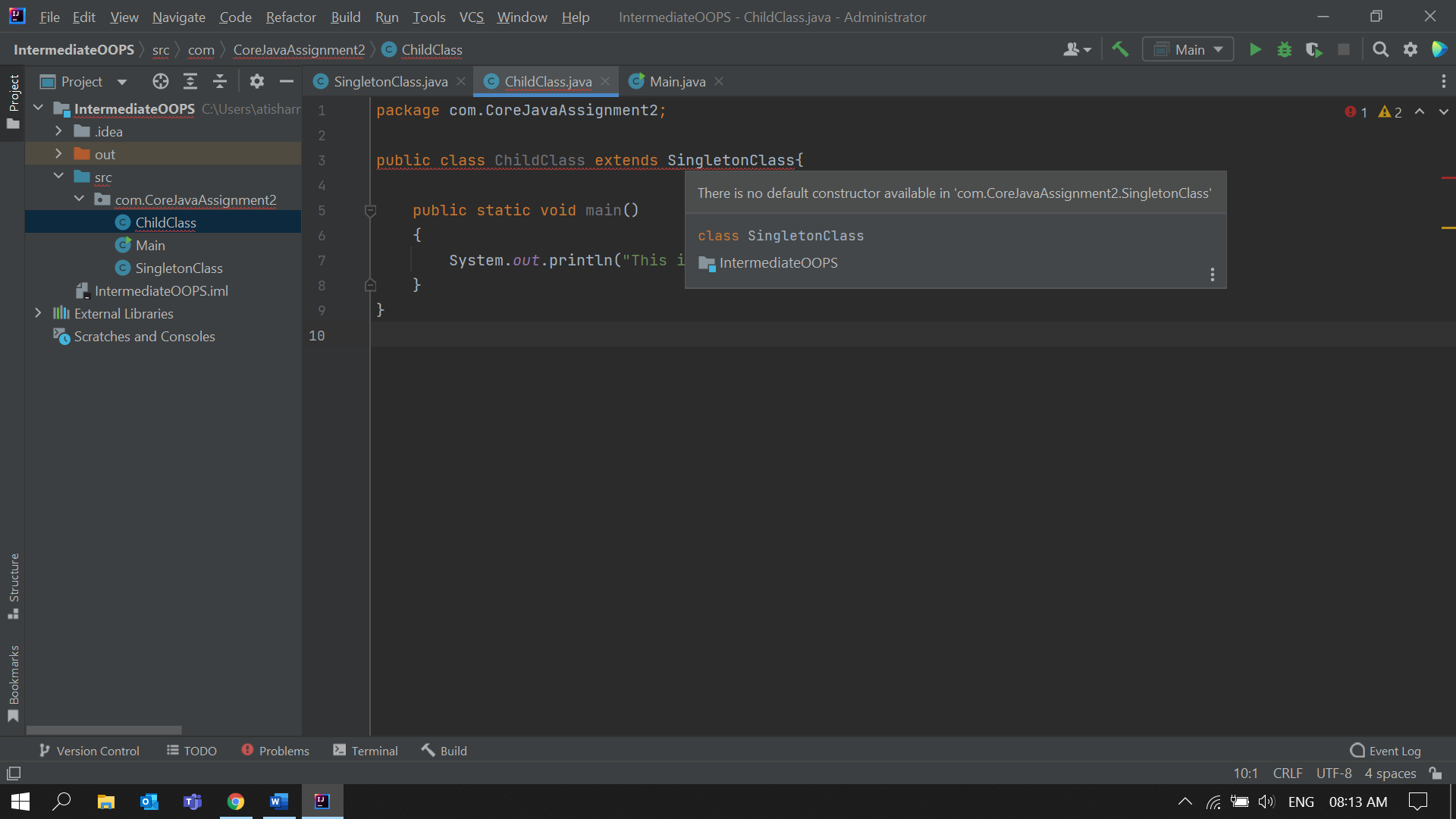
**CORE JAVA ASSIGNMENT 2 – Intermediate OOPS**

1. Write a singleton class. Confirm that singleton class cannot be inherited

**Solution:**

package com.CoreJavaAssignment2;  
  
class SingletonClass {  
 private static SingletonClass *instance*;  
  
 private SingletonClass()  
 {  
 System.*out*.println("This is the Constructor from Singleton Class");  
 }  
  
 public static SingletonClass getInstance()  
 {  
 if(*instance* == null)  
 {  
 *instance* = new SingletonClass();  
 }  
 return *instance*;  
 }  
}  
  
**In ChildClass.java**



2. Write a program that describes the hierarchy of an organization. Here we need to write 3 classes Employee, Manager & Labour where Manager & Labour are the sub classes of the Employee. Manager has incentive & Labour has over time. Add the functionality to calculate total salary of all the employees. Use polymorphism i.e. method overriding.

**Solution:**Employee.java

package com.OrganizationHierarchy;  
  
public class Employee {  
  
 String name;  
 int age;  
 double salary;  
 public Employee(String name , int age , double salary )  
 {  
 this.name = name;  
 this.age = age;  
 this.salary = salary;  
 }  
 void getTotalSalary()  
 {  
 System.*out*.println("Total Salary of Employee is : " + salary );  
 }  
}  
  
Manager.java

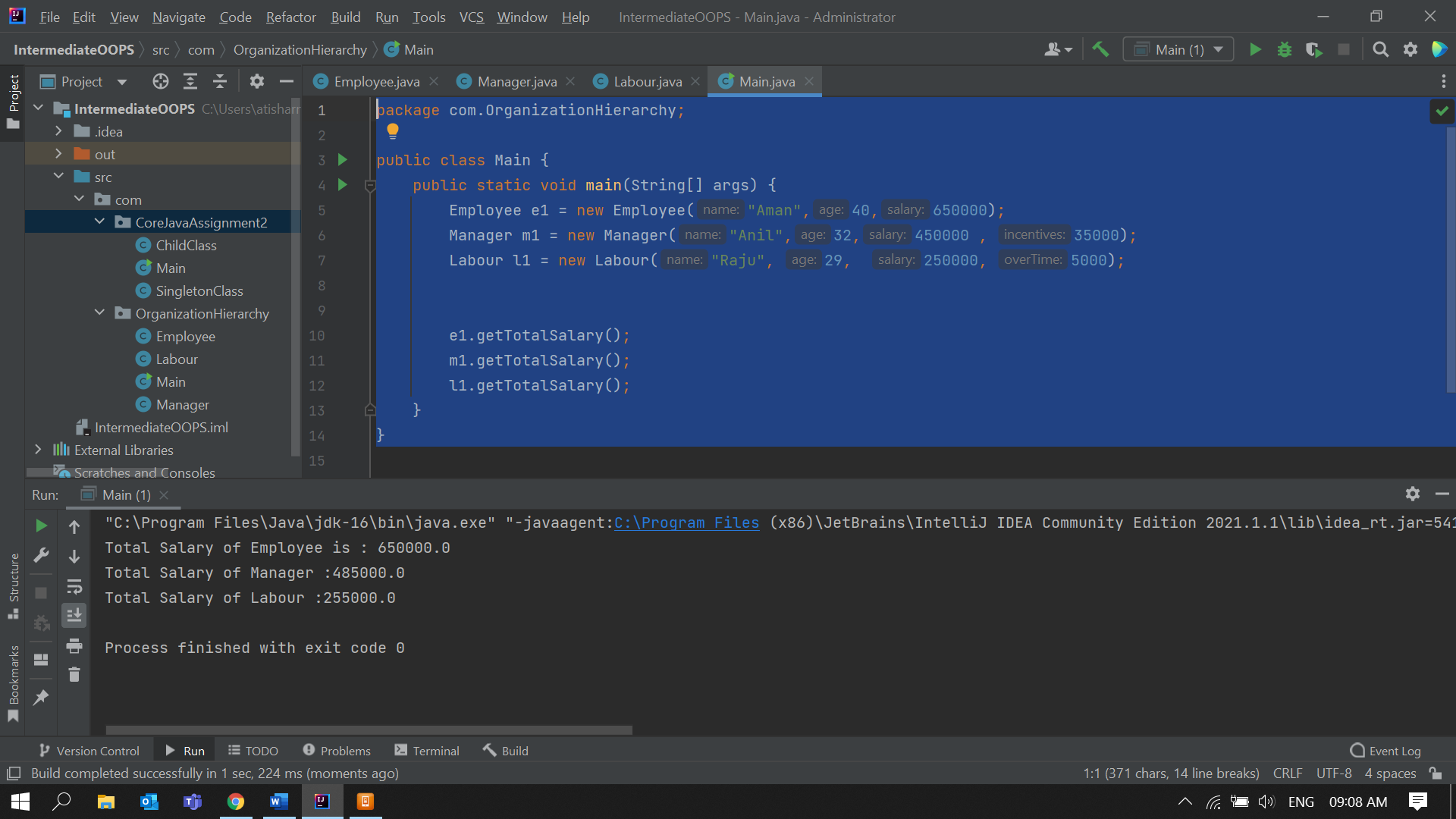
package com.OrganizationHierarchy;  
  
public class Manager extends Employee  
{  
 double incentives;  
 public Manager(String name , int age , double salary , double incentives)  
 {  
 super(name , age , salary);  
 this.incentives = incentives;  
 }  
 void getTotalSalary()  
 {  
 this.salary += incentives;  
 System.*out*.println("Total Salary of Manager :" + salary);  
 }  
  
}

Labour.java

package com.OrganizationHierarchy;  
  
public class Labour extends Employee{  
  
 double overTime;  
  
 public Labour(String name , int age , double salary , double overTime)  
 {  
 super(name , age , salary);  
 this.overTime = overTime;  
 }  
  
 void getTotalSalary()  
 {  
 this.salary += overTime;  
 System.*out*.println("Total Salary of Labour :" + salary);  
 }  
}

Main.java  
  
package com.OrganizationHierarchy;  
  
public class Main {  
 public static void main(String[] args) {  
 Employee e1 = new Employee("Aman",40,650000);  
 Manager m1 = new Manager("Anil",32,450000 , 35000);  
 Labour l1 = new Labour("Raju", 29, 250000, 5000);  
  
  
 e1.getTotalSalary();  
 m1.getTotalSalary();  
 l1.getTotalSalary();  
 }  
}

Output:

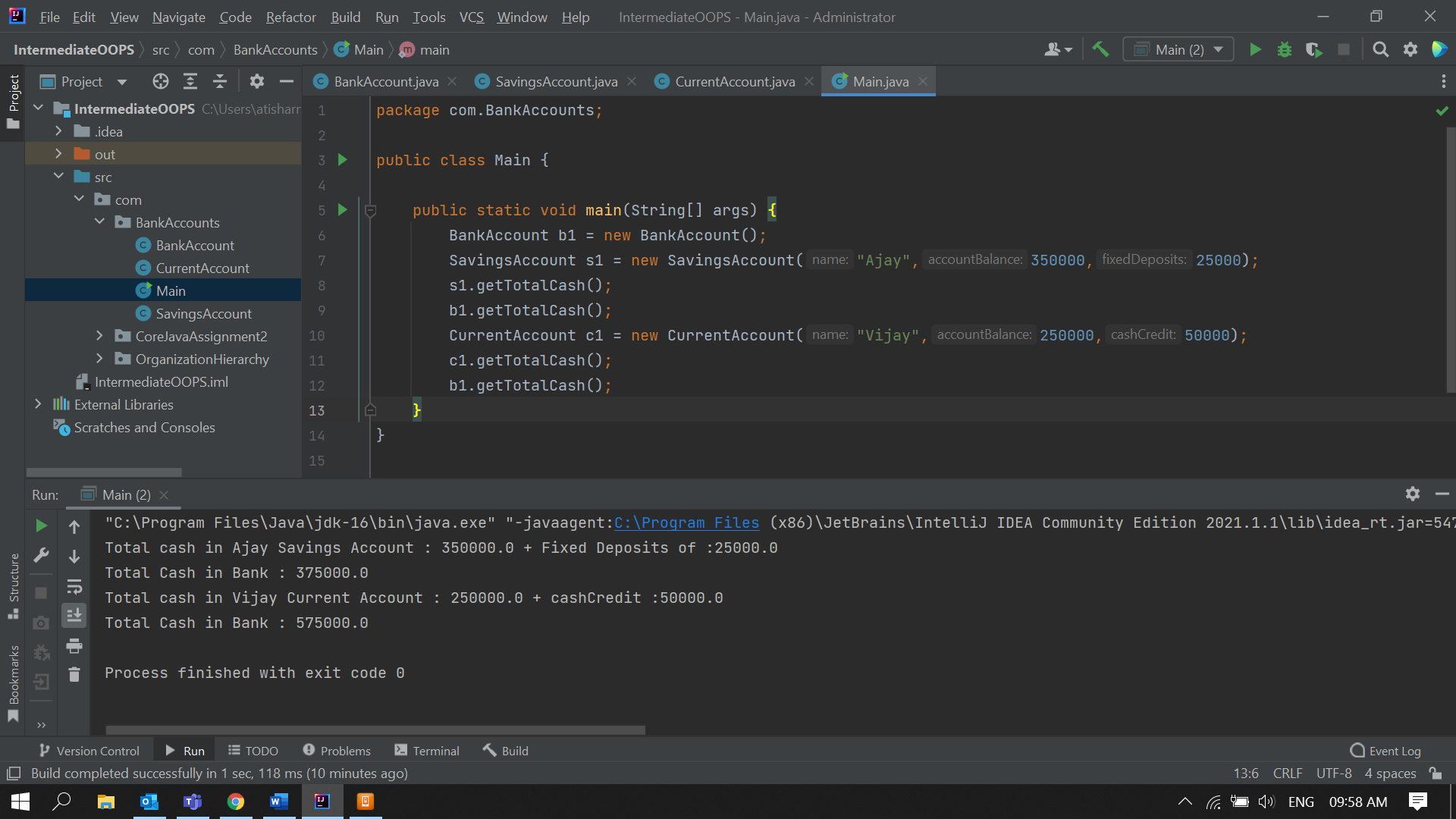
  
  
  
3. Write a program to consider saving & current account in the bank. Saving account holder has

'Fixed Deposits' whereas Current account holder has cash credit. Apply polymorphism to find

out total cash in the bank.  
  
**Solution:**BankAccount.java  
  
package com.BankAccounts;  
  
public class BankAccount {  
 String name;  
 double accountBalance;  
 static double *totalCashInBank*;  
 public void getTotalCash()  
 {  
 System.*out*.println("Total Cash in Bank : " + *totalCashInBank*);  
 }  
}  
  
SavingsAccount.java  
  
package com.BankAccounts;  
public class SavingsAccount extends BankAccount{  
 double fixedDeposits;  
 public SavingsAccount(String name, double accountBalance , double fixedDeposits )  
 {  
 this.name = name;  
 this.accountBalance = accountBalance;  
 this.fixedDeposits = fixedDeposits;  
 BankAccount.*totalCashInBank* += this.fixedDeposits + this.accountBalance;  
 }  
 public void getTotalCash()  
 {  
 System.*out*.println("Total cash in " + this.name + " Savings Account : " + accountBalance + " + FixedDeposits of :" + fixedDeposits);  
 }  
}

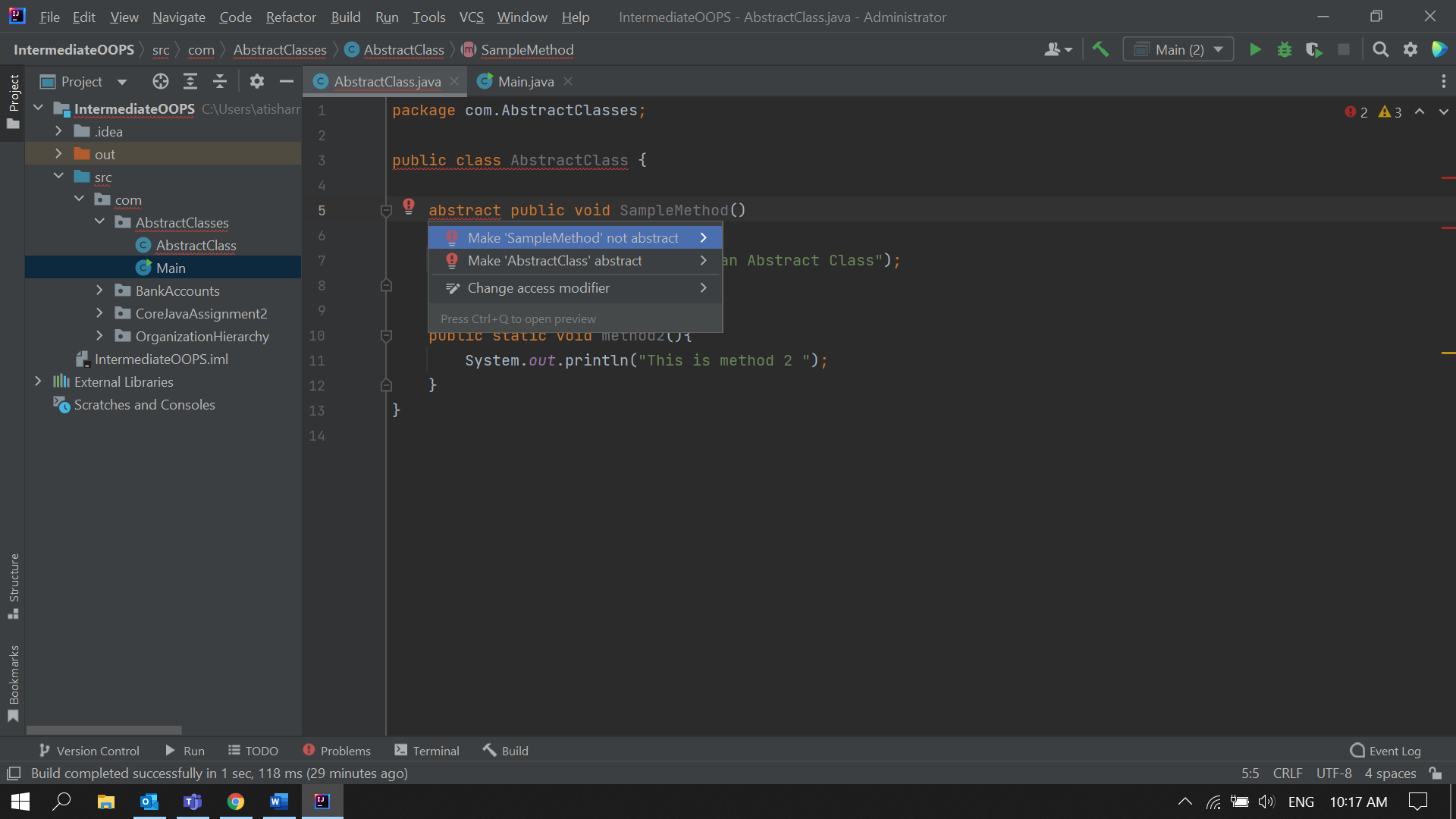
CurrentAccount.java

package com.BankAccounts;  
public class CurrentAccount extends BankAccount{  
 double cashCredit;  
 public CurrentAccount(String name, double accountBalance , double cashCredit )  
 {  
 this.name = name;  
 this.accountBalance = accountBalance;  
 BankAccount.*totalCashInBank* += this.accountBalance ;  
 this.cashCredit = cashCredit;  
 BankAccount.*totalCashInBank* -= this.cashCredit ;  
 }  
 public void getTotalCash()  
 {  
 System.*out*.println("Total cash in " + this.name + " Current Account : " + accountBalance + " + cashCredit :" +cashCredit);  
 }  
}  
  
Main.java  
  
package com.BankAccounts;  
public class Main {  
 public static void main(String[] args) {  
 BankAccount b1 = new BankAccount();  
 SavingsAccount s1 = new SavingsAccount("Ajay",350000,25000);  
 s1.getTotalCash();  
 b1.getTotalCash();  
 CurrentAccount c1 = new CurrentAccount("Vijay",250000,50000);  
 c1.getTotalCash();  
 b1.getTotalCash();  
 }  
}  
  
  
  
  
  
  
  
  
  
  
  
  
  
**Output:**



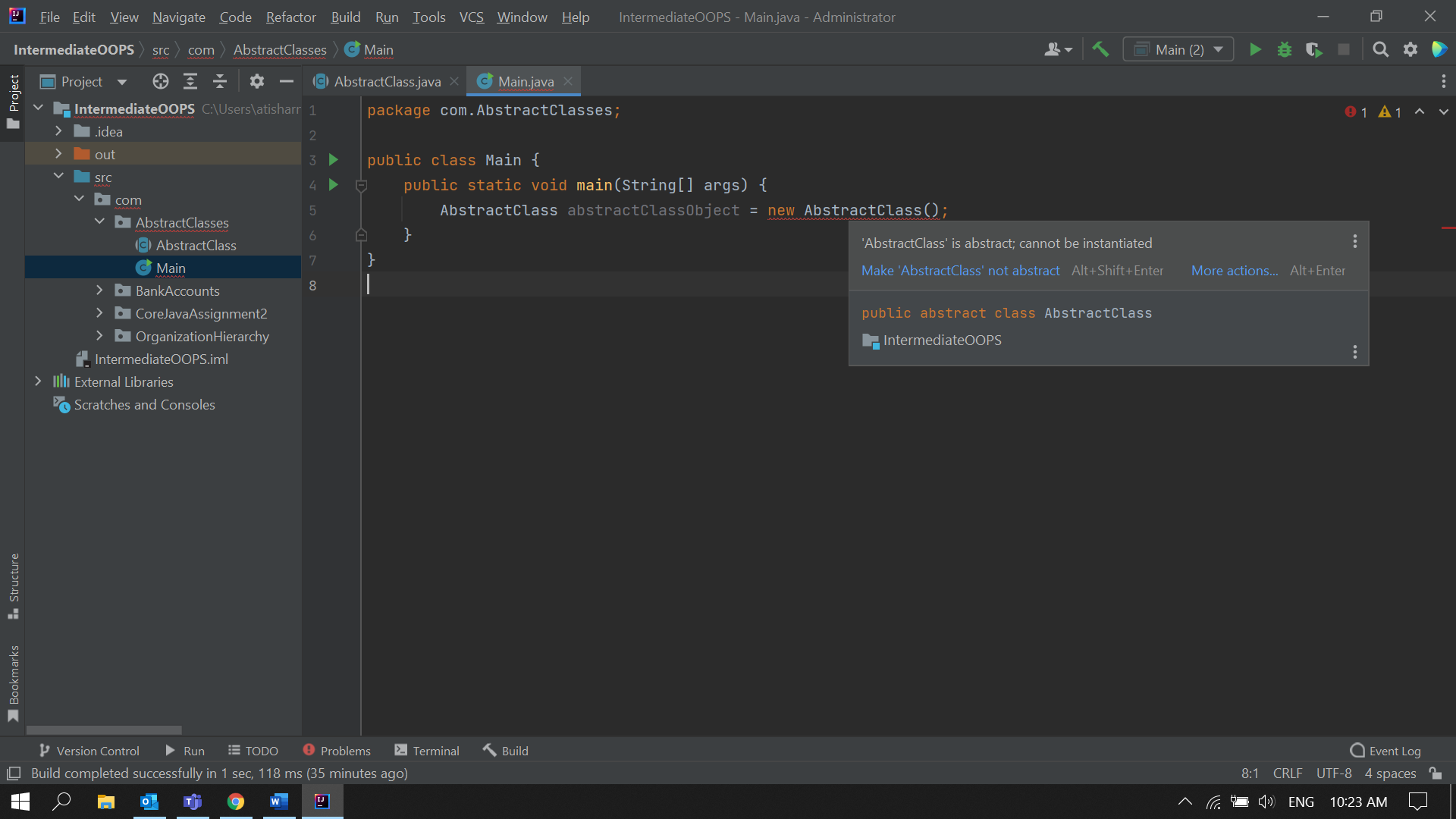
4. Test the following principles of an abstract class:   
  
• If any class has any of its method abstract then you must declare entire class abstract.

**Solution:**AbstractClass.javapackage com.AbstractClasses;  
  
public class AbstractClass {  
  
 abstract public void SampleMethod()  
 {  
 System.*out*.println("This is an Abstract Class");  
 }  
   
 public static void method2(){  
 System.*out*.println("This is method 2 ");  
 }  
}  
  
Error:



• Abstract class cannot be instantiated.   
  
**Solution:**  
  
AbstractClass.java  
  
package com.AbstractClasses;  
  
abstract public class AbstractClass {  
  
 public void SampleMethod()  
 {  
 System.*out*.println("This is an Abstract Class");  
 }  
  
 public static void method2(){  
 System.*out*.println("This is method 2 ");  
 }  
}

Main.java  
  
package com.AbstractClasses;  
  
public class Main {  
 public static void main(String[] args) {  
 AbstractClass abstractClassObject = new AbstractClass();  
 }  
}  
  
**Error:**



• When we extend an abstract class, we must either override all the abstract methods in sub class or declare subclass as abstract.   
  
**Solution:**  
**Case 1: Overrided all the abstract methods of AbstractClass in ChildClass.**  
   
AbstractClass.java

package com.AbstractClasses;  
  
public abstract class AbstractClass {

AbstractClass(){  
 System.*out*.println("AbstractClass object is created");  
 }

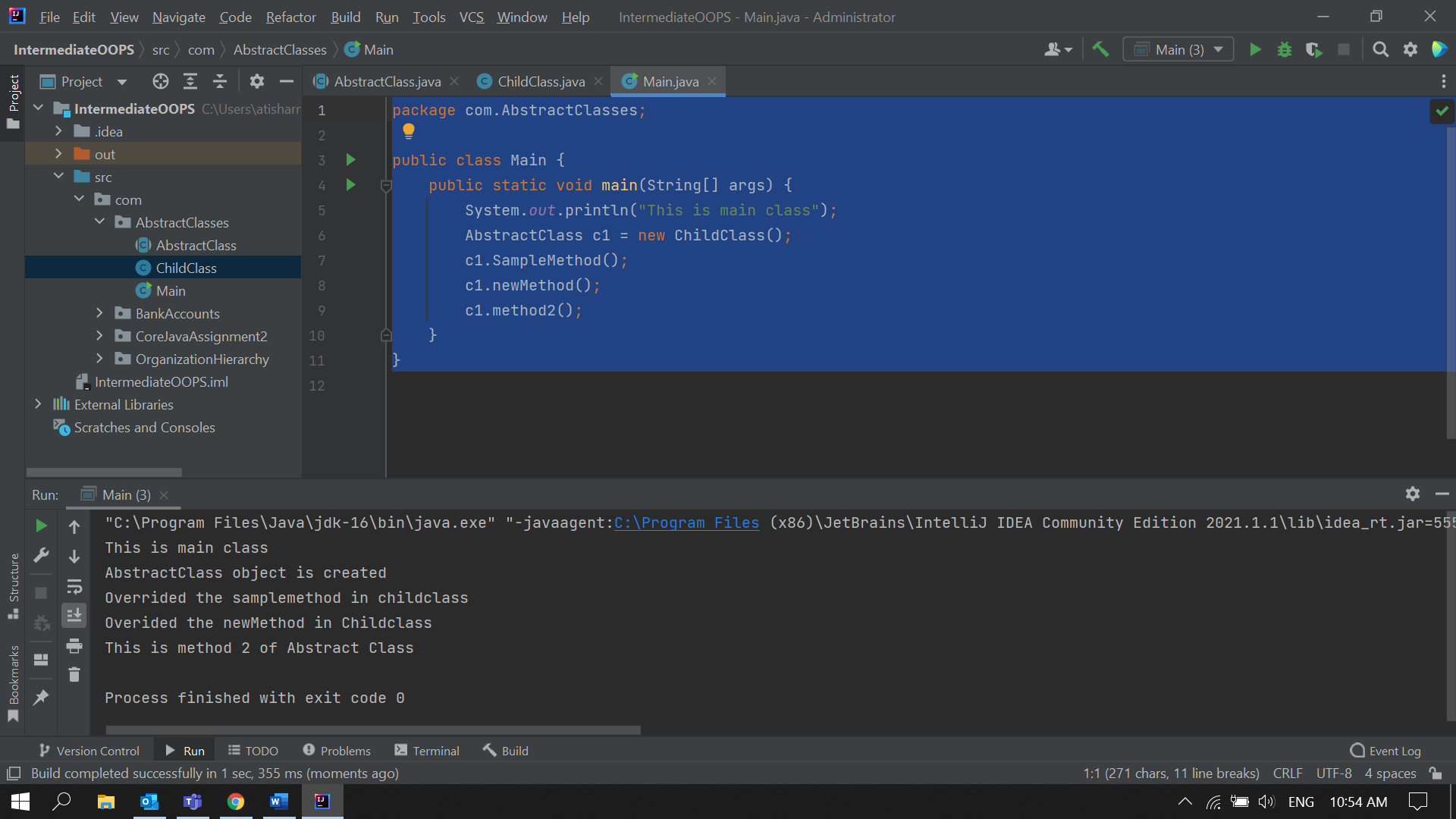
abstract public void SampleMethod();

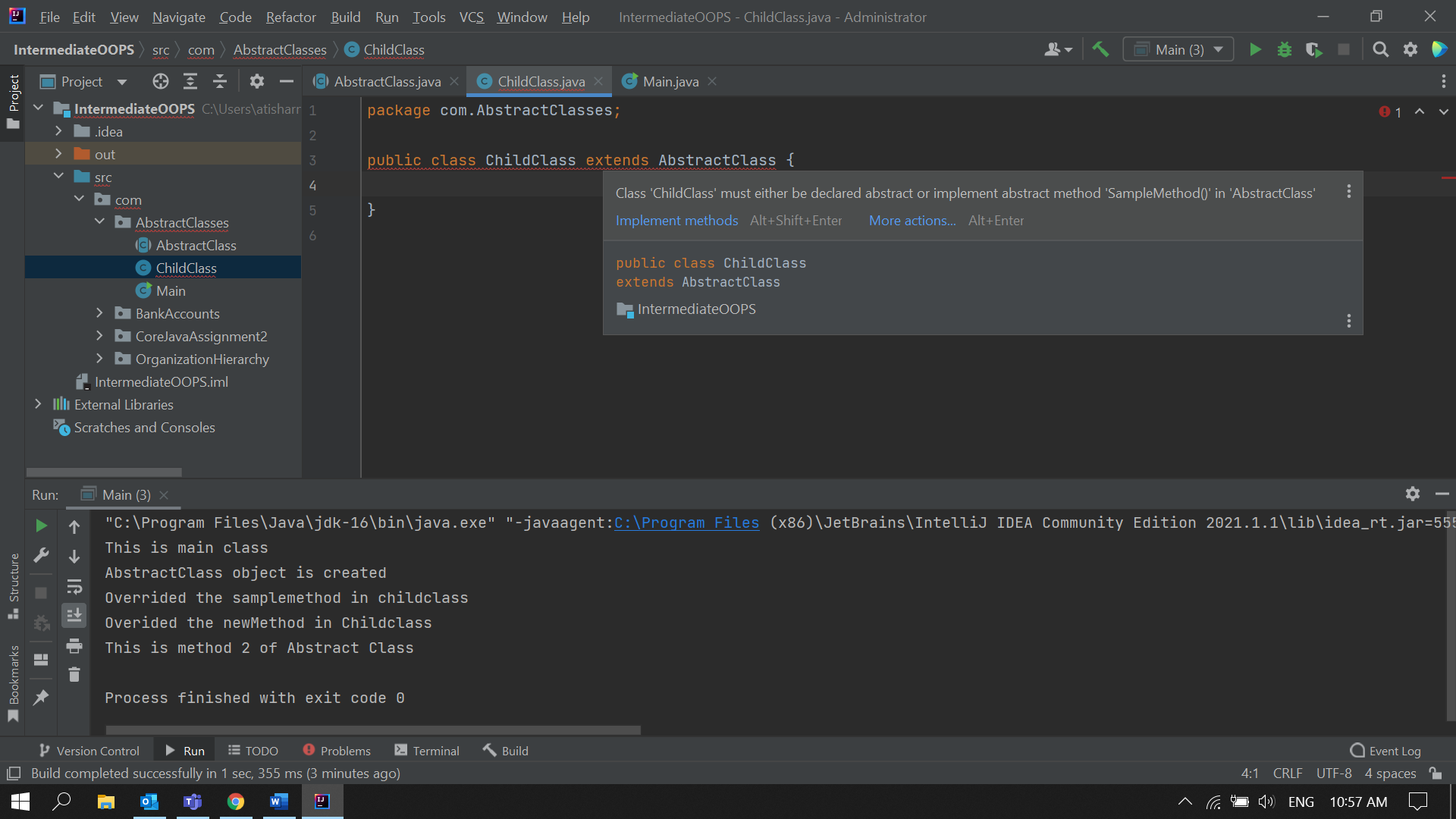
abstract public void newMethod();

public void method2(){  
 System.*out*.println("This is method 2 of Abstract Class");  
 }  
}

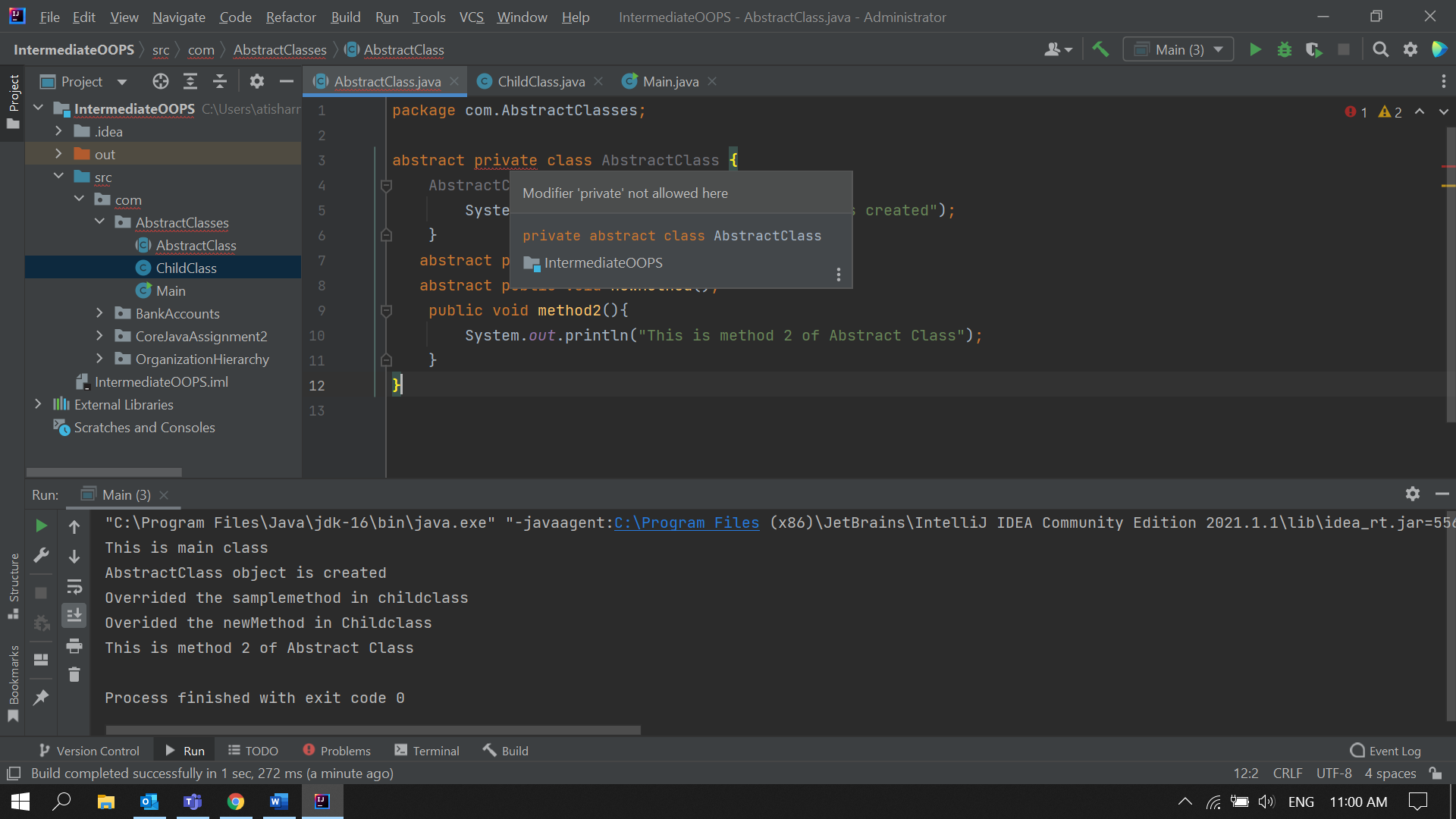
ChildClass.java  
  
package com.AbstractClasses;  
  
public class ChildClass extends AbstractClass {  
  
 @Override  
 public void SampleMethod() {  
 System.*out*.println("Overrided the samplemethod in childclass");  
 }  
 @Override  
 public void newMethod()  
 {  
 System.*out*.println("Overided the newMethod in Childclass");  
 }  
  
}  
  
  
Main.java  
  
package com.AbstractClasses;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("This is main class");  
 AbstractClass c1 = new ChildClass();  
 c1.SampleMethod();  
 c1.newMethod();  
 c1.method2();  
 }  
}

**Output:**

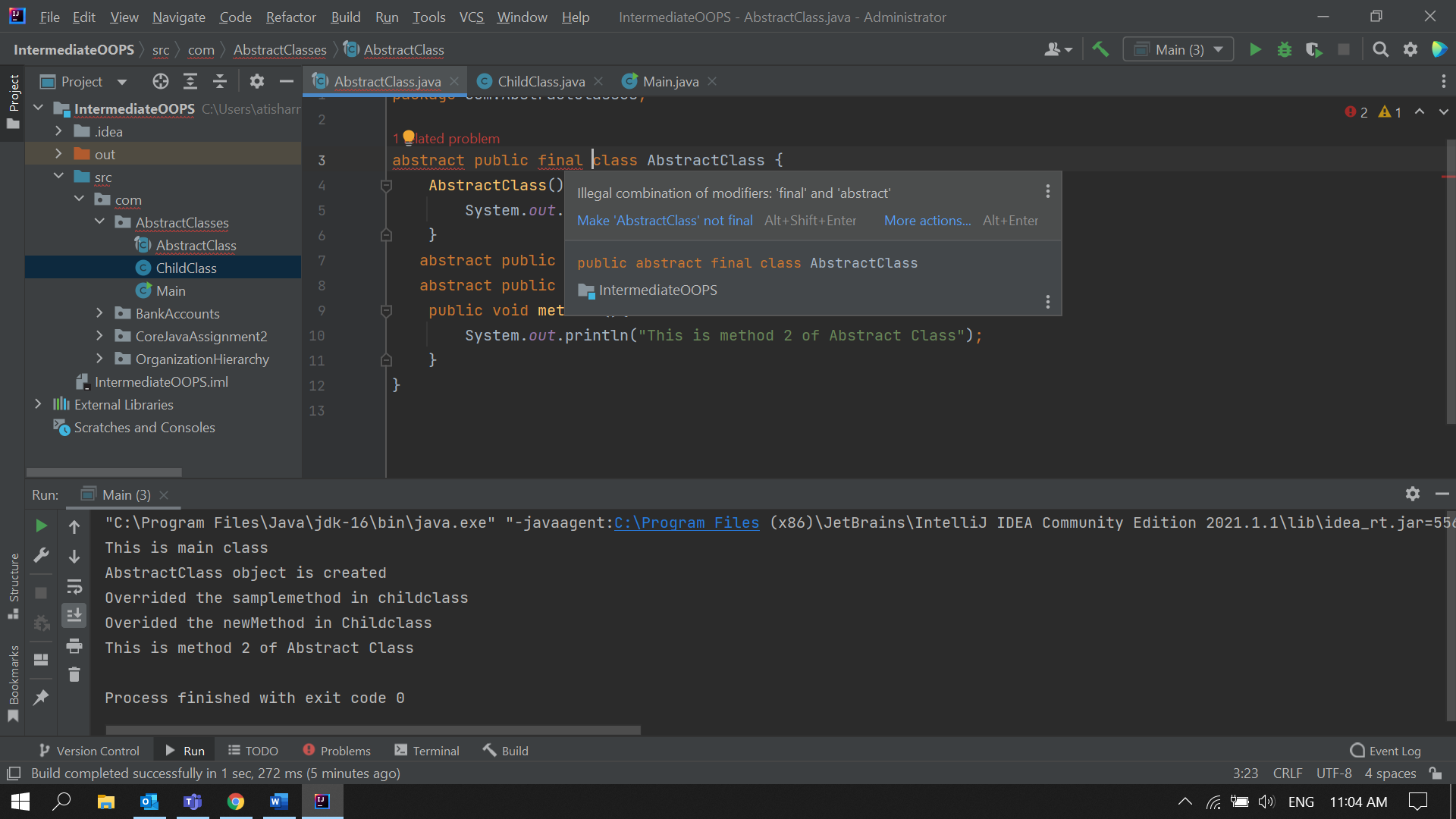
  
  
  
**Case 2: Not Overrided any abstract methods of AbstractClass in ChildClass.**  
  
**Output:**



• Abstract class cannot be private.



• Abstract class cannot be final.

  
  
• You can declare a class abstract without having any abstract method.  
  
**Solution:**  
Abstract.java  
  
package com.AbstractClasses;  
  
abstract public class AbstractClass {  
  
}

ChildClass.java  
  
package com.AbstractClasses;  
  
public class ChildClass extends AbstractClass {  
  
 public static void method1()  
 {  
 System.*out*.println("Method of ChildClass");  
 }  
  
}  
  
  
Main.java  
  
package com.AbstractClasses;  
  
public class Main {  
 public static void main(String[] args) {  
 System.*out*.println("This is main class");  
 AbstractClass c1 = new ChildClass();  
 c1.method1();  
  
 }  
}

**Output:**