**Assignment – 3**

**Sushant Kumar**

**fs\_108\_giet\_046**

**Pin : 17551A0552**

**Github link :-** <https://github.com/Sushant385/RightStrokeFSEngGrads>

1. **What is Inheritance?**

**Inheritance** is a concept in JAVA where one class (Child / Derive class) acquires all the properties and behaivours of another class (Parent / Base class).

For inherit the properties of Parent class there is a keyword called as “**extends**”.

1. **What is Multiple Inheritance?**

When one class extends more than one classes then this is called **multiple inheritance.**

In java, **multiple inheritance** not suppots but by using **interface** we can implement **multiple inheritance.**

1. **What is the use of Super keyword?**

**super** is a keyword in JAVA which is used to call **data member, member function**  of **Super/Prent** class.

* We can call data-member

**super.data\_member\_name;**

* We can call member-function

**super.member\_function\_name();**

1. **What is abstract method?**

**abstract methos** is a method in JAVA which is declared as **abstract** but does not have any implementation. **Abstract method** only can be declared as **public** and **protected** but not as **private.**

1. **What is abstract class?**

A **class** that is declared using **abstract** keyword is known as **abstract class**. In Java **abstract class** cannot be instantiated, which means we cannot create object for abstract classes.It contains abstract methods and also methods with body.

1. **What is the use of final modifier?**

**final** is a keyword in JAVA. It can be used to declare variable, methods, classes.

* If we declare any variable with **final** keyword and once we assigned value to it then we can not change it’s value in further program.
* If we declare any method with **final** keyword then we can not able to override that method.
* If we declare any class with **private** then we can not ectends it by other class.

1. **What is interface? Write the syntax interface.**

An **interface** is just like Java Class, but it only has static constants and abstract method. All methods in an **interface** are implicitly public and abstract.

**Syntax :**

**interface <interface\_name>**

**{**

**// declare constant fields**

**// declare methods that abstract**

**// by default.**

**}**

**Eg :**

**interface Player**

**{**

**final int id = 10;**

**int move();**

**}**

1. **What is package?**

A **package** is a container of related classes and interfaces. It helps to organize your classes into a folder structure and make it easy to locate and use them.

**Declaration :**

**Package package\_name ;**

**Importing :**

**Import package\_name.class\_name ;**

1. **What is exception?**

**Exceptions** are the problems which can occur at runtime and compile time and stops the flow of a program.These exceptions are handled and continues the normal flow of execution.

**Exceptions** are divided into two categories :

* **checked** **exceptions**
* **unchecked** **exceptions**

1. **What is the use of finally block?**

A **finally block** is used to execute all the crucial statements that must be executed whether exception occurs or not.If we declare any block as **finally** then it will be executed at runtime.

**PROGRAMS**

1. **Create a class Publication with data members title(String) and price(int). From this class derive two classes Book and CD. Class Book adds pages(int) and CD adds Size(int). Each of these classes should have constructors and display(). Write a java program to implement this using super, this and method overriding concepts.**

**package** com.rsc.babystepprogram;

**import** java.util.\*;

**public** **class** InheritanceConcept {

**public** **static** **void** main(String[] args)

{

Scanner sc = **new** Scanner(System.***in***);

String t = sc.next();

**int** pg = sc.nextInt();

**int** pr = sc.nextInt();

**int** s = sc.nextInt();

CD cd = **new** CD(t,pr,pg,s);

cd.display();

}

}

**class** Publication

{

String title;

**int** price;

Publication(String title,**int** price)

{

**this**.title = title;

**this**.price = price;

}

**public** **void** display()

{

System.***out***.println("The Title of book is : "+ title);

System.***out***.println("Price : " + price);

}

}

**class** Book **extends** Publication

{

**int** page;

Book(String title, **int** price ,**int** page)

{

**super**(title,price);

**this**.page = page;

}

**public** **void** display()

{

**super**.display();

System.***out***.println("Pages : " + page);

}

}

**class** CD **extends** Book

{

**int** size;

CD(String title,**int** price,**int** page,**int** size)

{

**super**(title , price, page);

**this**.size = size;

}

**public** **void** display()

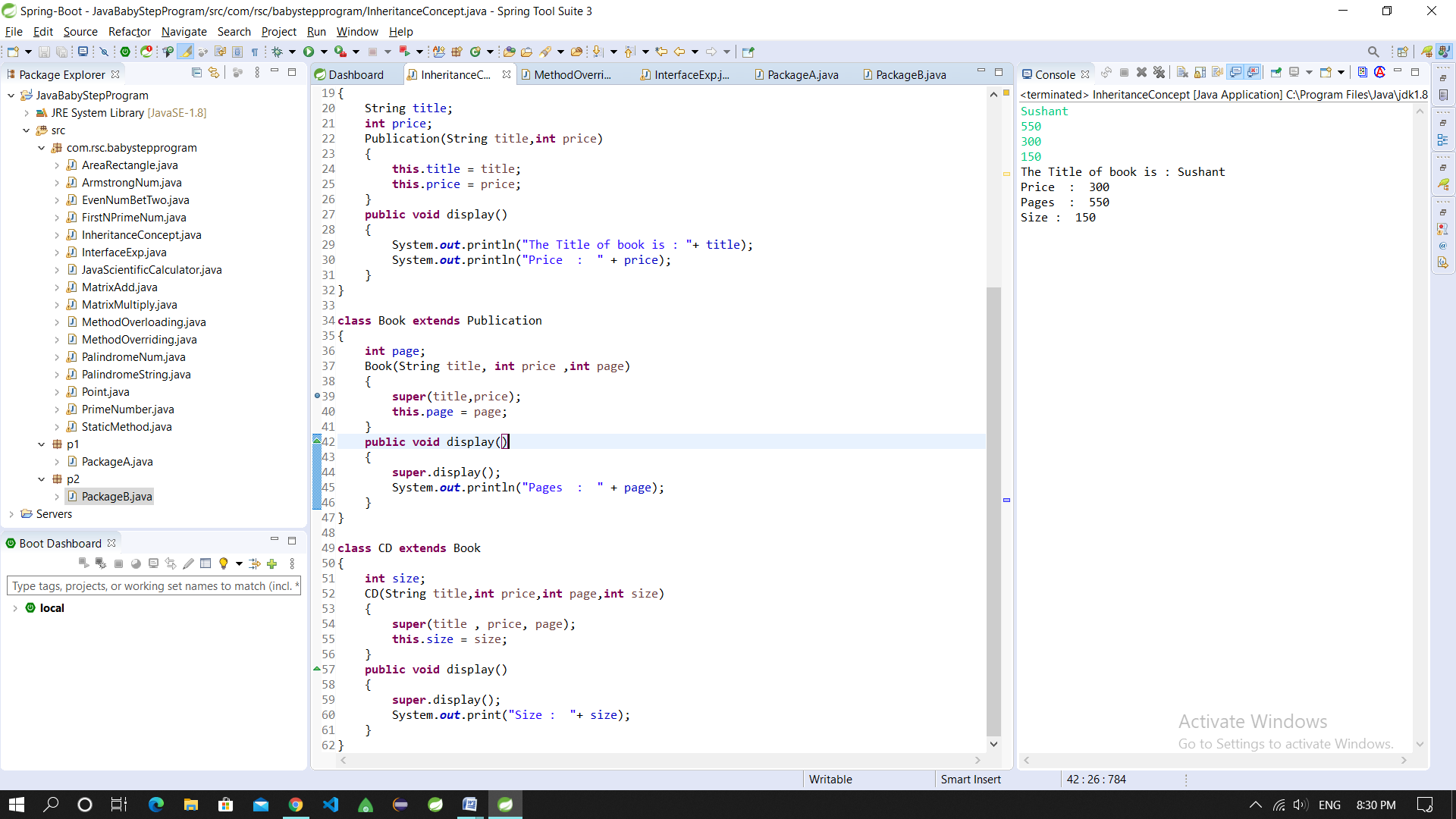
{

**super**.display();

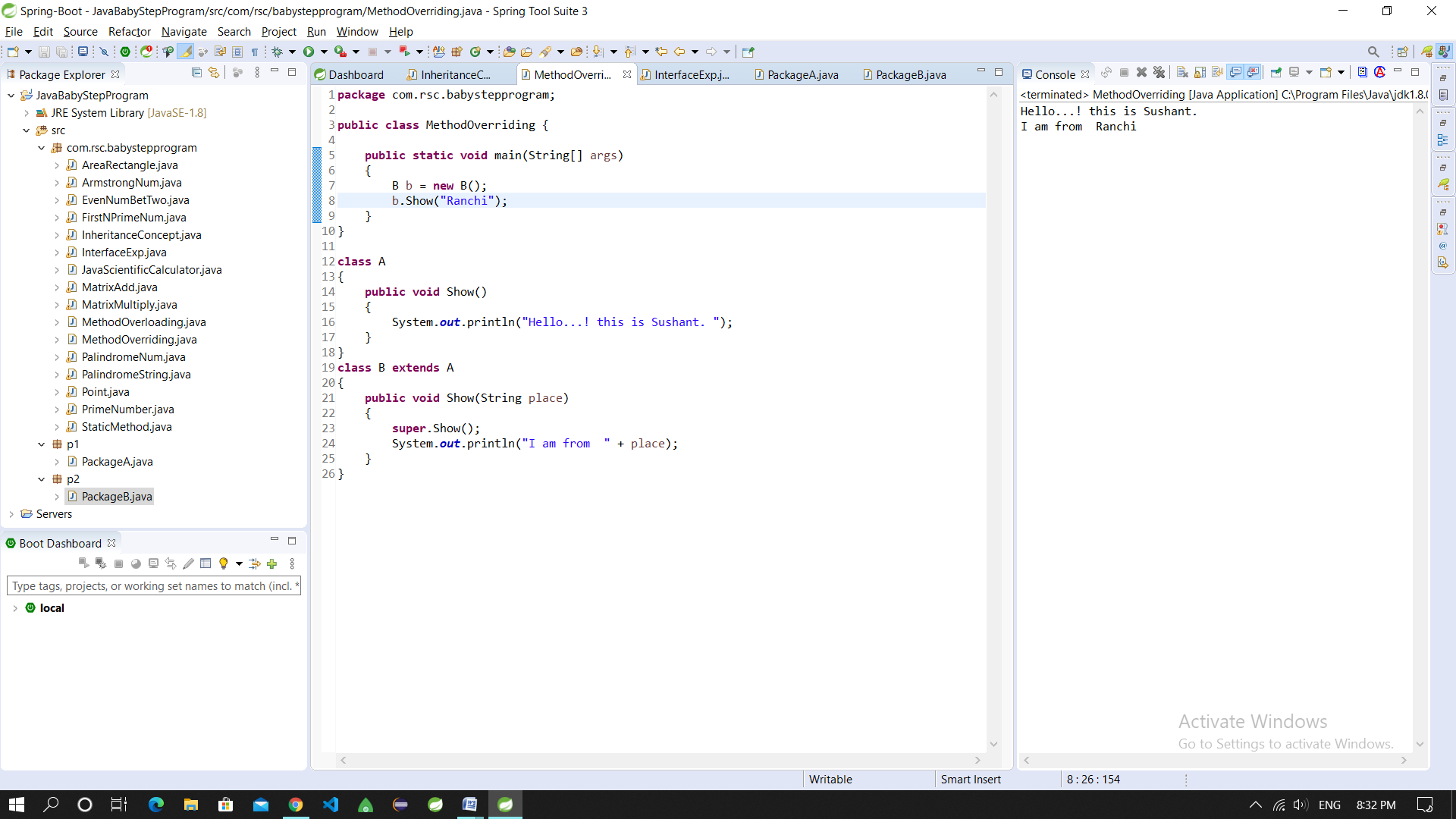
System.***out***.print("Size : "+ size);

}

}

****

1. **Write a simple java program to demonstrate method overriding.**

****

1. **Write a java program to create an interface called Shape with CalculateArea(). Create three classes namely Square,Circle,Triangle which implements Shape.**

**package** com.rsc.babystepprogram;

**import** java.util.\*;

**interface** Shape

{

**public** **void** CalculateArea();

}

**class** Square **implements** Shape

{

**int** s;

Square(**int** s){

**this**.s = s;

}

**public** **void** CalculateArea(){

System.***out***.println("The area of Square is : " + (s\*s));

}

}

**class** Circle **implements** Shape

{

**int** r;

Circle(**int** r){

**this**.r = r;

}

**public** **void** CalculateArea(){

System.***out***.println("The area of Circle is : " + (3.14\*r\*r));

}

}

**class** Triangle **implements** Shape

{

**int** b,h;

Triangle(**int** b, **int** h){

**this**.b = b;

**this**.h = h;

}

**public** **void** CalculateArea(){

System.***out***.println("The arae of Triangle is : " + (0.5\*b\*h));

}

}

**public** **class** InterfaceExp {

**public** **static** **void** main(String[] args)

{

Scanner sc = **new** Scanner(System.***in***);

Square sq = **new** Square(sc.nextInt());

Circle c = **new** Circle(sc.nextInt());

Triangle t = **new** Triangle(sc.nextInt(),sc.nextInt());

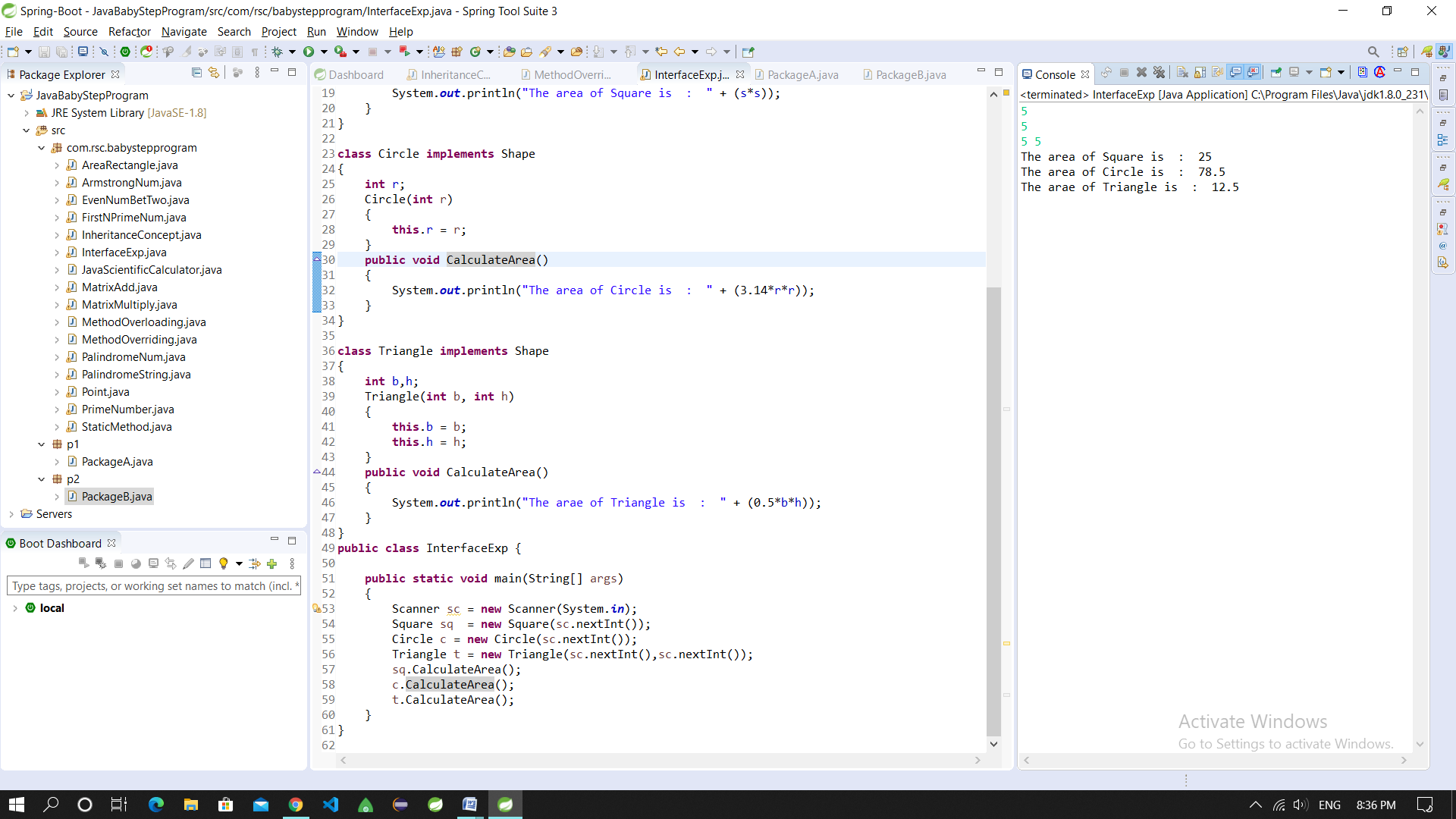
sq.CalculateArea();

c.CalculateArea();

t.CalculateArea();

}

}



1. **Create two packages p1 and p2. The package p1 contains class A which contains one display(). Create class B in package p2. The main method of class B invoke A’s display(). Write a java program to do this.**

**package** p1;

**public** **class** PackageA {

**public** **void** display()

{

System.***out***.println("Hey.. This is Package\_A from package p1");

}

}

**package** p2;

**import** p1.PackageA;

**public** **class** PackageB {

**public** **static** **void** main(String[] args)

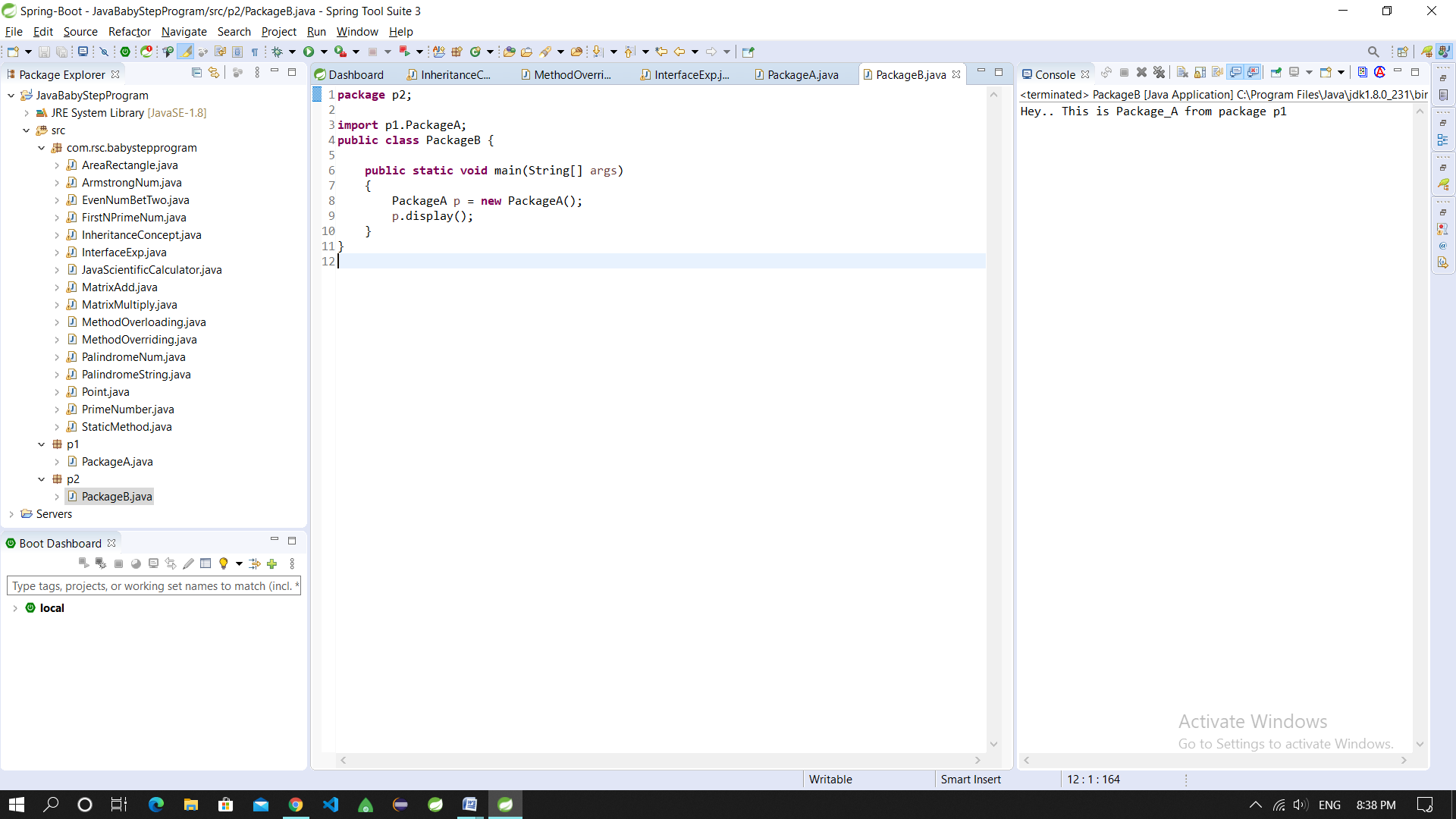
{

PackageA p = **new** PackageA();

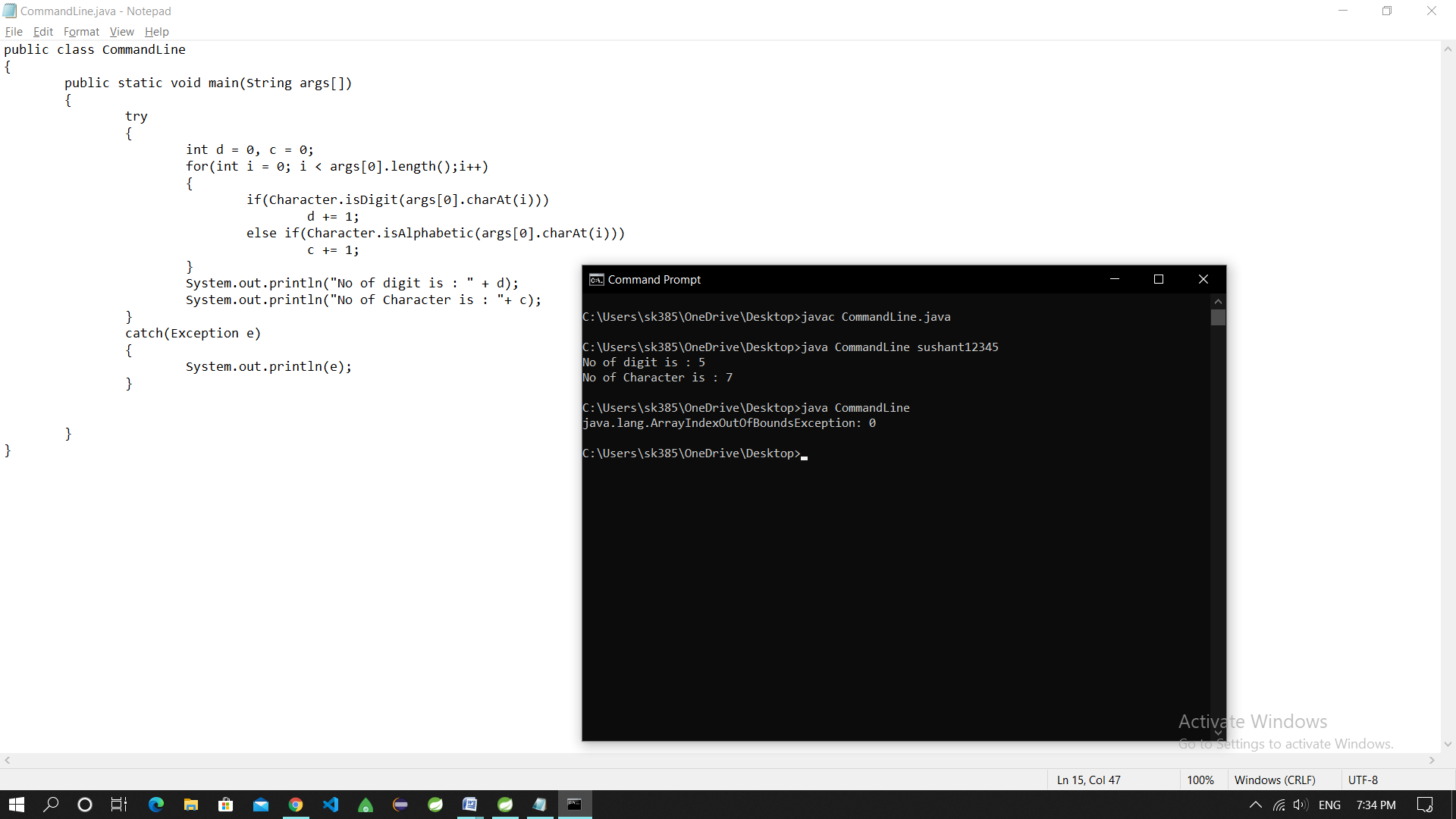
p.display();

}

}



1. **Write a java program to count numbers, characters in the command line arguments using Exception handling mechanism.**

****