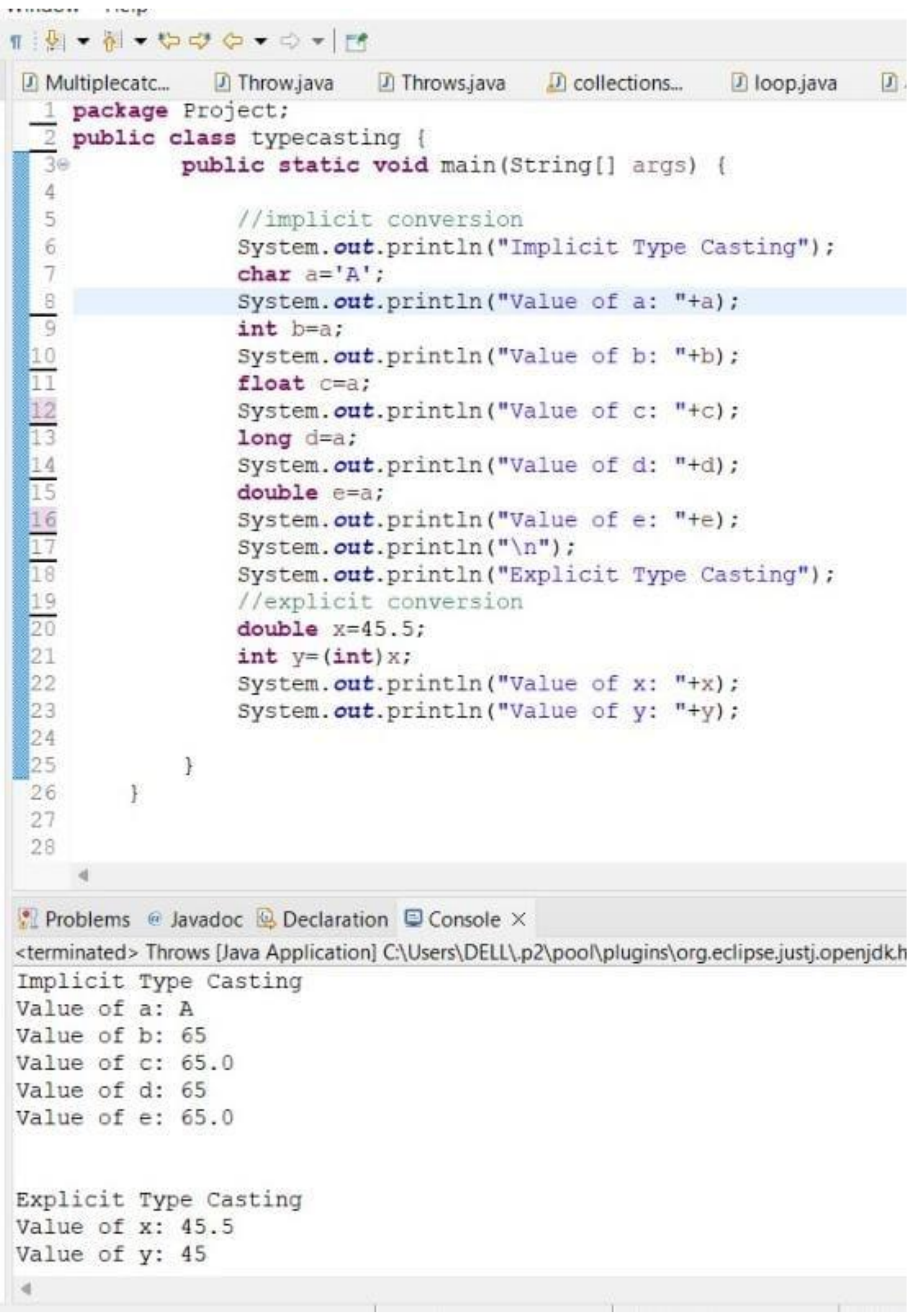
1) Write a program in Java to perform implicit and explicit type casting

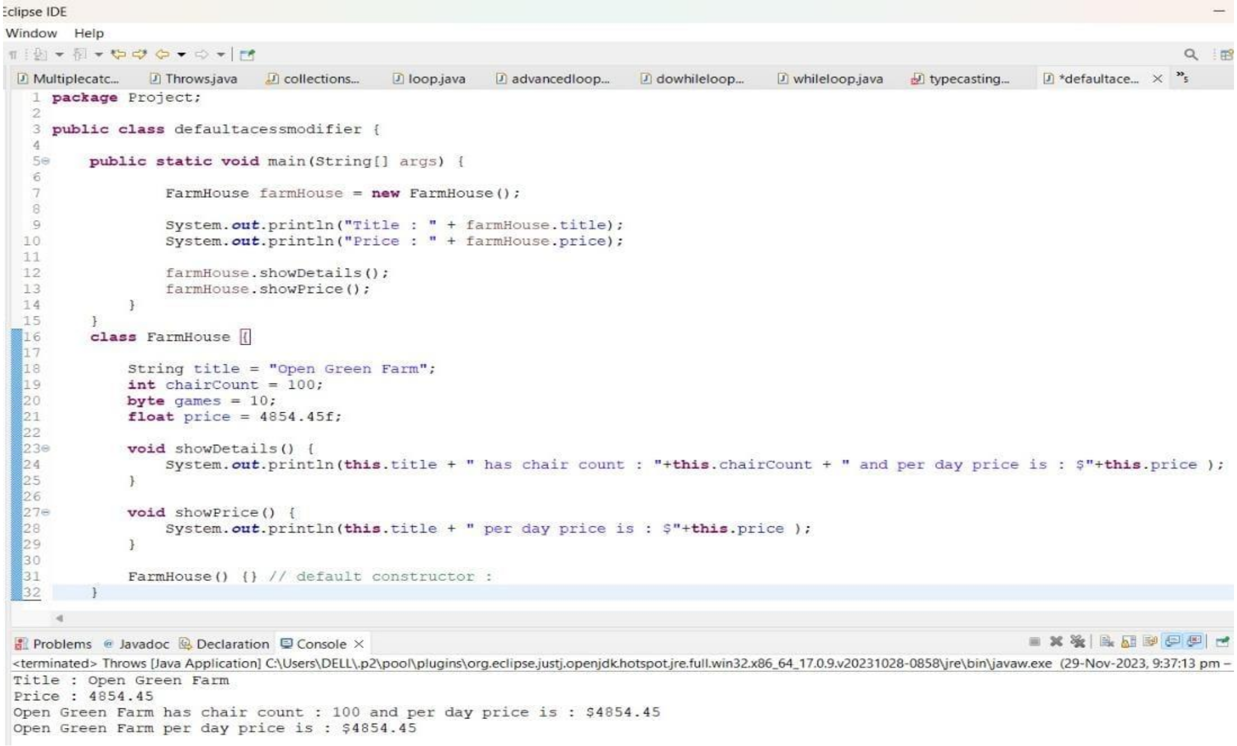


```java
package Project;
public class typecasting {
        public static void main(String[] args) {

            //implicit conversion
            System.out.println("Implicit Type Casting");
            char a='A';
            System.out.println("Value of a: "+a);
            int b=a;
            System.out.println("Value of b: "+b);
            float c=a;
            System.out.println("Value of c: "+c);
            long d=a;
            System.out.println("Value of d: "+d);
            double e=a;
            System.out.println("Value of e: "+e);
            System.out.println("\n");
            System.out.println("Explicit Type Casting");
            //explicit conversion
            double x=45.5;
            int y=(int)x;
            System.out.println("Value of x: "+x);
            System.out.println("Value of y: "+y);

        }
    }
```

```
<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.h
Implicit Type Casting
Value of a: A
Value of b: 65
Value of c: 65.0
Value of d: 65
Value of e: 65.0


Explicit Type Casting
Value of x: 45.5
Value of y: 45
```

2) Write a program in Java to verify the working of access modifiers

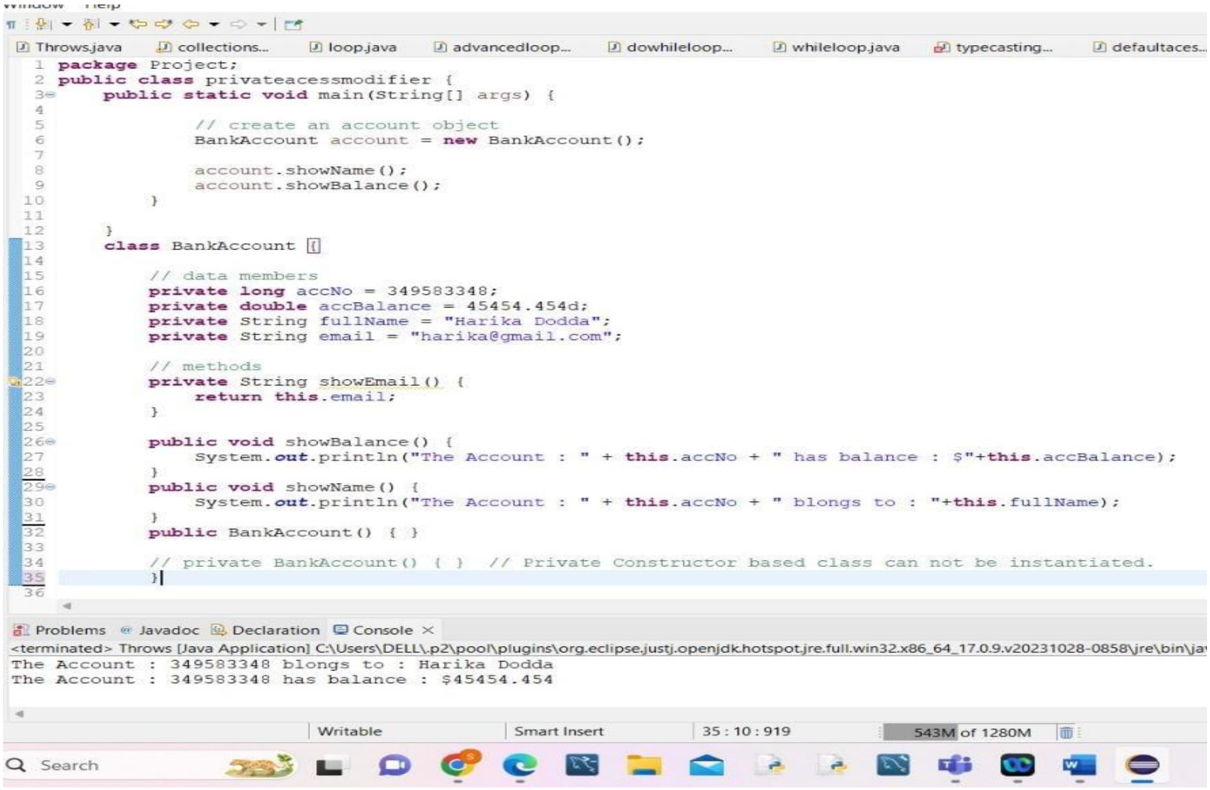
(a) Default Access Modifiers



```java
package Project;

public class defaultacessmodifier {

    public static void main(String[] args) {

            FarmHouse farmHouse = new FarmHouse();

            System.out.println("Title : " + farmHouse.title);
            System.out.println("Price : " + farmHouse.price);

            farmHouse.showDetails();
            farmHouse.showPrice();
        }
    }
    class FarmHouse {

        String title = "Open Green Farm";
        int chairCount = 100;
        byte games = 10;
        float price = 4854.45f;

        void showDetails() {
            System.out.println(this.title + " has chair count : "+this.chairCount + " and per day price is : $"+this.price );
        }

        void showPrice() {
            System.out.println(this.title + " per day price is : $"+this.price );
        }

        FarmHouse() {} // default constructor :
    }
```

```
<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe  (29-Nov-2023, 9:37:13 pm –
Title : Open Green Farm
Price : 4854.45
Open Green Farm has chair count : 100 and per day price is : $4854.45
Open Green Farm per day price is : $4854.45
```

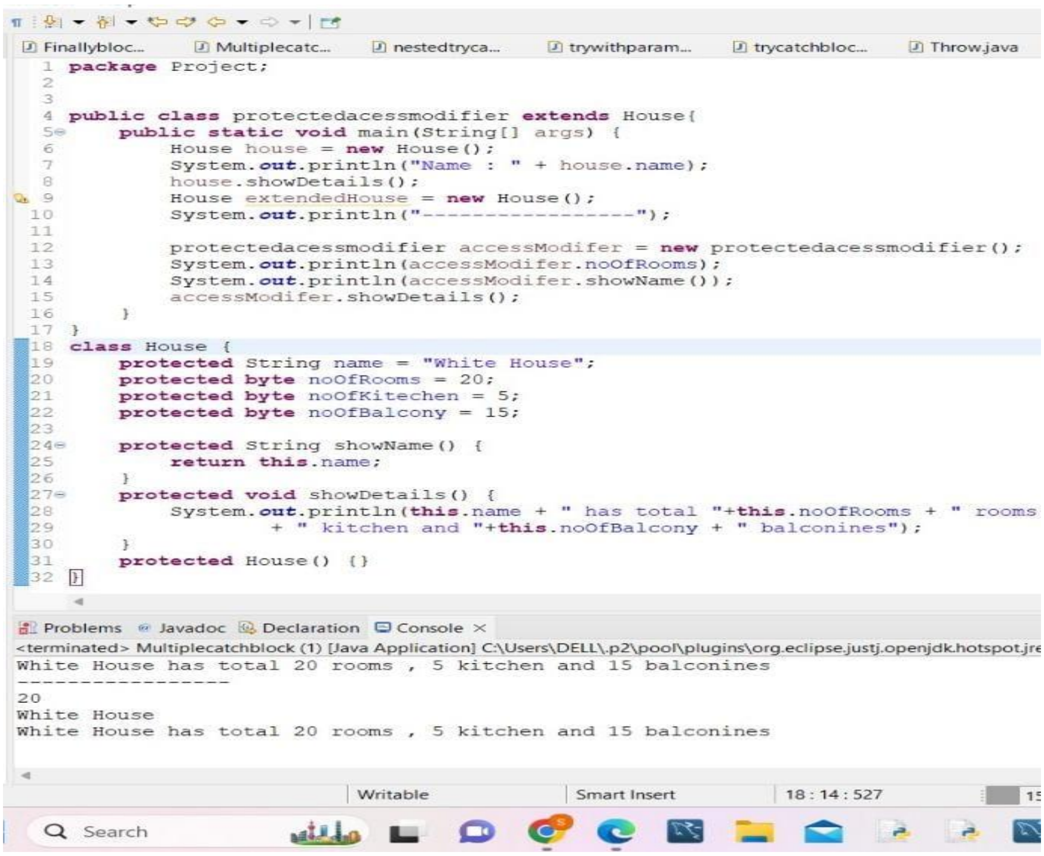
(b) Private Access Modifiers



```java
package Project;
public class privateacessmodifier {
    public static void main(String[] args) {

            // create an account object
            BankAccount account = new BankAccount();

            account.showName();
            account.showBalance();
        }

    }
    class BankAccount {

        // data members
        private long accNo = 349583348;
        private double accBalance = 45454.454d;
        private String fullName = "Harika Dodda";
        private String email = "harika@gmail.com";

        // methods
        private String showEmail() {
            return this.email;
        }

        public void showBalance() {
            System.out.println("The Account : " + this.accNo + " has balance : $"+this.accBalance);
        }
        public void showName() {
            System.out.println("The Account : " + this.accNo + " blongs to : "+this.fullName);
        }
        public BankAccount() { }

        // private BankAccount() { }  // Private Constructor based class can not be instantiated.
        }
```

```
<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\ja
The Account : 349583348 blongs to : Harika Dodda
The Account : 349583348 has balance : $45454.454
```

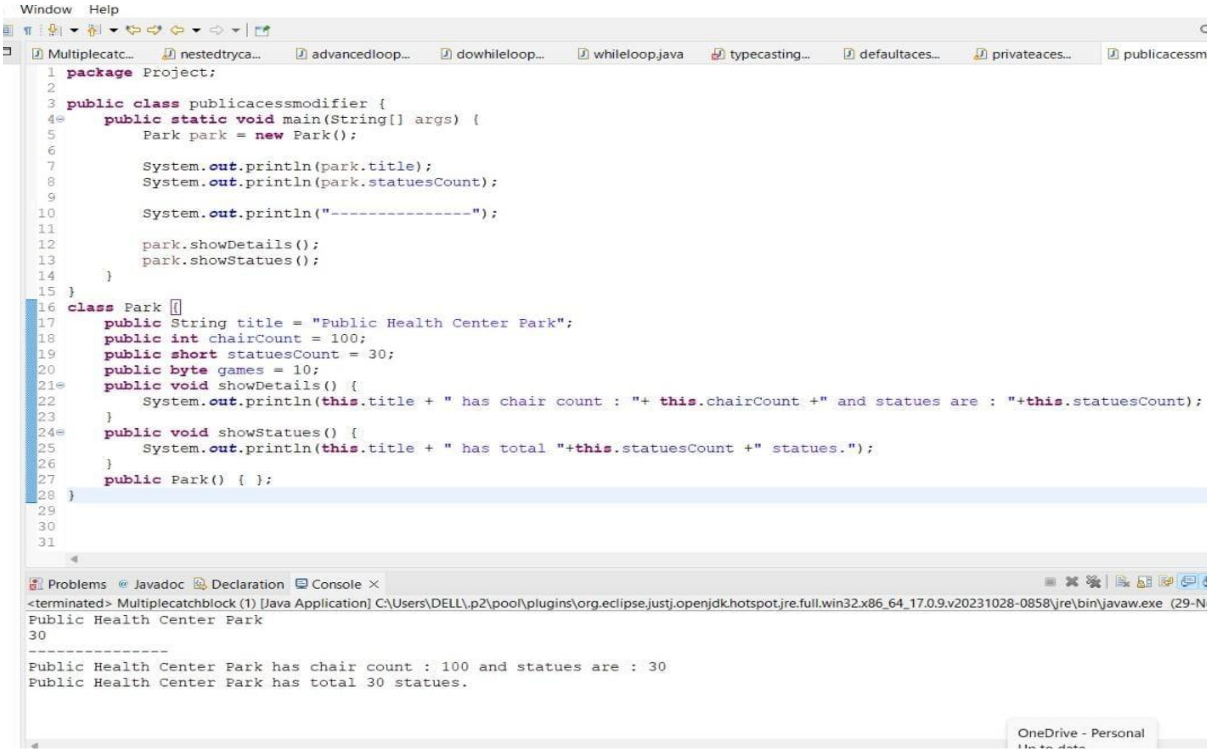
(c) Protected Access Modifiers



```java
package Project;


public class protectedacessmodifier extends House{
    public static void main(String[] args) {
        House house = new House();
        System.out.println("Name : " + house.name);
        house.showDetails();
        House extendedHouse = new House();
        System.out.println("------------------");

        protectedacessmodifier accessModifer = new protectedacessmodifier();
        System.out.println(accessModifer.noOfRooms);
        System.out.println(accessModifer.showName());
        accessModifer.showDetails();
    }
}
class House {
    protected String name = "White House";
    protected byte noOfRooms = 20;
    protected byte noOfKitechen = 5;
    protected byte noOfBalcony = 15;

    protected String showName() {
        return this.name;
    }
    protected void showDetails() {
        System.out.println(this.name + " has total "+this.noOfRooms + " rooms
                + " kitchen and "+this.noOfBalcony + " balconines");
    }
    protected House() {}
}
```

```
<terminated> Multiplecatchblock (1) [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre
White House has total 20 rooms , 5 kitchen and 15 balconines
-----------------
20
White House
White House has total 20 rooms , 5 kitchen and 15 balconines
```

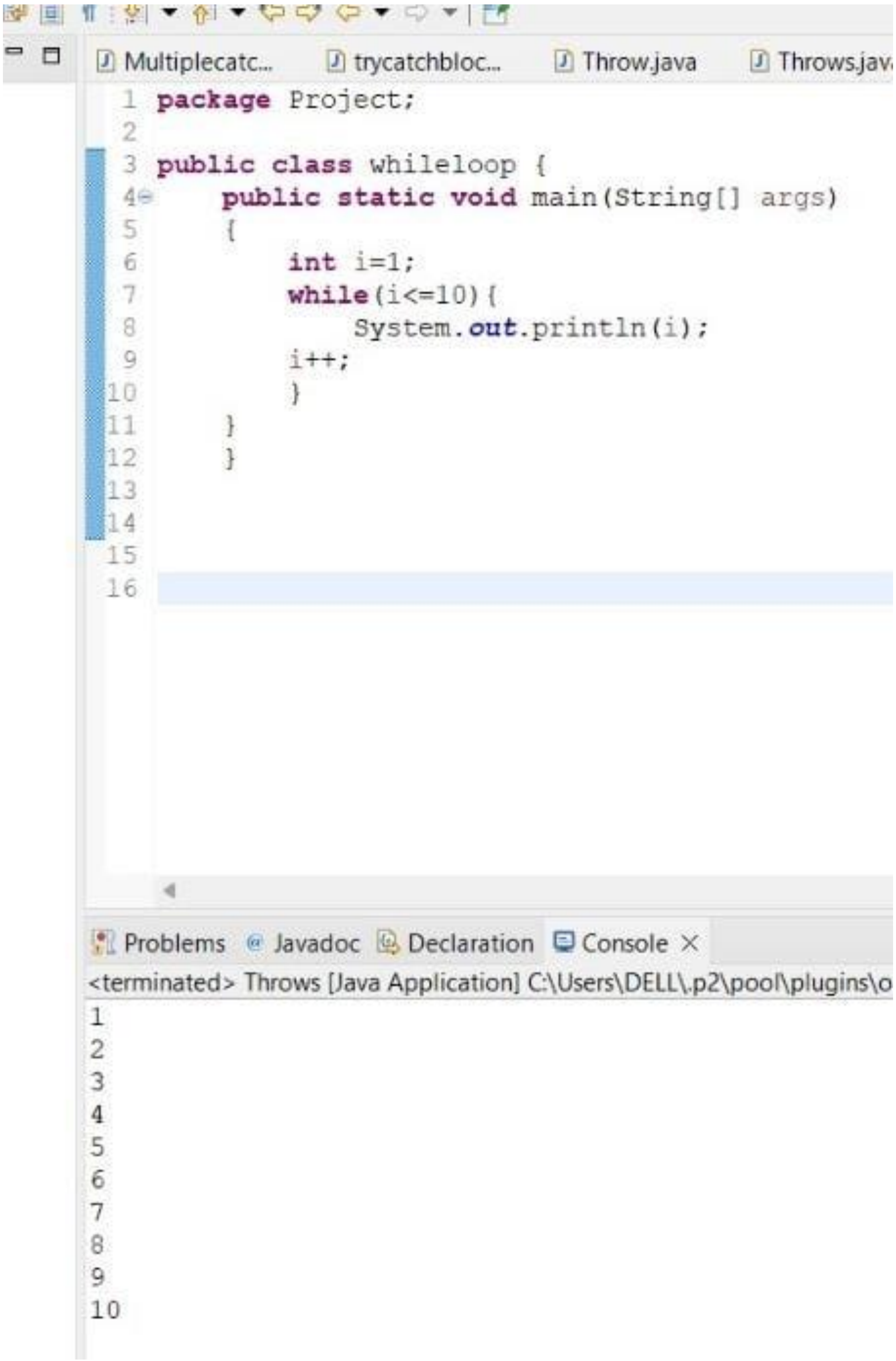
(d) Public Access Modifiers



```java
package Project;

public class publicacessmodifier {
    public static void main(String[] args) {
        Park park = new Park();

        System.out.println(park.title);
        System.out.println(park.statuesCount);

        System.out.println("---------------");

        park.showDetails();
        park.showStatues();
    }
}
class Park {
    public String title = "Public Health Center Park";
    public int chairCount = 100;
    public short statuesCount = 30;
    public byte games = 10;
    public void showDetails() {
        System.out.println(this.title + " has chair count : "+ this.chairCount +" and statues are : "+this.statuesCount);
    }
    public void showStatues() {
        System.out.println(this.title + " has total "+this.statuesCount +" statues.");
    }
    public Park() { };
}
```

```
<terminated> Multiplecatchblock (1) [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.9.v20231028-0858\jre\bin\javaw.exe (29-N
Public Health Center Park
30
---------------
Public Health Center Park has chair count : 100 and statues are : 30
Public Health Center Park has total 30 statues.
```

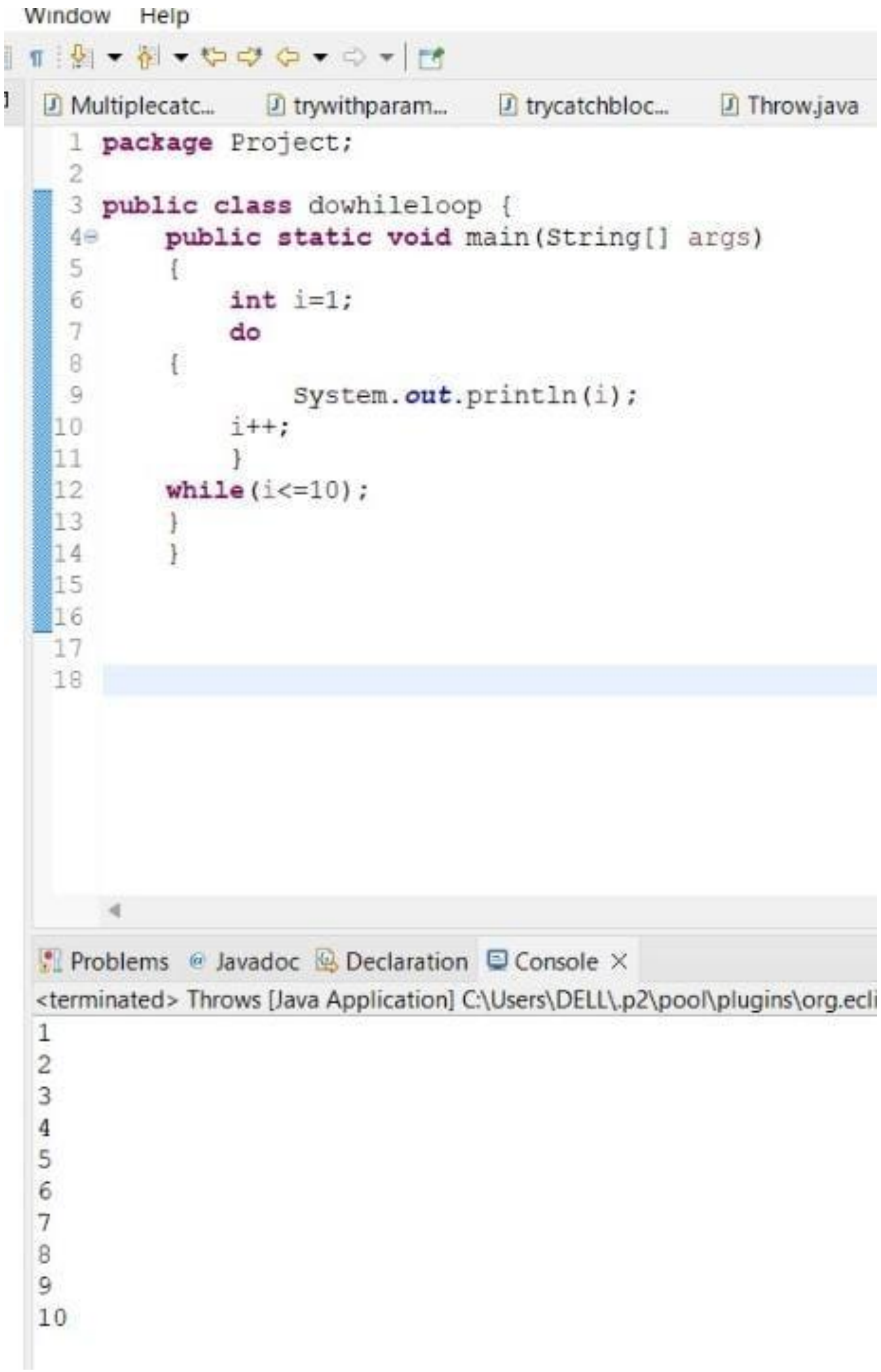
3) Write a program to demonstrate the while loop



```java
package Project;

public class whileloop {
    public static void main(String[] args)
    {
        int i=1;
        while(i<=10){
            System.out.println(i);
        i++;
        }
    }
    }
```

```
<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\o
1
2
3
4
5
6
7
8
9
10
```

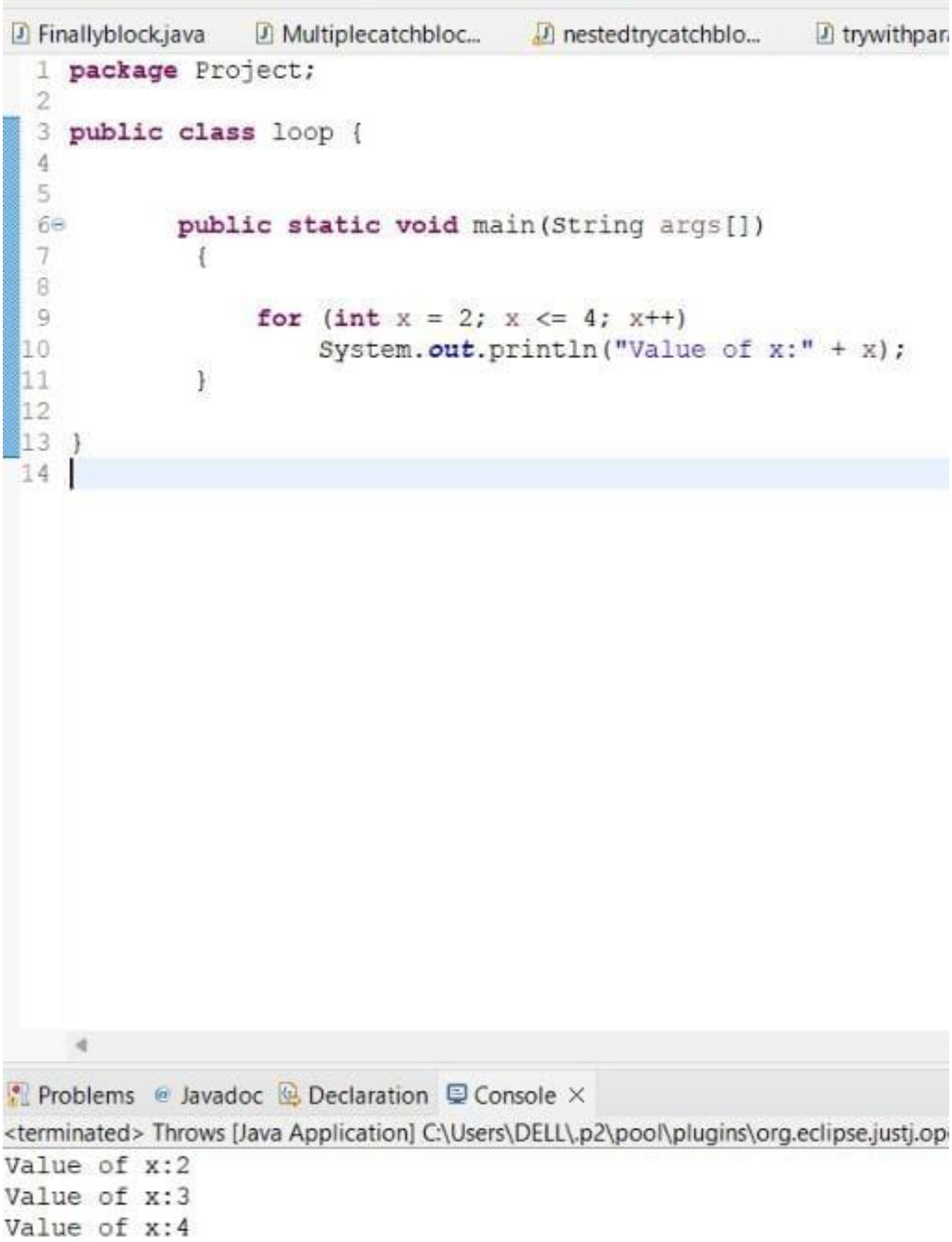
4) Write a program to demonstrate the do while loop



```java
package Project;

public class dowhileloop {
    public static void main(String[] args)
    {
        int i=1;
        do
    {
            System.out.println(i);
        i++;
        }
    while(i<=10);
    }
    }
```

```
<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org.ecli
1
2
3
4
5
6
7
8
9
10
```

5) Write a program to demonstrate the for loop

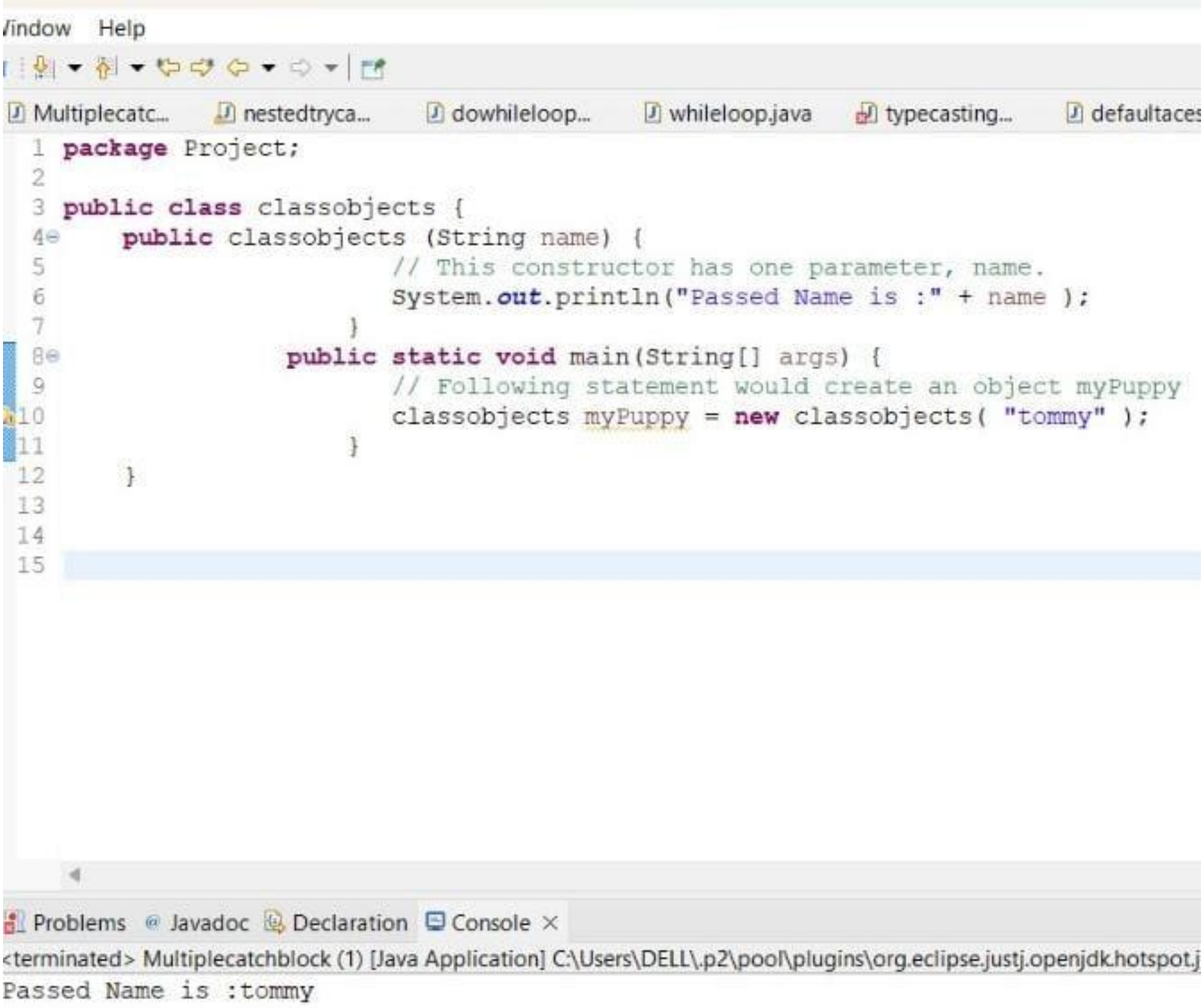


```java
package Project;

public class loop {


	public static void main(String args[])
	 {

	     for (int x = 2; x <= 4; x++)
	         System.out.println("Value of x:" + x);
	 }

}

```

<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.op

```
Value of x:2
Value of x:3
Value of x:4
```

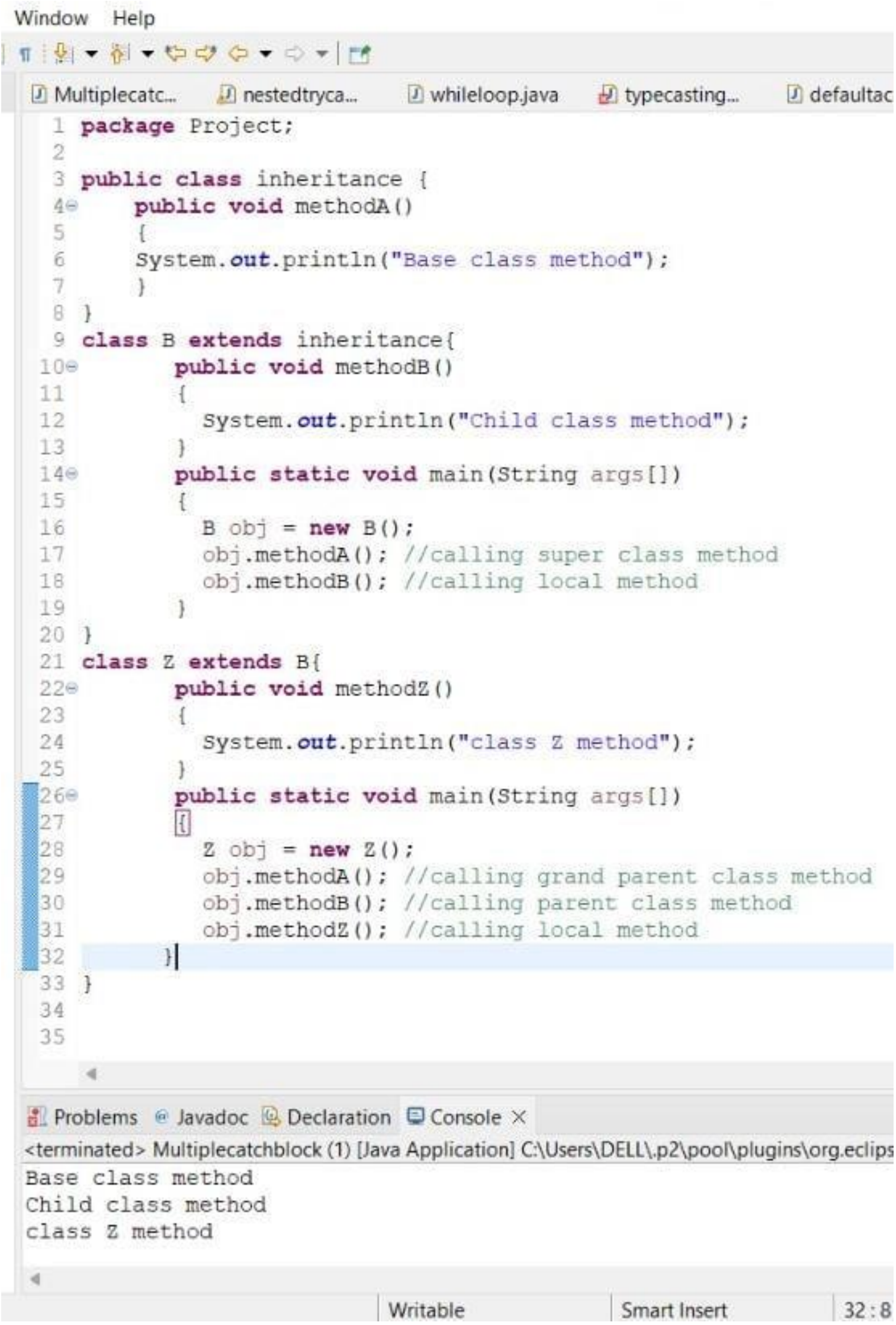
6) Demonstrate the Classes, Objects, and Constructors



```
package Project;

public class classobjects {
    public classobjects (String name) {
                    // This constructor has one parameter, name.
                    System.out.println("Passed Name is :" + name );
                 }
            public static void main(String[] args) {
                    // Following statement would create an object myPuppy
                    classobjects myPuppy = new classobjects( "tommy" );
                 }
    }
```

```
<terminated> Multiplecatchblock (1) [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.j
Passed Name is :tommy
```

7) Demonstrate types of inheritance

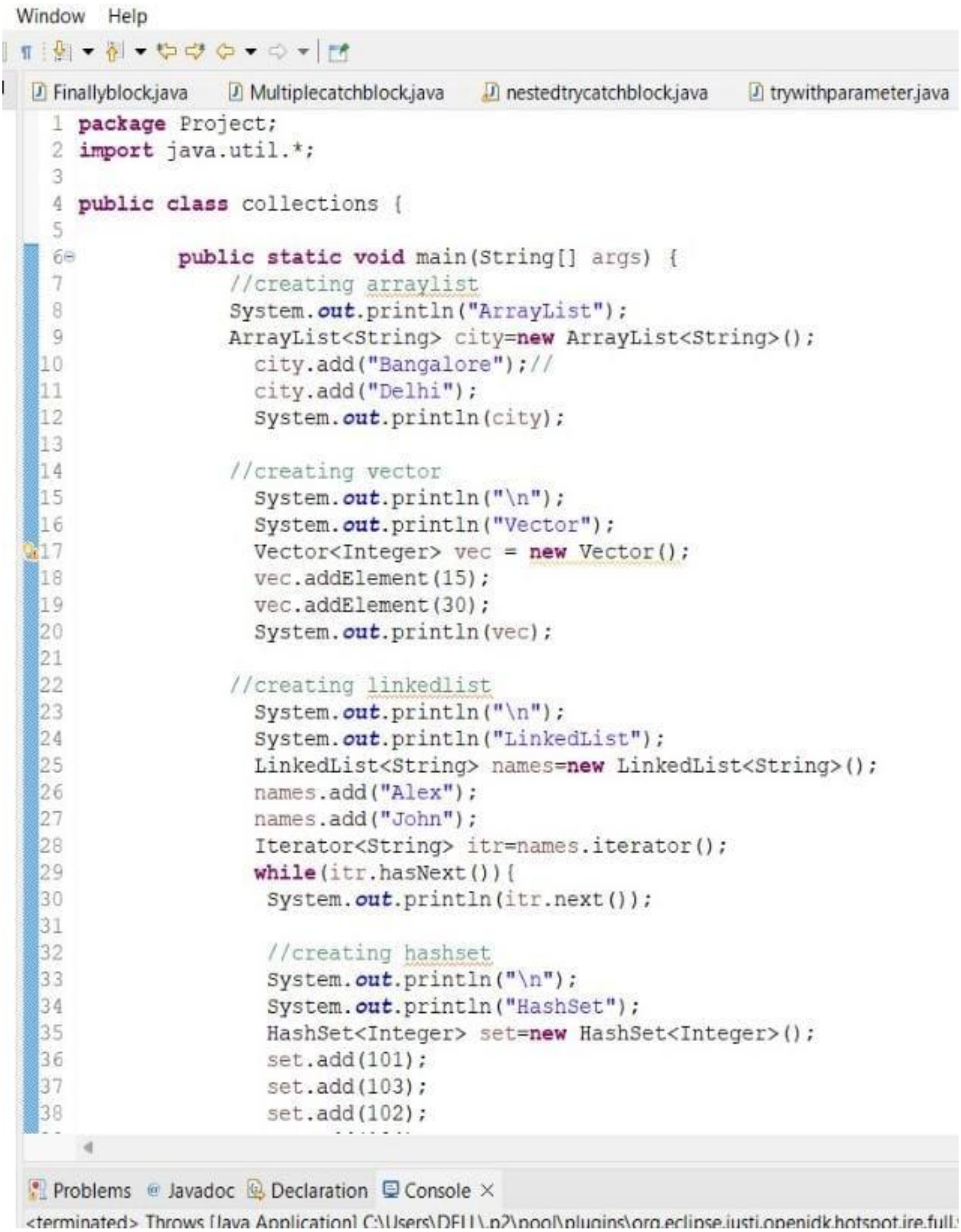


```java
package Project;

public class inheritance {
	public void methodA()
	{
	System.out.println("Base class method");
	}
}
class B extends inheritance{
		public void methodB()
		{
		  System.out.println("Child class method");
		}
		public static void main(String args[])
		{
		  B obj = new B();
		  obj.methodA(); //calling super class method
		  obj.methodB(); //calling local method
		}
}
class Z extends B{
		public void methodZ()
		{
		  System.out.println("class Z method");
		}
		public static void main(String args[])
		{
		  Z obj = new Z();
		  obj.methodA(); //calling grand parent class method
		  obj.methodB(); //calling parent class method
		  obj.methodZ(); //calling local method
		}
}
```

<terminated> Multiplecatchblock (1) [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclips

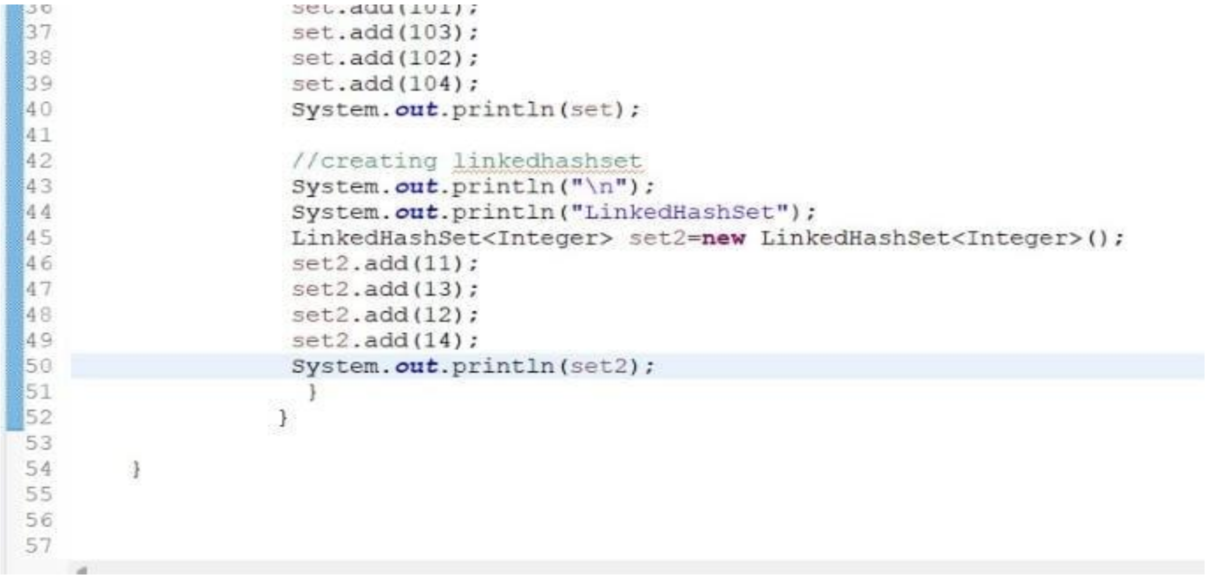
```
Base class method
Child class method
class Z method
```

7) Writing a program in Java to verify implementations of collections

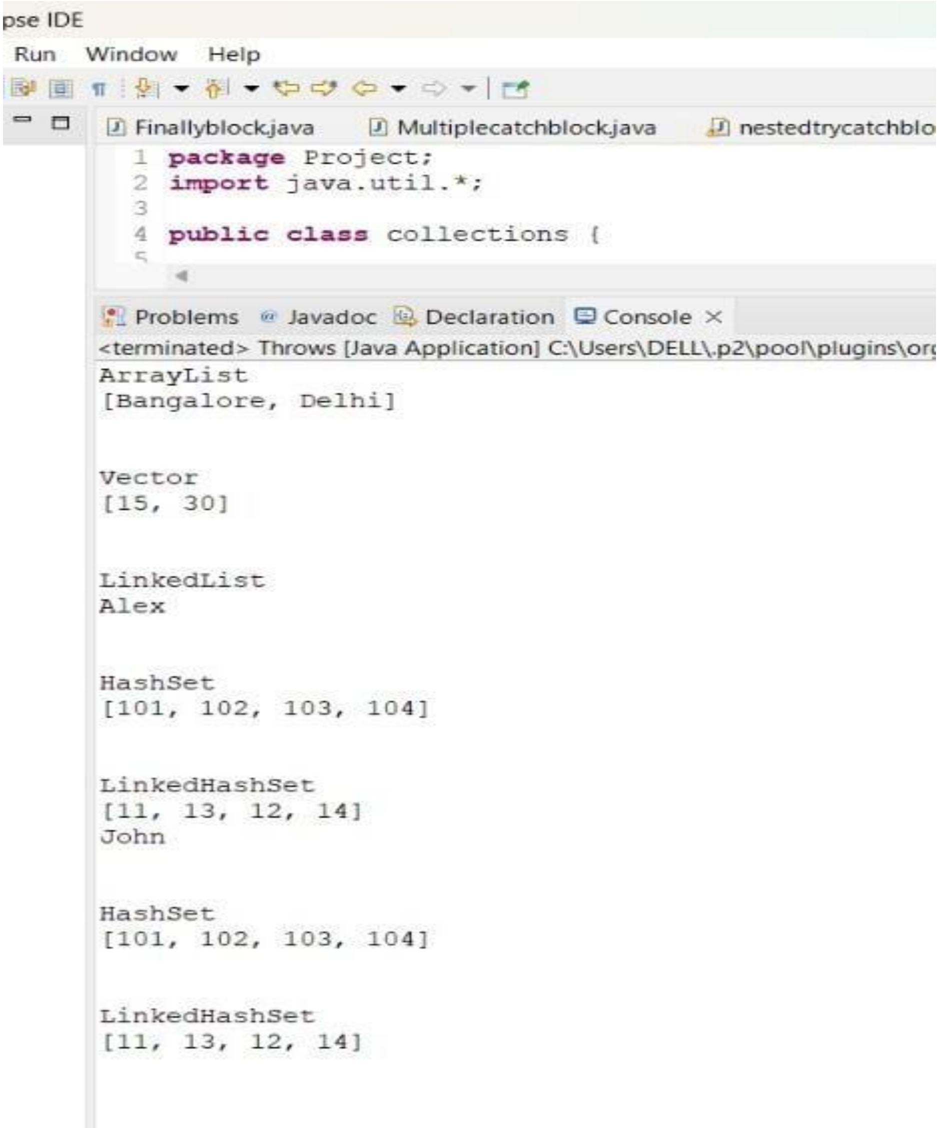


```
package Project;
import java.util.*;

public class collections {

        public static void main(String[] args) {
            //creating arraylist
            System.out.println("ArrayList");
            ArrayList<String> city=new ArrayList<String>();
              city.add("Bangalore");//
              city.add("Delhi");
              System.out.println(city);

            //creating vector
              System.out.println("\n");
              System.out.println("Vector");
              Vector<Integer> vec = new Vector();
              vec.addElement(15);
              vec.addElement(30);
              System.out.println(vec);

            //creating linkedlist
              System.out.println("\n");
              System.out.println("LinkedList");
              LinkedList<String> names=new LinkedList<String>();
              names.add("Alex");
              names.add("John");
              Iterator<String> itr=names.iterator();
              while(itr.hasNext()){
               System.out.println(itr.next());

               //creating hashset
               System.out.println("\n");
               System.out.println("HashSet");
               HashSet<Integer> set=new HashSet<Integer>();
               set.add(101);
               set.add(103);
               set.add(102);
```

```
<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.
```



```
                set.add(101);
                set.add(103);
                set.add(102);
                set.add(104);
                System.out.println(set);

                //creating linkedhashset
                System.out.println("\n");
                System.out.println("LinkedHashSet");
                LinkedHashSet<Integer> set2=new LinkedHashSet<Integer>();
                set2.add(11);
                set2.add(13);
                set2.add(12);
                set2.add(14);
                System.out.println(set2);
                 }
               }

     }
```

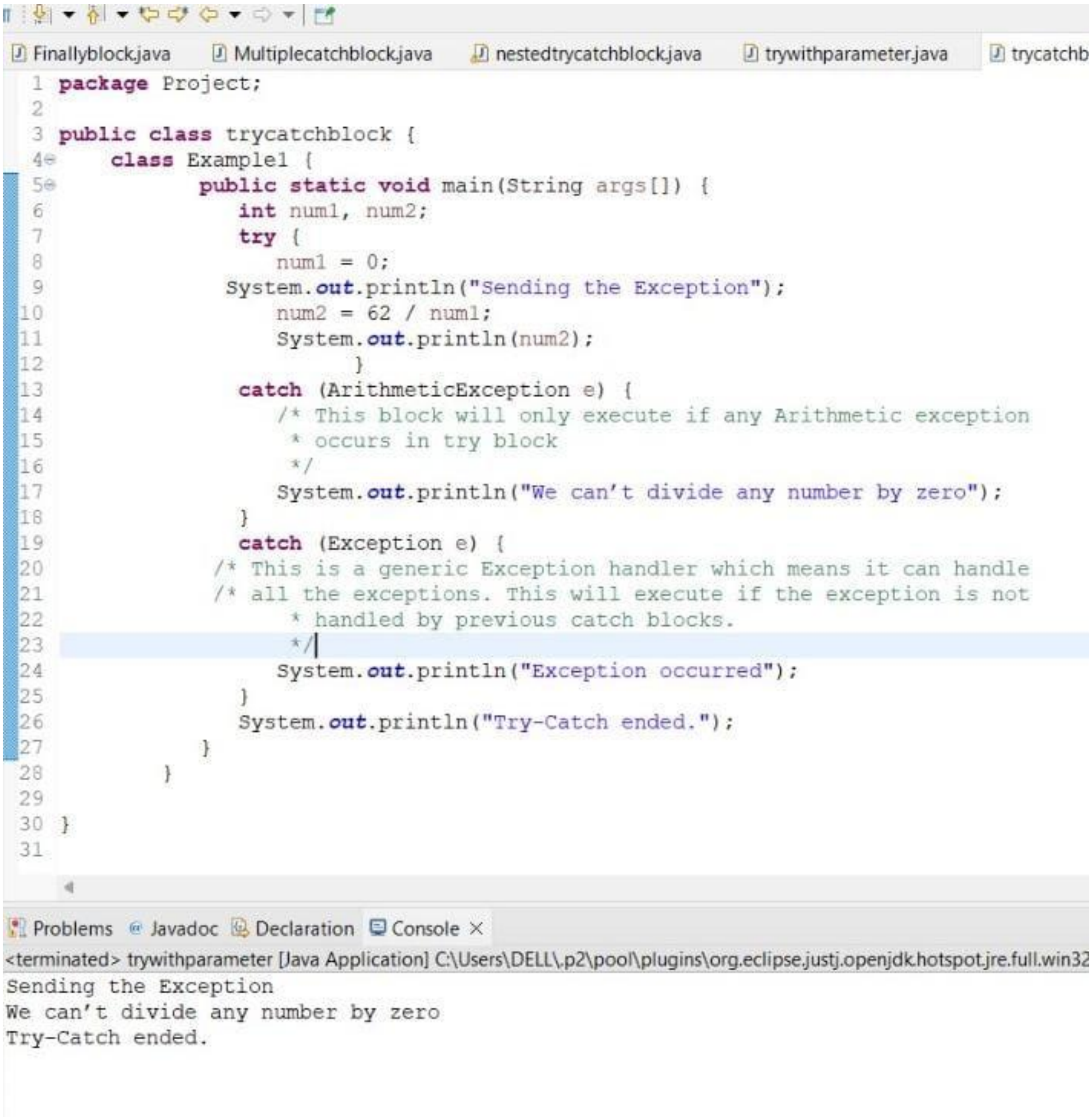


```
package Project;
import java.util.*;

public class collections {
```

<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org

```
ArrayList
[Bangalore, Delhi]


Vector
[15, 30]


LinkedList
Alex


HashSet
[101, 102, 103, 104]


LinkedHashSet
[11, 13, 12, 14]
John


HashSet
[101, 102, 103, 104]


LinkedHashSet
[11, 13, 12, 14]
```

8) Writing a program to perform try-catch block



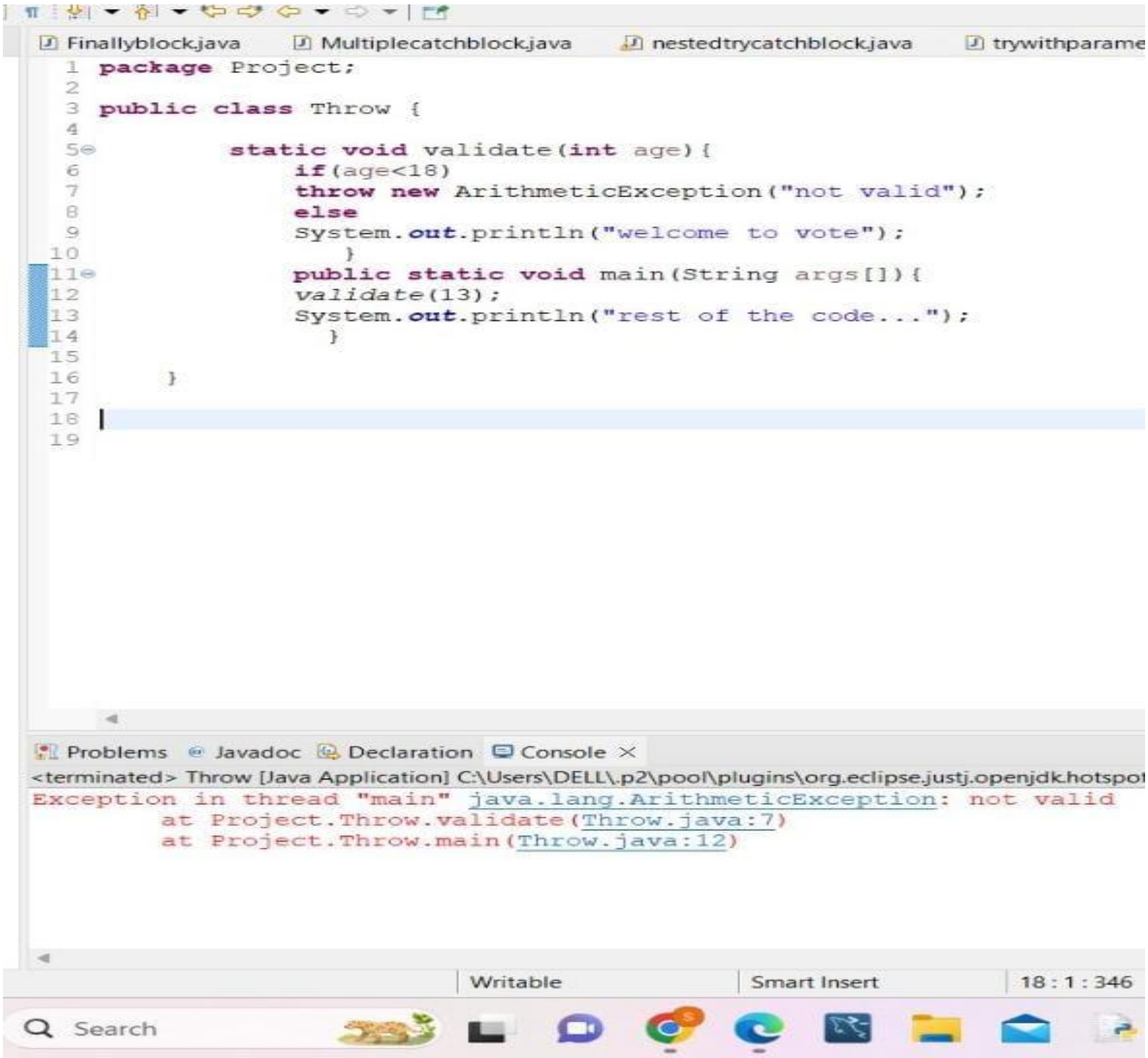
```java
package Project;

public class trycatchblock {
    class Example1 {
           public static void main(String args[]) {
              int num1, num2;
              try {
                 num1 = 0;
             System.out.println("Sending the Exception");
                 num2 = 62 / num1;
                 System.out.println(num2);
                       }
              catch (ArithmeticException e) {
                 /* This block will only execute if any Arithmetic exception
                  * occurs in try block
                  */
                 System.out.println("We can’t divide any number by zero");
              }
              catch (Exception e) {
            /* This is a generic Exception handler which means it can handle
            /* all the exceptions. This will execute if the exception is not
                  * handled by previous catch blocks.
                  */
                 System.out.println("Exception occurred");
              }
              System.out.println("Try-Catch ended.");
           }
        }

}
```

<terminated> trywithparameter [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32

```
Sending the Exception
We can’t divide any number by zero
Try-Catch ended.
```

9) Writing code for throw and throws keyword

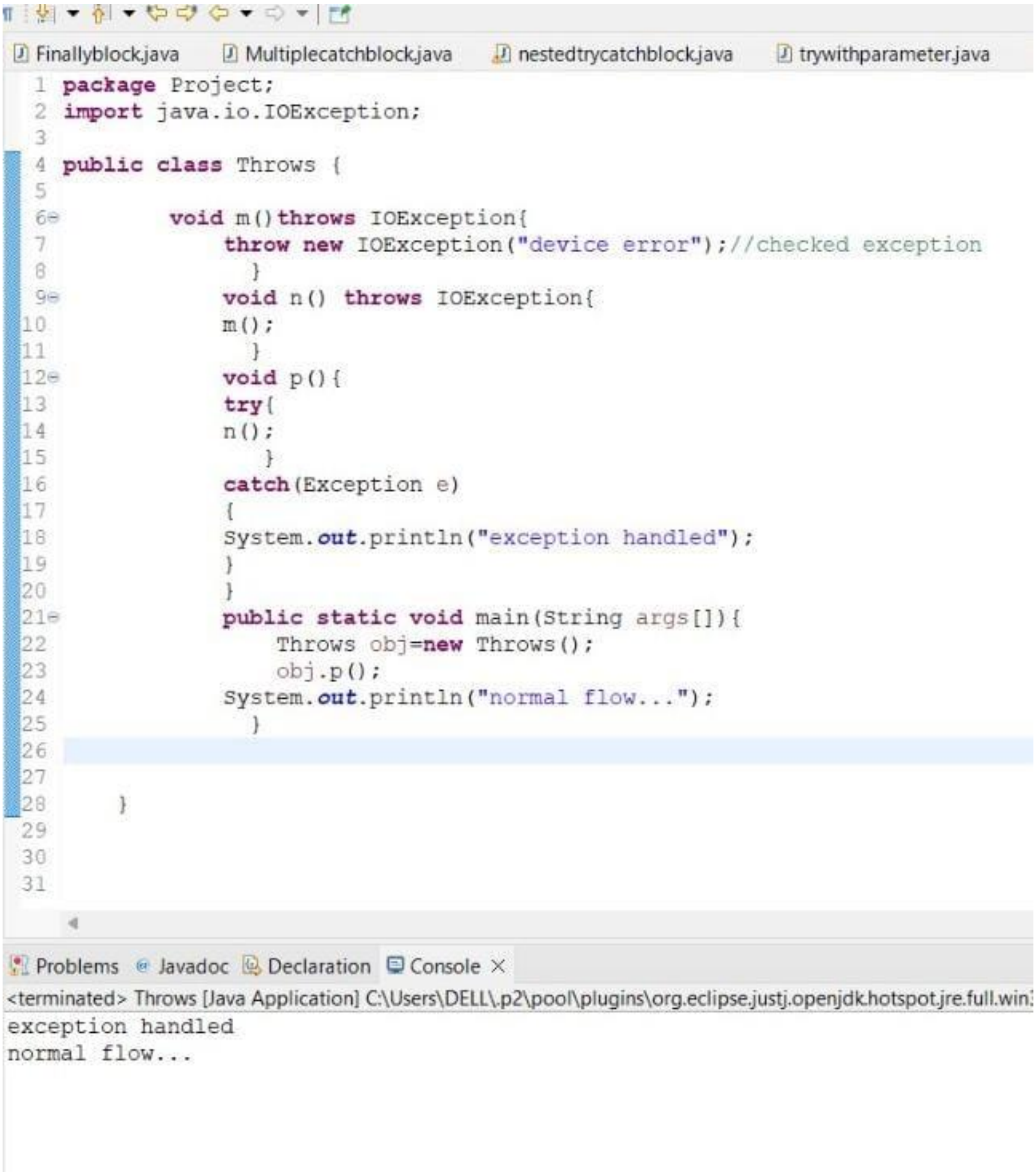
(a) Throw



```
package Project;

public class Throw {

        static void validate(int age){
            if(age<18)
            throw new ArithmeticException("not valid");
            else
            System.out.println("welcome to vote");
               }
            public static void main(String args[]){
            validate(13);
            System.out.println("rest of the code...");
             }

    }


```

```
<terminated> Throw [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot
Exception in thread "main" java.lang.ArithmeticException: not valid
        at Project.Throw.validate(Throw.java:7)
        at Project.Throw.main(Throw.java:12)
```

(b) Throws

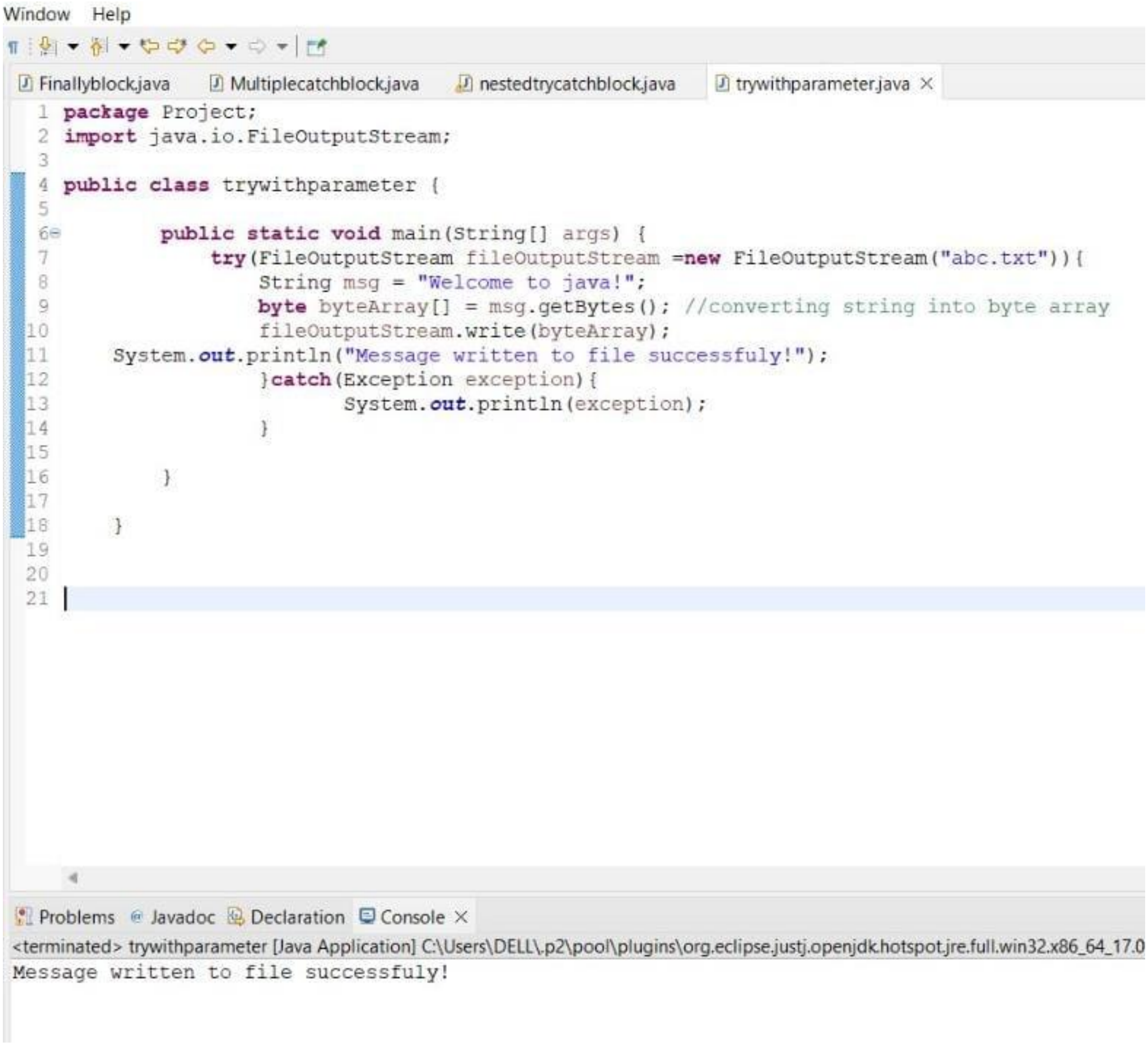


```
package Project;
import java.io.IOException;

public class Throws {

        void m()throws IOException{
            throw new IOException("device error");//checked exception
              }
            void n() throws IOException{
            m();
              }
            void p(){
            try{
            n();
               }
            catch(Exception e)
            {
            System.out.println("exception handled");
            }
            }
            public static void main(String args[]){
                Throws obj=new Throws();
                obj.p();
            System.out.println("normal flow...");
              }


    }
```

<terminated> Throws [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win3

```
exception handled
normal flow...
```

10) Writing code for a try block with parameters

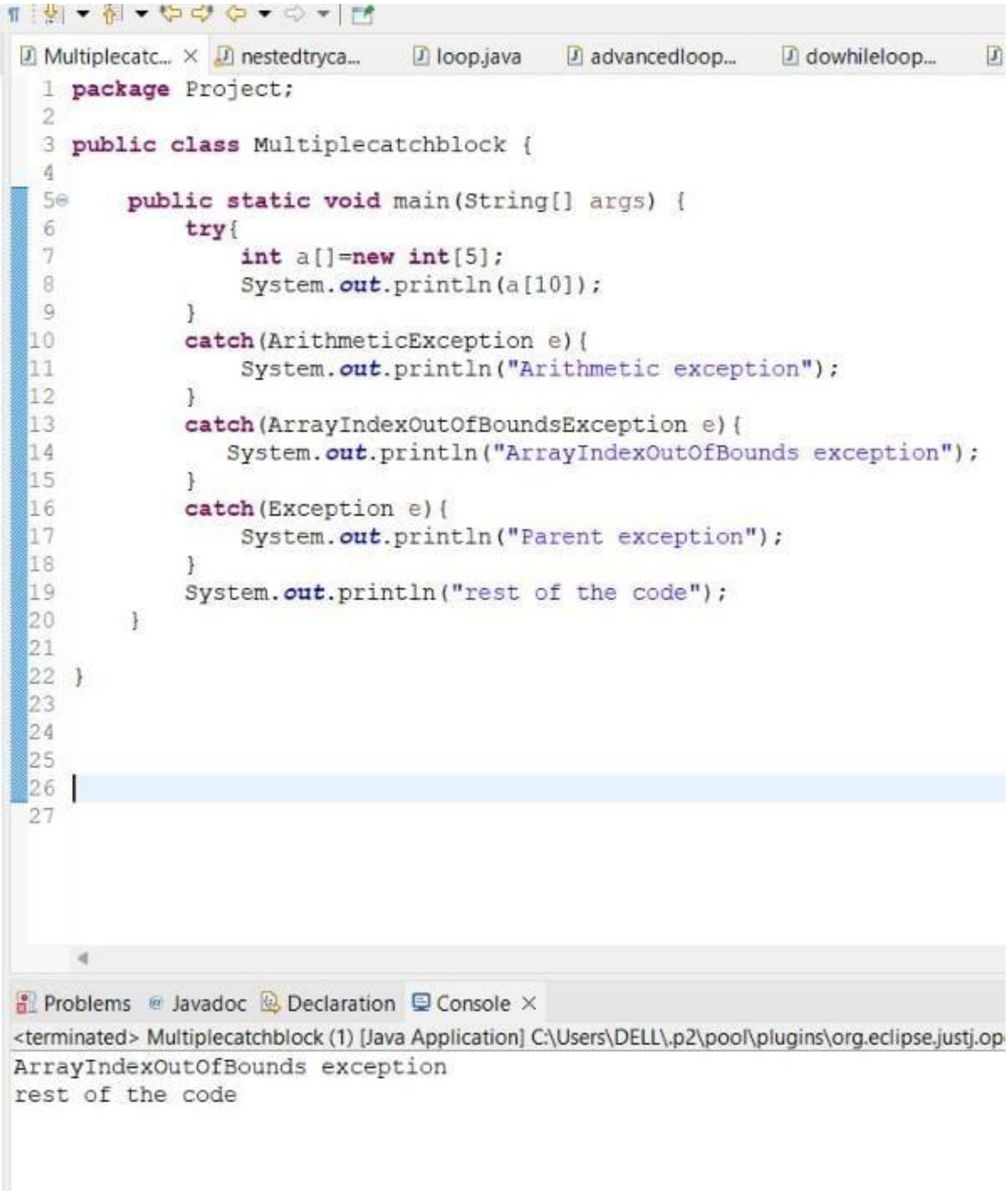


```
package Project;
import java.io.FileOutputStream;

public class trywithparameter {

        public static void main(String[] args) {
            try(FileOutputStream fileOutputStream =new FileOutputStream("abc.txt")){
                String msg = "Welcome to java!";
                byte byteArray[] = msg.getBytes(); //converting string into byte array
                fileOutputStream.write(byteArray);
    System.out.println("Message written to file successfuly!");
                }catch(Exception exception){
                       System.out.println(exception);
                }


        }

    }


```

```
<terminated> trywithparameter [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0
Message written to file successfuly!
```

11) Writing code for multiple catch blocks

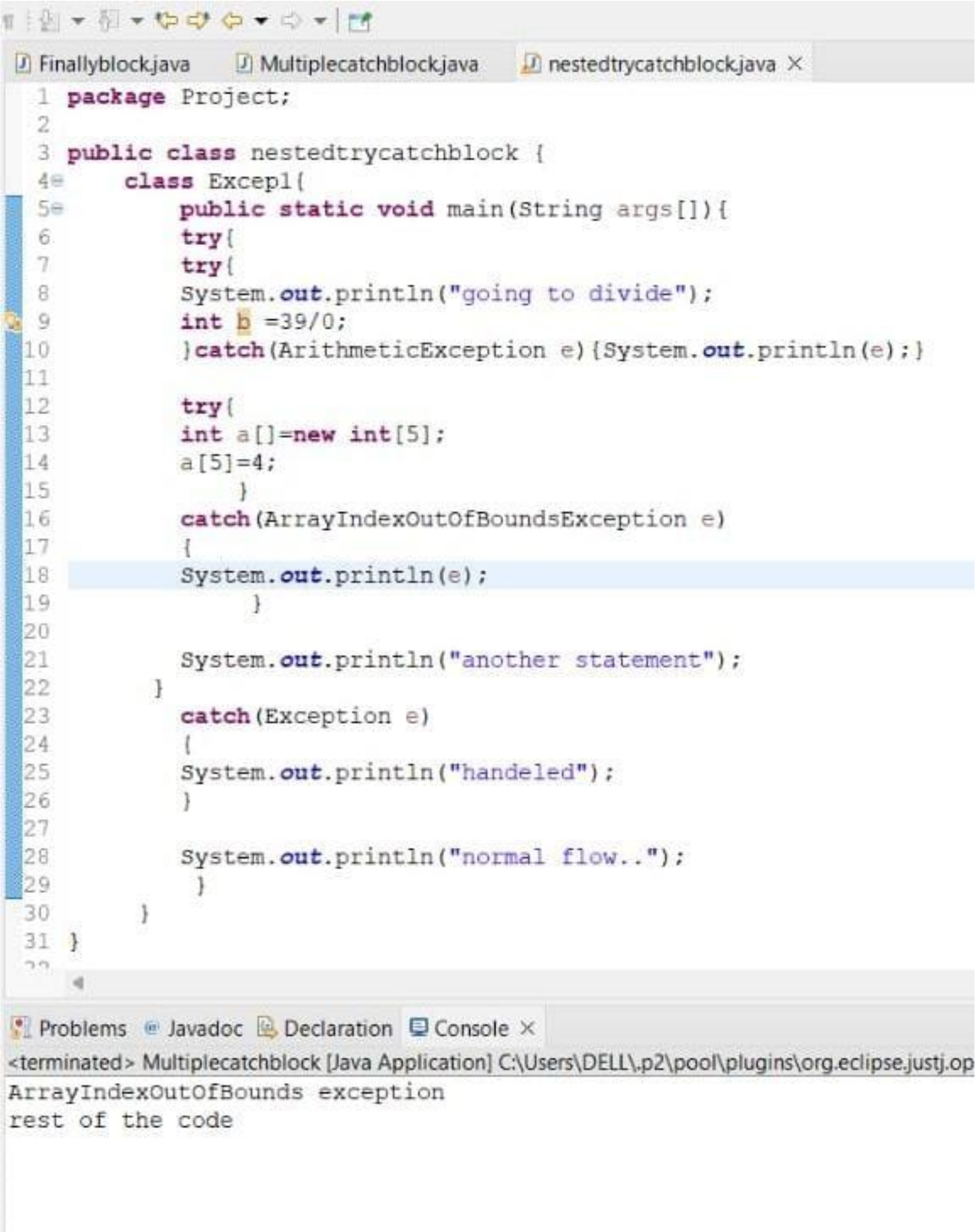
(a) Multi Catch Blocks



```java
package Project;

public class Multiplecatchblock {

    public static void main(String[] args) {
        try{
            int a[]=new int[5];
            System.out.println(a[10]);
        }
        catch(ArithmeticException e){
            System.out.println("Arithmetic exception");
        }
        catch(ArrayIndexOutOfBoundsException e){
           System.out.println("ArrayIndexOutOfBounds exception");
        }
        catch(Exception e){
            System.out.println("Parent exception");
        }
        System.out.println("rest of the code");
    }

}
```

```
<terminated> Multiplecatchblock (1) [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.op
ArrayIndexOutOfBounds exception
rest of the code
```

(b) Nested try catch block

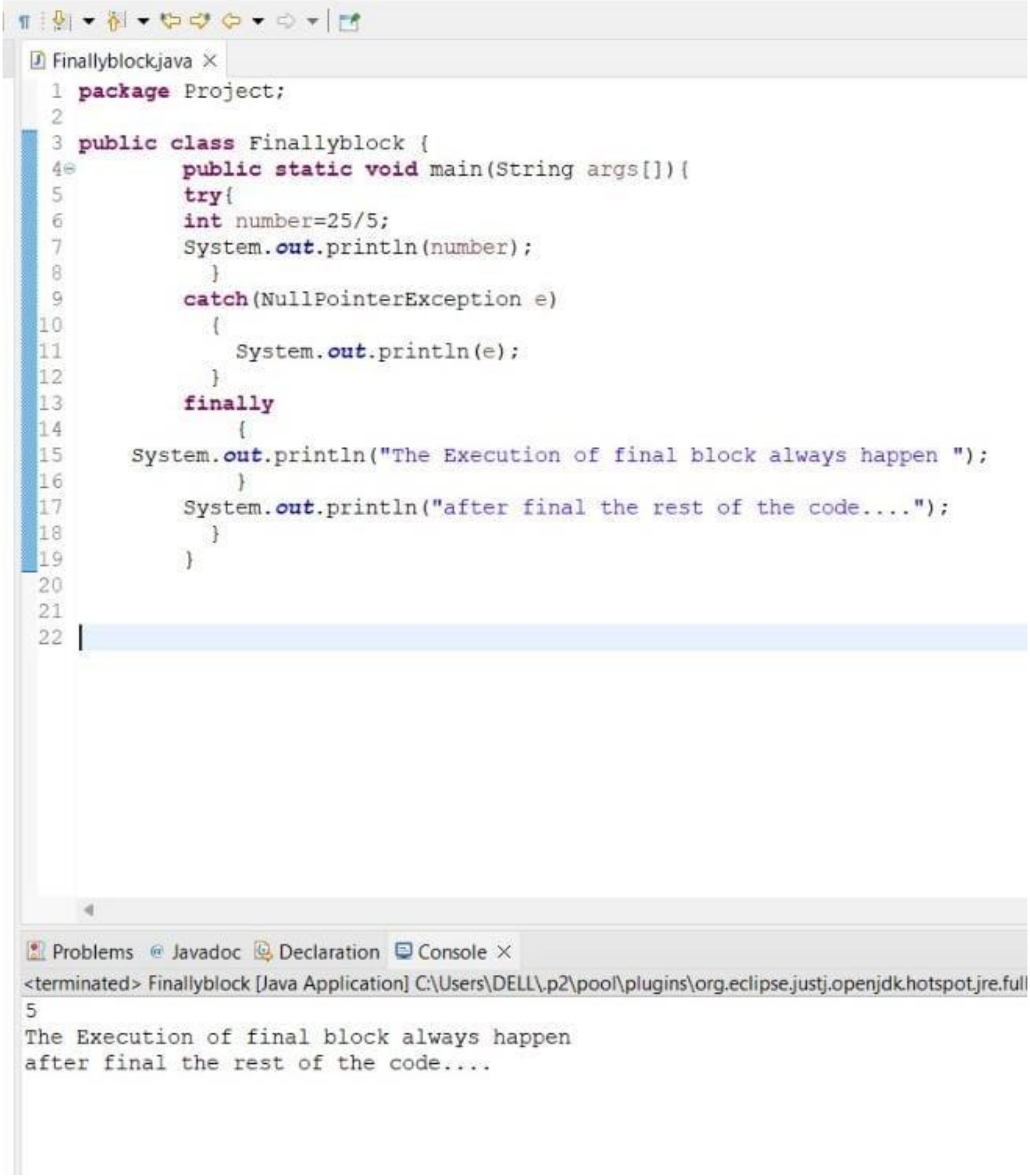


```
package Project;

public class nestedtrycatchblock {
    class Excep1{
        public static void main(String args[]){
        try{
        try{
        System.out.println("going to divide");
        int b =39/0;
        }catch(ArithmeticException e){System.out.println(e);}

        try{
        int a[]=new int[5];
        a[5]=4;
            }
        catch(ArrayIndexOutOfBoundsException e)
        {
        System.out.println(e);
             }

        System.out.println("another statement");
      }
        catch(Exception e)
        {
        System.out.println("handeled");
        }

        System.out.println("normal flow..");
         }
     }
}
```

<terminated> Multiplecatchblock [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.op

```
ArrayIndexOutOfBounds exception
rest of the code
```

12) Writing code for finally{} block



```
package Project;

public class Finallyblock {
        public static void main(String args[]){
        try{
        int number=25/5;
        System.out.println(number);
          }
        catch(NullPointerException e)
          {
            System.out.println(e);
          }
        finally
            {
    System.out.println("The Execution of final block always happen ");
            }
        System.out.println("after final the rest of the code....");
          }
        }
```

```
<terminated> Finallyblock [Java Application] C:\Users\DELL\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full
5
The Execution of final block always happen
after final the rest of the code....
```