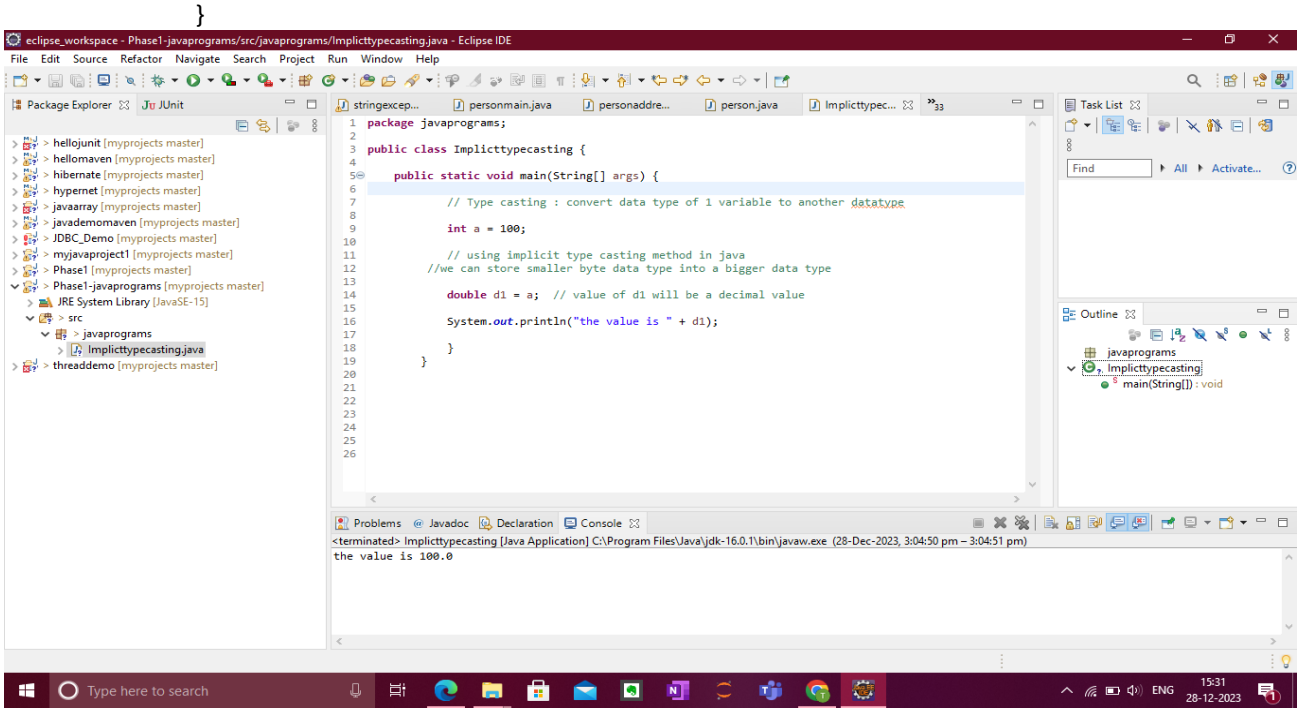
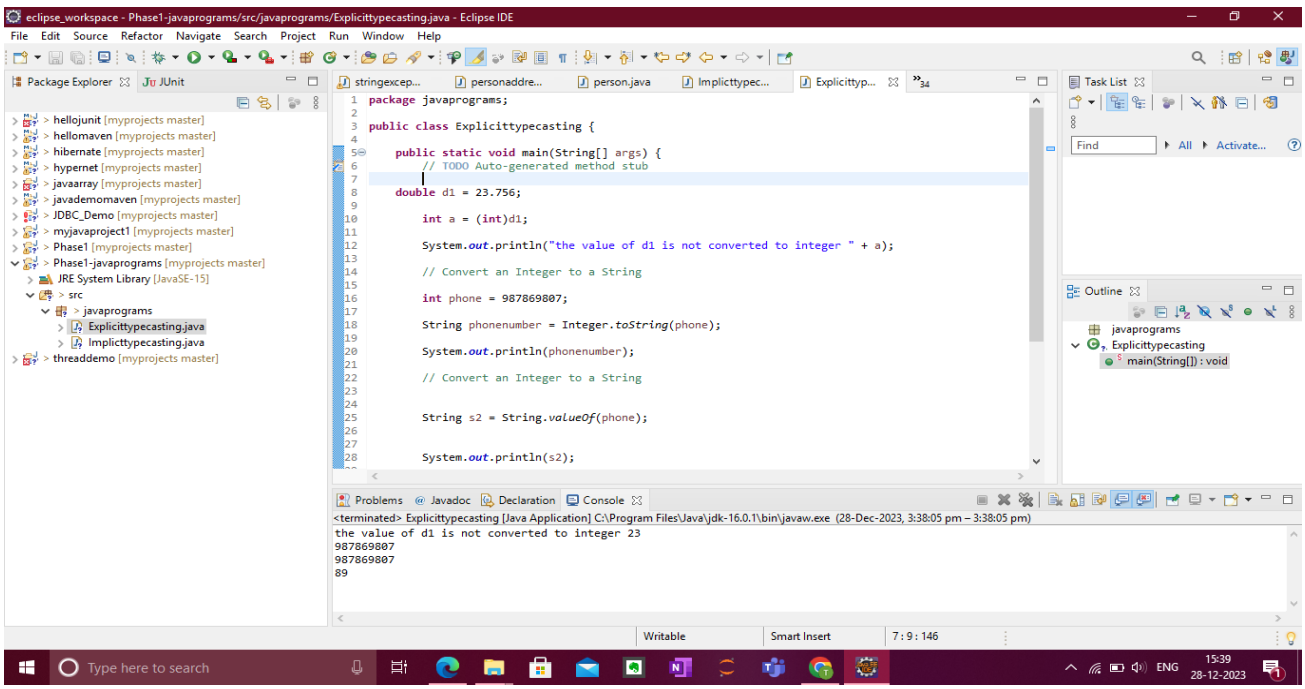
1.implict and explicit type casting :

```
package javaprograms;

public class Implicttypecasting {

        public static void main(String[] args) {

                // Type casting : convert data type of 1 variable to another datatype

                int a = 100;

                // using implicit type casting method in java
        //we can store smaller byte data type into a bigger data type

                double d1 = a;  // value of d1 will be a decimal value

                System.out.println("the value is " + d1);

                }
        }
```



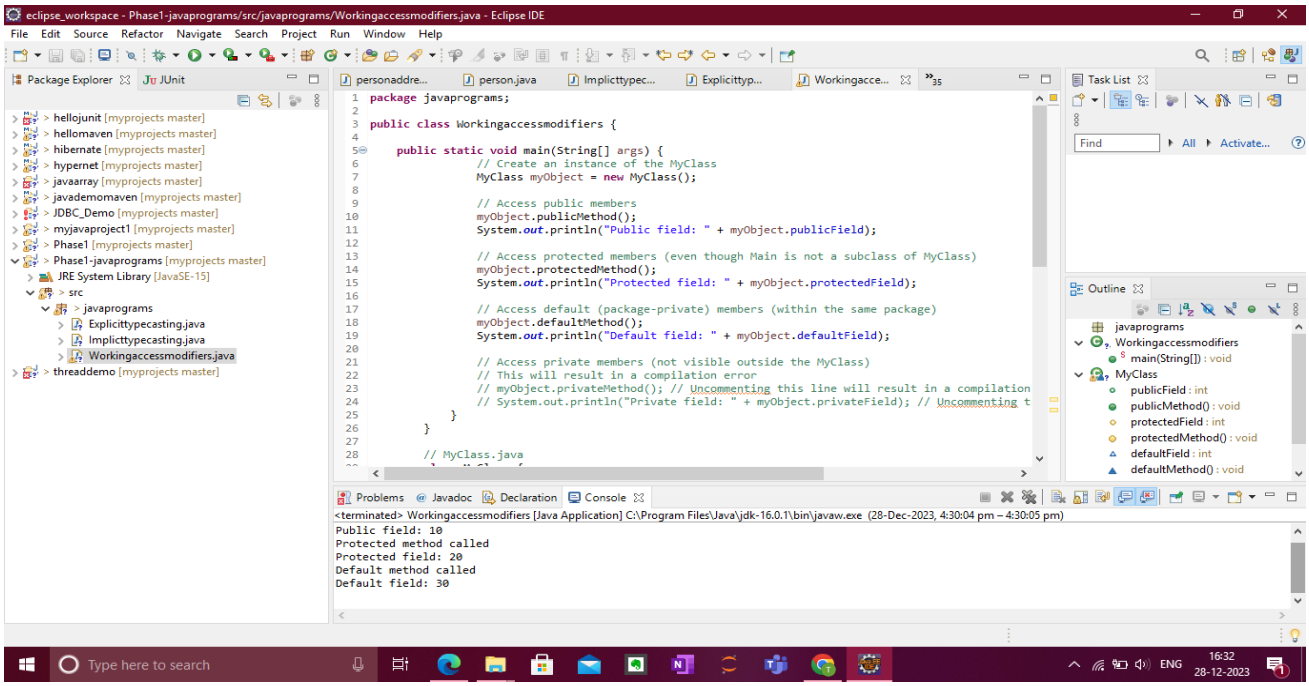

```
b.package javaprograms;

 public class Explicittypecasting {

        public static void main(String[] args) {

        double d1 = 23.756;

                int a = (int)d1;
```

```
        System.out.println("the value of d1 is not converted to integer " + a);

                // Convert an Integer to a String

                int phone = 987869807;

                String phonenumber = Integer.toString(phone);

                System.out.println(phonenumber);

                // Convert an Integer to a String


                String s2 = String.valueOf(phone);


                System.out.println(s2);

                // Convert String to an Integer

                String age = "89";

                int i = Integer.parseInt(age);

                System.out.println(i);


        }
}
```



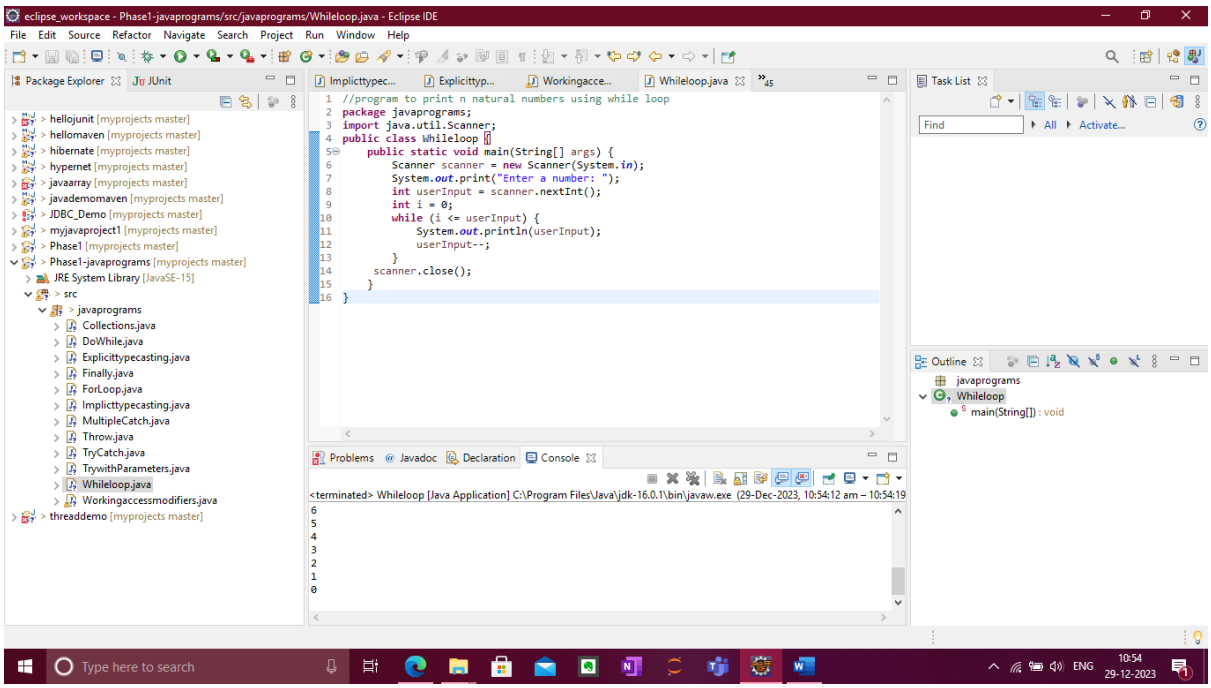
2.working of access modifiers :

```java
package javaprograms;
public class Workingaccessmodifiers {
	public static void main(String[] args) {
		MyClass myObject = new MyClass();
		myObject.publicMethod();
		System.out.println("Public field: " + myObject.publicField);
		myObject.protectedMethod();
		System.out.println("Protected field: " + myObject.protectedField);

		myObject.defaultMethod();
		System.out.println("Default field: " + myObject.defaultField);
	  }
	}
	class MyClass {
	  public int publicField = 10;
	  public void publicMethod() {
	    System.out.println("Public method called");
	  }
	  protected int protectedField = 20;
	  protected void protectedMethod() {
	    System.out.println("Protected method called");
	  }
	  int defaultField = 30;
	  void defaultMethod() {
	    System.out.println("Default method called");
	  }
	  private int privateField = 40;
	  private void privateMethod() {
	    System.out.println("Private method called");
	  }
	}
}
```



3.while loop :

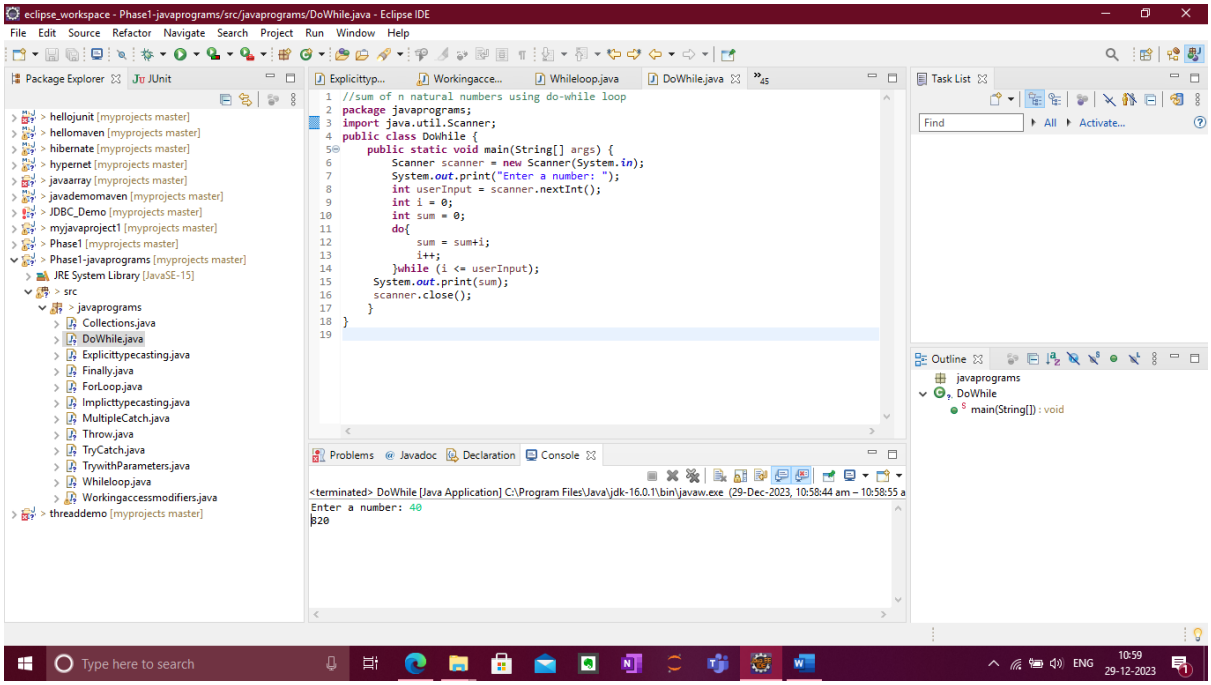
```
//program to print n natural numbers using while loop
package javaprograms;
import java.util.Scanner;
public class Whileloop {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number: ");
    int userInput = scanner.nextInt();
    int i = 0;
    while (i <= userInput) {
      System.out.println(userInput);
      userInput--;
    }
   scanner.close();
  }
}
```



4.Do while loop :

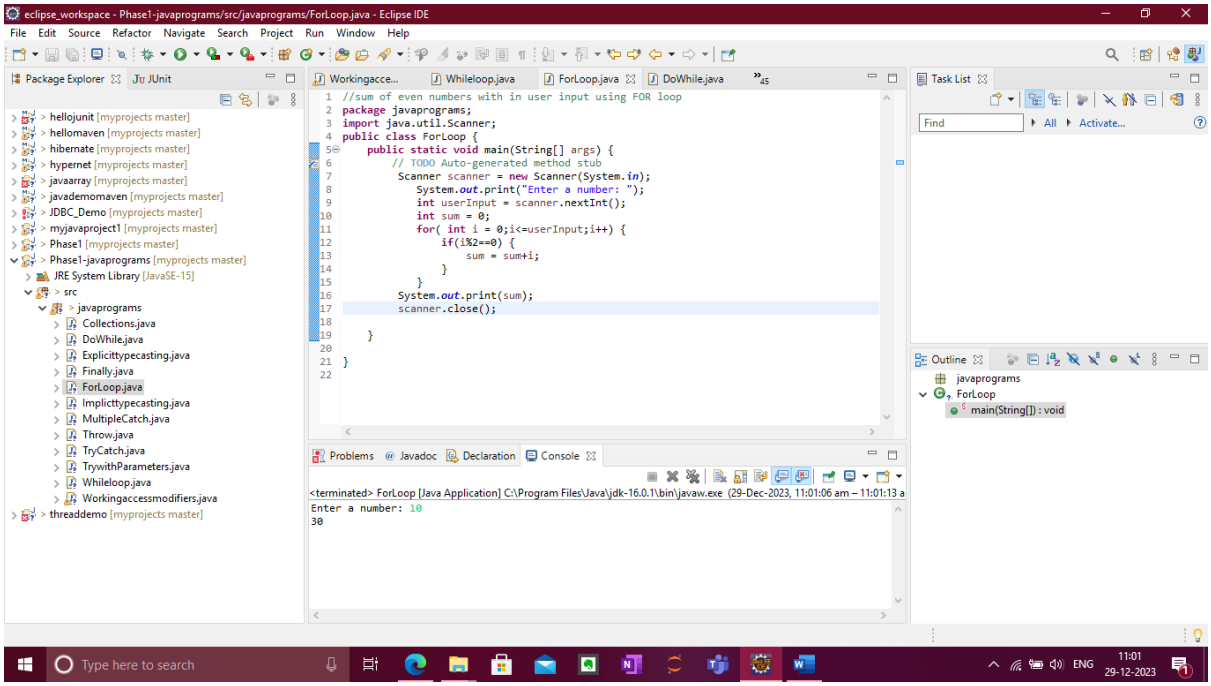
```
//sum of n natural numbers using do-while loop
package javaprograms;
import java.util.Scanner;
public class DoWhile {
        public static void main(String[] args) {
                Scanner scanner = new Scanner(System.in);
    System.out.print("Enter a number: ");
    int userInput = scanner.nextInt();
    int i = 0;
    int sum = 0;
```

```
    do{
      sum = sum+i;
      i++;
    }while (i <= userInput);
   System.out.print(sum);
   scanner.close();
         }
}
```



5.For loop :

```
package javaprograms;
import java.util.Scanner;
public class ForLoop {
        public static void main(String[] args) {
                // TODO Auto-generated method stub
                 Scanner scanner = new Scanner(System.in);
            System.out.print("Enter a number: ");
            int userInput = scanner.nextInt();
            int sum = 0;
            for( int i = 0;i<=userInput;i++) {
                if(i%2==0) {
                        sum = sum+i;
                }
            }
           System.out.print(sum);
           scanner.close();

        }

}
```



6. Classes,Objects,Constructors :

> creates an instance of a class
> similar to a java method,except :
> name is same as the class name
> it will not have a return type
> Whenever we write the keyword new, to create an instance of a class. A default constructor will be invoked and an object of the class is returned.
Types of constructors:
> default constructor
> the role of default contructor is to initialize the object and return it to the calling code
> default constructor is always without an argument

> No argument constructor
> parameterized constructor

```
package constructorDemo;
public class ConstructorDemo {
	// no argument constructor - syntax
	public ConstructorDemo()
	{
		System.out.println(" this is no argument constructor");
	}

	// a constructor with argument

	public ConstructorDemo( int a)
	{
		System.out.println(" this is  argument constructor");
		System.out.println(" this value of argument a : " + a);
	}

	public ConstructorDemo( int a, int b){
```

```
		System.out.println(" this is  multiple argument constructor");
		System.out.println(" this value of argument a : " + a);
		System.out.println(" this value of argument b : " + b);
	}
	public static void main(String[] args) {
		// to execute a constructor just create an object
		ConstructorDemo obj	= new ConstructorDemo();

		ConstructorDemo obj2	= new ConstructorDemo(23); // value of a is 23

		ConstructorDemo obj3	= new ConstructorDemo(68,34);

		}
```

7.Inheritence :

> define variables at class level -> Global variables
> these variable can be used in any method of the class
> A base Class in selenium will be useful to create
> methods/variables that are used by other classes
> A child class can inherit properties of a single parent class only.
> multiple inheritance not allowed, multiple extends keyword not allowed
> Multi level inheritance is allowed
> Calling static methods :

> call just with methodname ⇒
Example: methodname();
OR
> call with classname.methodname()
Example: BaseClass.login()
First create a package
> create a Base class→ this will super parent class.
> Create some base methods in it

```
package inheritenceDemo;
public class BaseClass {
public static void openbrowser(String browserName,String appURL)
	{
		System.out.println("Code to open the browser for testing in : " + browserName);
		System.out.println("Code to open the app on the browser for testing in : " + appURL);
		}
	public static void login(String userName, String password)
	{
		System.out.println("Code for username "+ userName);
		System.out.println("code for password" + password);
		System.out.println("User logged successfully");
	}
		 }
```

Create Class 2: inheriting Base class

```
package inheritenceDemo;

// we will make this class as a child to parent class -> BaseClass.java
// use the extends keyword
public class HomePageTest extends BaseClass {
	public static void testHomePage()
	{
		System.out.println("test code for search box");
```

```
		System.out.println("test code for addtoCart button");
		System.out.println("test code for todays delas link");
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		HomePageTest obj = new HomePageTest();

		BaseClass.openbrowser("Chrome", "amazon.com");
		BaseClass.login("admin", "password");
		obj.testHomePage();
		}
}
```

Create Class3 : inheriting class2 which inturn inherits the base class methods [multi- level inheritance]

```
package inheritenceDemo;
public class ProductPage extends HomePageTest{
public static void addProduct()
	{
		System.out.println("test code for add mobilephone");
		System.out.println("test code for add product to cart");
		System.out.println("test code for click on order button");
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		ProductPage obj = new ProductPage();

		BaseClass.openbrowser("FireFox", "amazon.com");
		BaseClass.login("admin", "135");
	      HomePageTest.testHomePage();
		obj.addProduct();

	}

}
```

Create class 4: this inherited class 3⇒ which inherited class 2⇒ class 2 inherited Base class
⇒ [multi level inheritance]

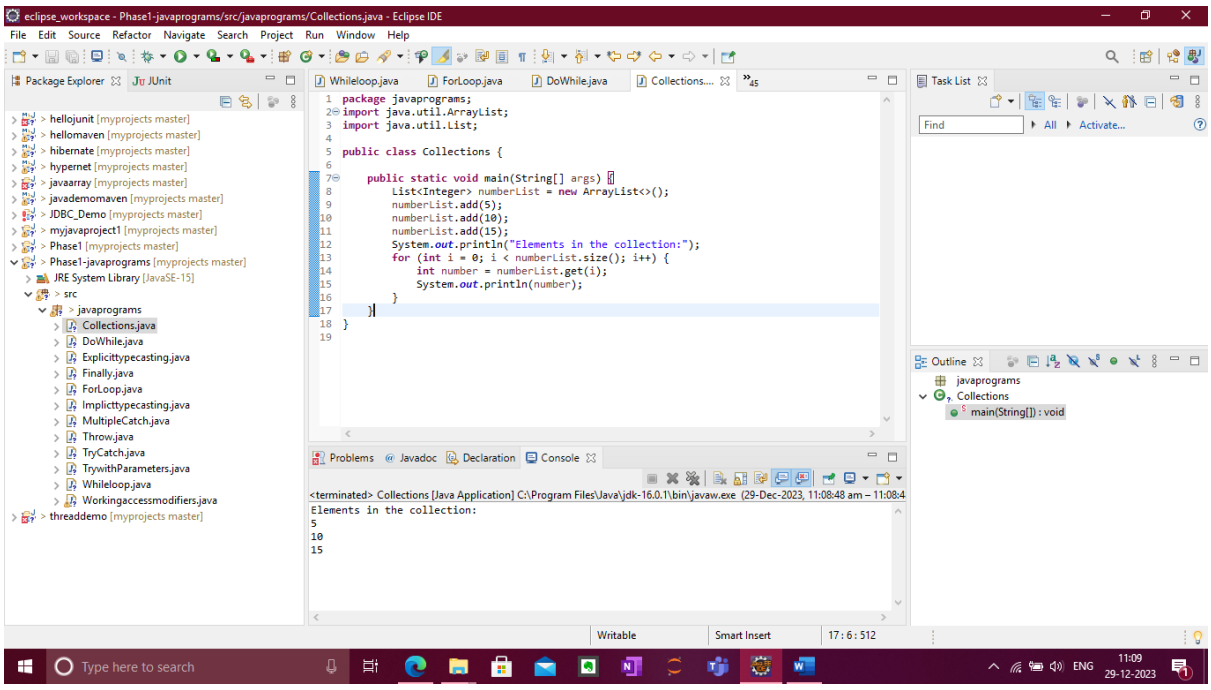
In this example class 4 can call methods of base class, class 2 , class3

```
package inheritenceDemo;
public class PaymentPageTest  extends ProductPage {
	public void makePayment()
	{
		System.out.println("test code for makepayment");
	}

	public static void main(String[] args) {
		// TODO Auto-generated method stub

		PaymentPageTest obj = new PaymentPageTest();

	    BaseClass.openbrowser("FireFox", "amazon.com");
		BaseClass.login("admin", "135");
		HomePageTest.testHomePage();   // classname.methodname() ==>static method
		addProduct();  // OR   static method ==> just methodname
```

```
        obj.makePayment();
            }

}
```

8. Collections :

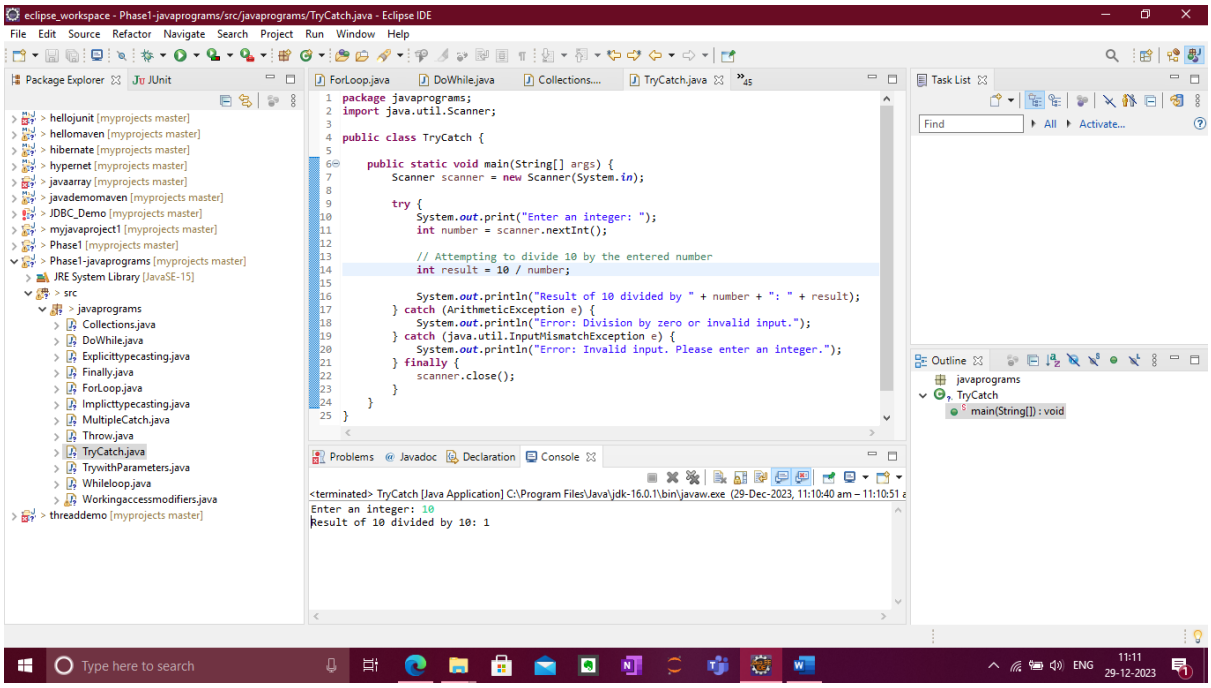
```
package javaprograms;
import java.util.ArrayList;
import java.util.List;

public class Collections {

   public static void main(String[] args) {
      List<Integer> numberList = new ArrayList<>();
      numberList.add(5);
      numberList.add(10);
      numberList.add(15);
      System.out.println("Elements in the collection:");
      for (int i = 0; i < numberList.size(); i++) {
         int number = numberList.get(i);
         System.out.println(number);
      }
   }
}
```



9.Try catch block :

```
package javaprograms;
```

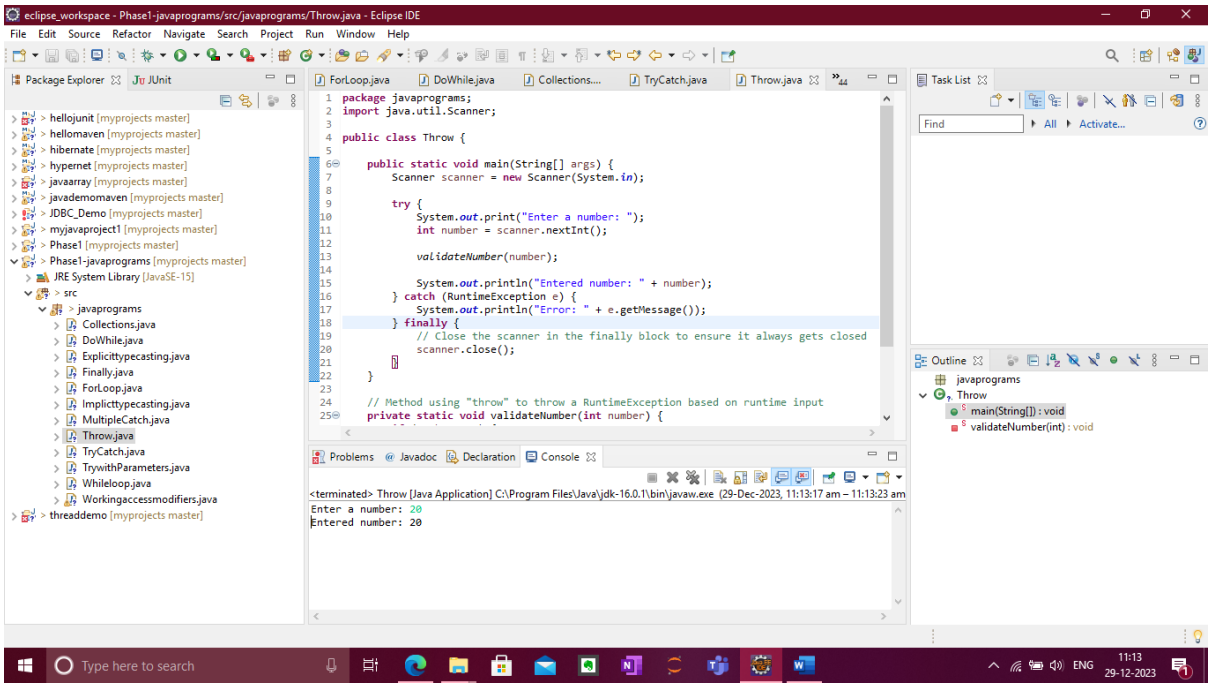
```java
import java.util.Scanner;

public class TryCatch {

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

      try {
         System.out.print("Enter an integer: ");
         int number = scanner.nextInt();

         // Attempting to divide 10 by the entered number
         int result = 10 / number;

         System.out.println("Result of 10 divided by " + number + ": " + result);
      } catch (ArithmeticException e) {
         System.out.println("Error: Division by zero or invalid input.");
      } catch (java.util.InputMismatchException e) {
         System.out.println("Error: Invalid input. Please enter an integer.");
      } finally {
         scanner.close();
      }
   }
}
```



10.Throwns and throw keyword :

```java
package javaprograms;
import java.util.Scanner;

public class Throw {

   public static void main(String[] args) {
```

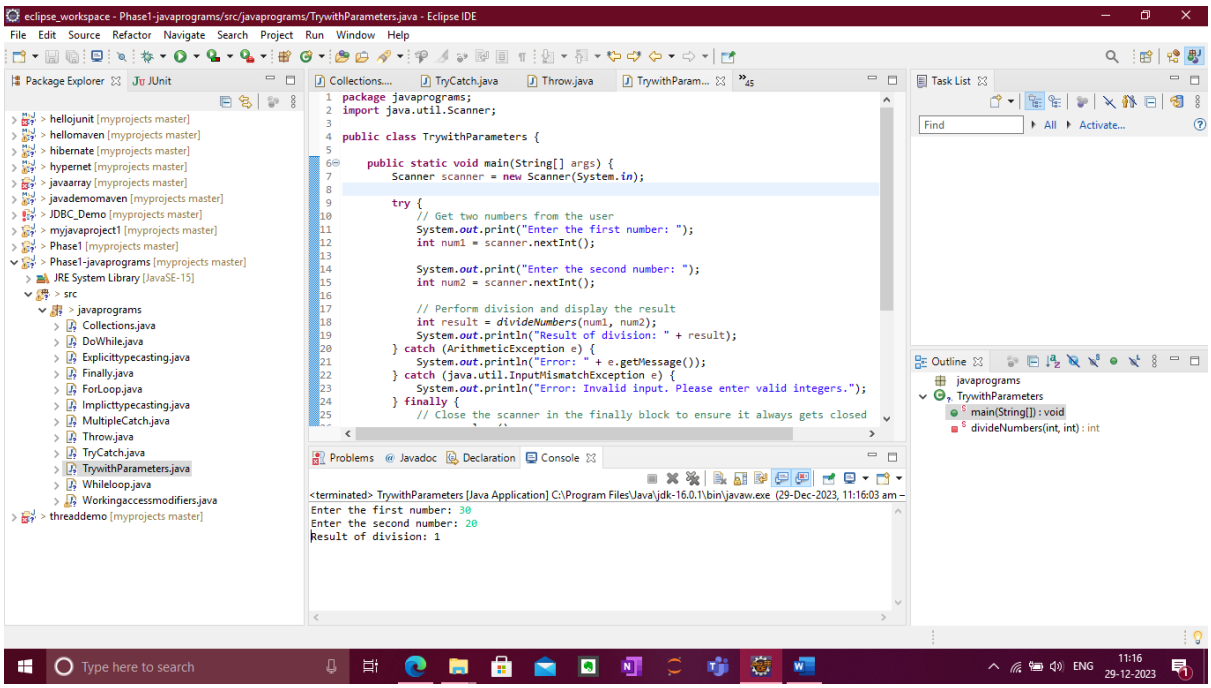
```
    Scanner scanner = new Scanner(System.in);

    try {
      System.out.print("Enter a number: ");
      int number = scanner.nextInt();

      validateNumber(number);

      System.out.println("Entered number: " + number);
    } catch (RuntimeException e) {
      System.out.println("Error: " + e.getMessage());
    } finally {
      // Close the scanner in the finally block to ensure it always gets closed
      scanner.close();
    }
  }

  // Method using "throw" to throw a RuntimeException based on runtime input
  private static void validateNumber(int number) {
    if (number < 0) {
      throw new RuntimeException("Negative numbers are not allowed");
    }
  }
}
```



11. Try block with prameters :

```
package javaprograms;
import java.util.Scanner;

public class TrywithParameters {

  public static void main(String[] args) {
```

```
    Scanner scanner = new Scanner(System.in);

    try {
      // Get two numbers from the user
      System.out.print("Enter the first number: ");
      int num1 = scanner.nextInt();

      System.out.print("Enter the second number: ");
      int num2 = scanner.nextInt();

      // Perform division and display the result
      int result = divideNumbers(num1, num2);
      System.out.println("Result of division: " + result);
    } catch (ArithmeticException e) {
      System.out.println("Error: " + e.getMessage());
    } catch (java.util.InputMismatchException e) {
      System.out.println("Error: Invalid input. Please enter valid integers.");
    } finally {
      // Close the scanner in the finally block to ensure it always gets closed
      scanner.close();
    }
  }

  // Method that divides two numbers and throws an ArithmeticException if the second number is zero
  private static int divideNumbers(int numerator, int denominator) {
    if (denominator == 0) {
      throw new ArithmeticException("Cannot divide by zero");
    }
    return numerator / denominator;
  }
}
```



12. Catch blocks :

```java
package javaprograms;
import java.util.Scanner;

public class MultipleCatch {

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

    try {
      // Get two numbers from the user
      System.out.print("Enter the first number: ");
      int num1 = scanner.nextInt();

      System.out.print("Enter the second number: ");
      int num2 = scanner.nextInt();

      // Perform division and display the result
      int result = divideNumbers(num1, num2);
      System.out.println("Result of division: " + result);
    } catch (ArithmeticException e) {
      System.out.println("Error: " + e.getMessage());
    } catch (java.util.InputMismatchException e) {
      System.out.println("Error: Invalid input. Please enter valid integers.");
    } finally {
      // Close the scanner in the finally block to ensure it always gets closed
      scanner.close();
    }
  }

  // Method that divides two numbers and throws an ArithmeticException if the second number is zero
  private static int divideNumbers(int numerator, int denominator) {
    if (denominator == 0) {
      throw new ArithmeticException("Cannot divide by zero");
    }
    return numerator / denominator;
  }
}
```

13. Finally block :

```java
package javaprograms;

import java.util.Scanner;

public class Finally{

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

         // Get a number from the user
          System.out.print("Enter a number: ");
```

```
            int number = scanner.nextInt();

            // Display the entered number

            System.out.println("Entered number: " + number);

            // Simulate some processing

            int result = processNumber(number);
            System.out.println("Processed result: " + result);

            } catch (java.util.InputMismatchException e) {

            System.out.println("Error: Invalid input. Please enter a valid integer.");

            } finally {

            // Always close the scanner in the finally block to ensure it gets closed
System.out.println("Finally block: Closing the scanner.");

            scanner.close();
            }
 }

  // Method simulating some processing

            private static int processNumber(int number) {

            // Simulate processing by doubling the number

            return number * 2;
            }

}
```