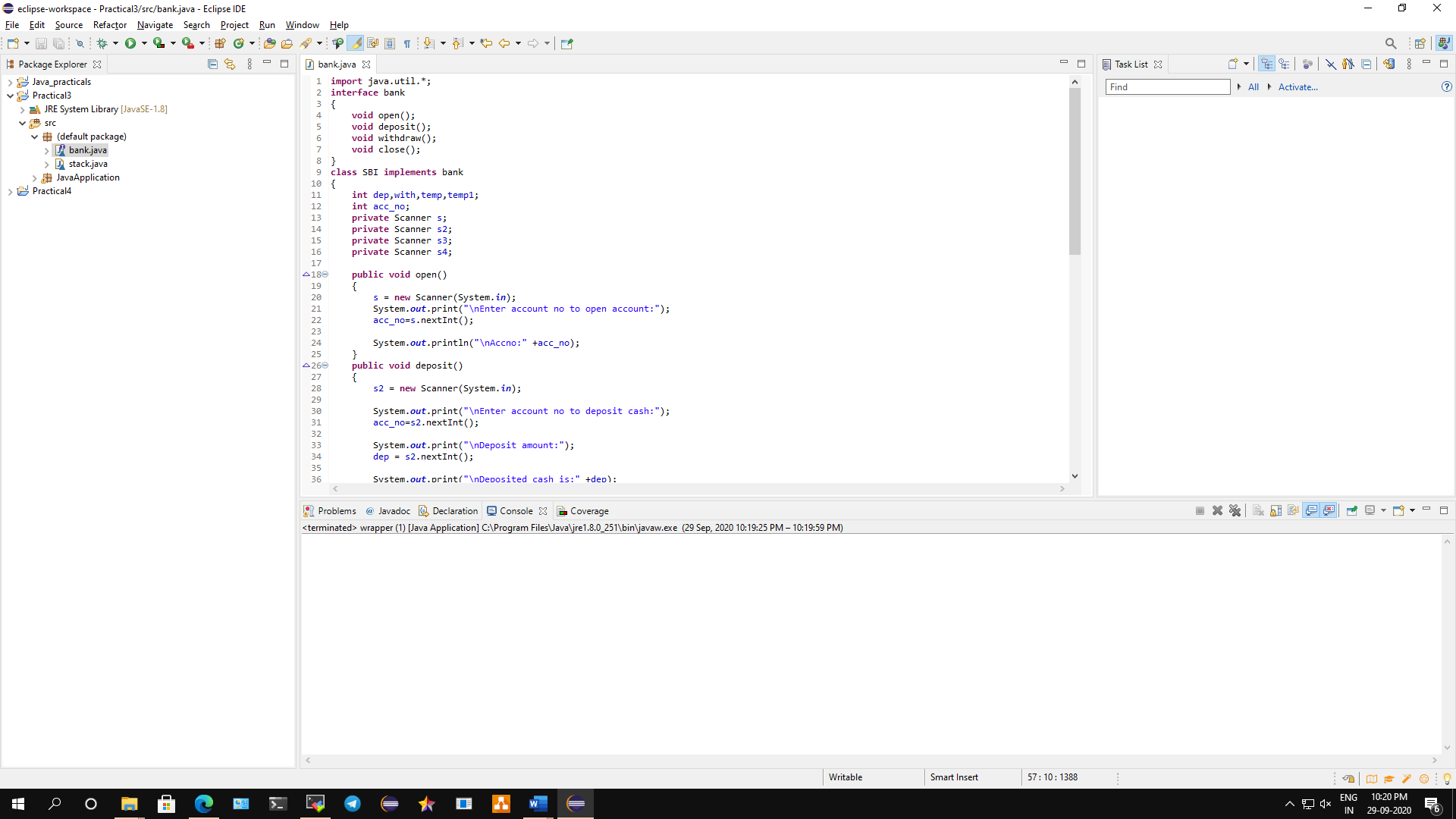
**Practical No 3**

**A) Program to demonstrate interface: Create an interface for Bank with following operation: -Deposit, Withdraw, Account open, Close account.**

**Aim: Write a program to demonstrate interface: Create an interface for Bank with following operation: -Deposit, Withdraw, Account open, Close account in java.**

**Description:**

**Interface in java** is a blueprint of a class. It has static constants and abstract methods. The interface in Java is a mechanism to achieve abstraction. There can be only abstract methods in the Java interface, not method body. It is used to achieve abstraction and multiple inheritance in Java. In other words, you can say that interfaces can have abstract methods and variables. It cannot have a method body. Java Interface also **represents the IS-A relationship**. It cannot be instantiated just like the abstract class. In this program we have defined the interface as bank, then we have defined the main class as SBI which implements bank interface then we have defined the private scanner class like S1, S2, S3 and S4. Scanner S1 will ask for the account number. Scanner S2 will ask for the account number in which you want to deposit cash. Scanner S3 will ask to enter the account number in which you want to withdraw cash and Scanner S4 will ask the account number in which you want to close the account. And then we have extended the class interface with bank1.



**Conclusion: We have written a program to demonstrate interface in java.**

**Code:**

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

**interface** bank

{

**void** open();

**void** deposit();

**void** withdraw();

**void** close();

}

**class** SBI **implements** bank

{

**int** dep,with,temp,temp1;

**int** acc\_no;

**private** Scanner s;

**private** Scanner s2;

**private** Scanner s3;

**private** Scanner s4;

**public** **void** open()

{

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

System.***out***.print("\nEnter account no to open account:");

acc\_no=s.nextInt();

System.***out***.println("\nAccount number:" +acc\_no);

}

**public** **void** deposit()

{

s2 = **new** Scanner(System.***in***);

System.***out***.print("\nEnter account no to deposit cash:");

acc\_no=s2.nextInt();

System.***out***.print("\nDeposit amount:");

dep = s2.nextInt();

System.***out***.print("\nDeposited cash is:" +dep);

}

**public** **void** withdraw()

{

s3 = **new** Scanner(System.***in***);

System.***out***.print("\n\nEnter account no to withdraw cash:");

acc\_no=s3.nextInt();

System.***out***.print("\nWithdraw cash:");

with=s3.nextInt();

**if**(with<dep)

{

dep=dep-with;

System.***out***.println("\nWithdrew cash is:" +with);

System.***out***.println("\nBalance cash is:" +dep);

}

**else**

{

System.***out***.print("\nNo funds");

}

}

**public** **void** close()

{

s4 = **new** Scanner(System.***in***);

System.***out***.print("\nEnter account to close account:");

acc\_no=s4.nextInt();

System.***out***.println("\nAccount Closed");

System.*exit*(0);

}

}

**class** bank1

{

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

{

SBI obj = **new** SBI();

obj.open();

obj.deposit();

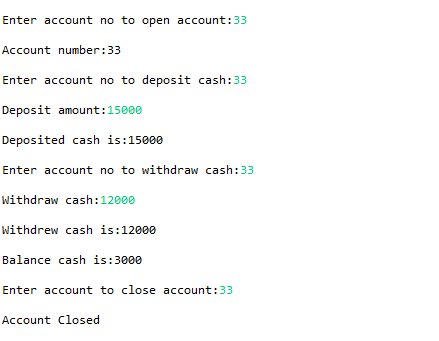
obj.withdraw();

obj.close();

}

}

**Output:**

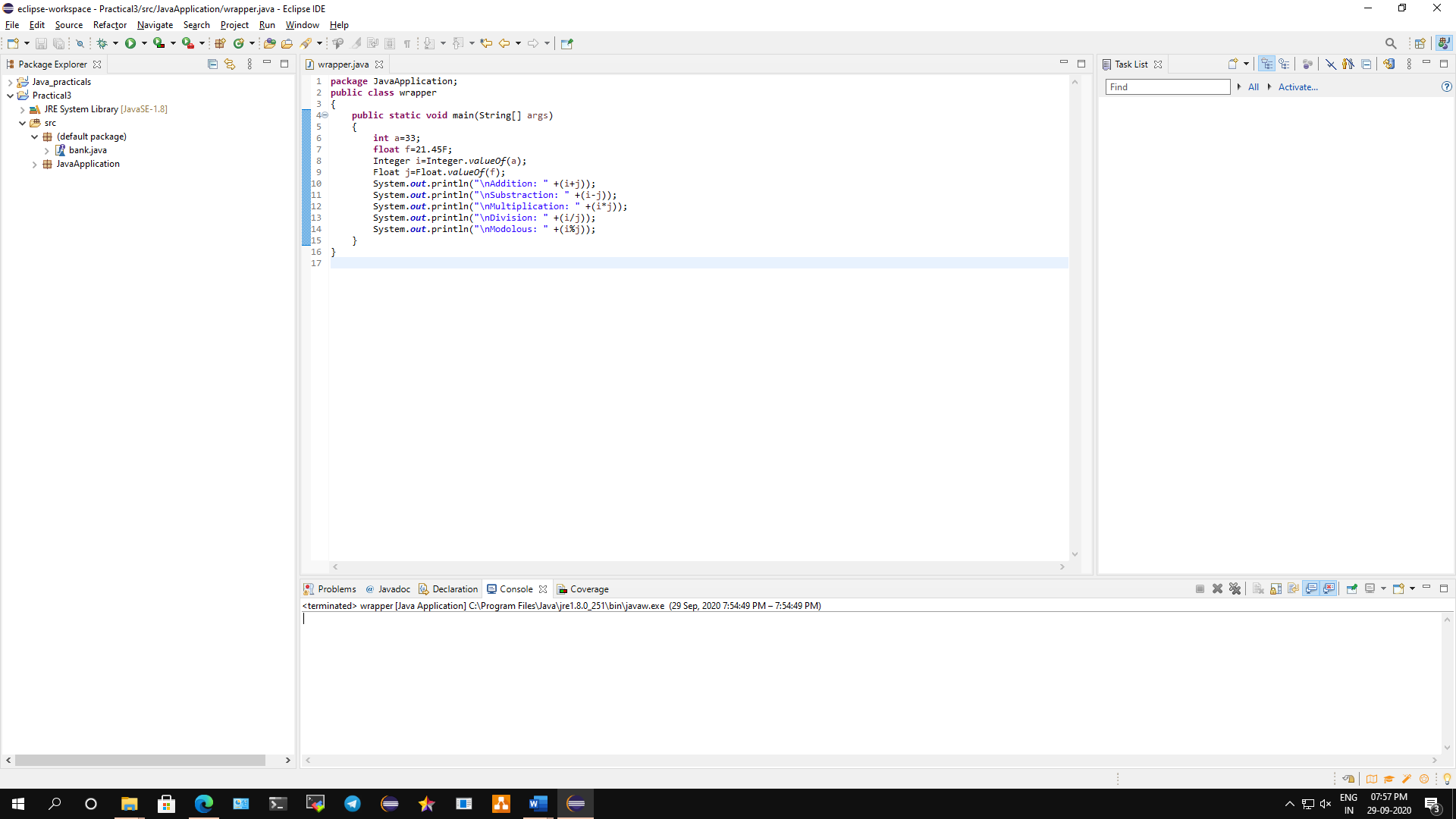


**B) Program to implement basic arithmetic operation using Wrapper Class.**

**Aim: Write a java program to implement basic arithmetic operation using Wrapper Class.**

**Description:**

Wrapper classes are those whose objects wraps a primitive data type within them. In the java.lang package java provides a separate class for each of the primitive data types namely Byte, Character, Double, Integer, Float, Long, Short. At the time of instantiation, these classes accept a primitive datatype directly, or in the form of String. Wrapper classes provide methods to, convert primitive datatypes within them to String objects and, to compare them with other objects etc. Using wrapper classes, you can also add primitive datatypes to various Collection objects such as ArrayList, HashMap etc. You can also pass primitive values over a network using wrapper classes. We have defined JavaApplication as packages and wrapper as the public class. Then we defined main function which will read integer and float. Once it read those functions it will solve the basic arithmetic operation and we will get the output.



**Conclusion: We have implemented a program to perform basic arithmetic operation using Wrapper Class.**

**Code:**

**package** JavaApplication;

**public** **class** wrapper

{

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

{

**int** a=33;

**float** f=21.45F;

Integer i=Integer.*valueOf*(a);

Float j=Float.*valueOf*(f);

System.***out***.println("\nAddition: " +(i+j));

System.***out***.println("\nSubstraction: " +(i-j));

System.***out***.println("\nMultiplication: " +(i\*j));

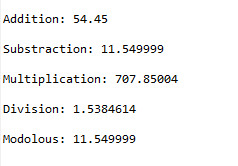
System.***out***.println("\nDivision: " +(i/j));

System.***out***.println("\nModolous: " +(i%j));

}

}

**Output:**



**C) Implement a program to demonstrate basic in-built function used in String**

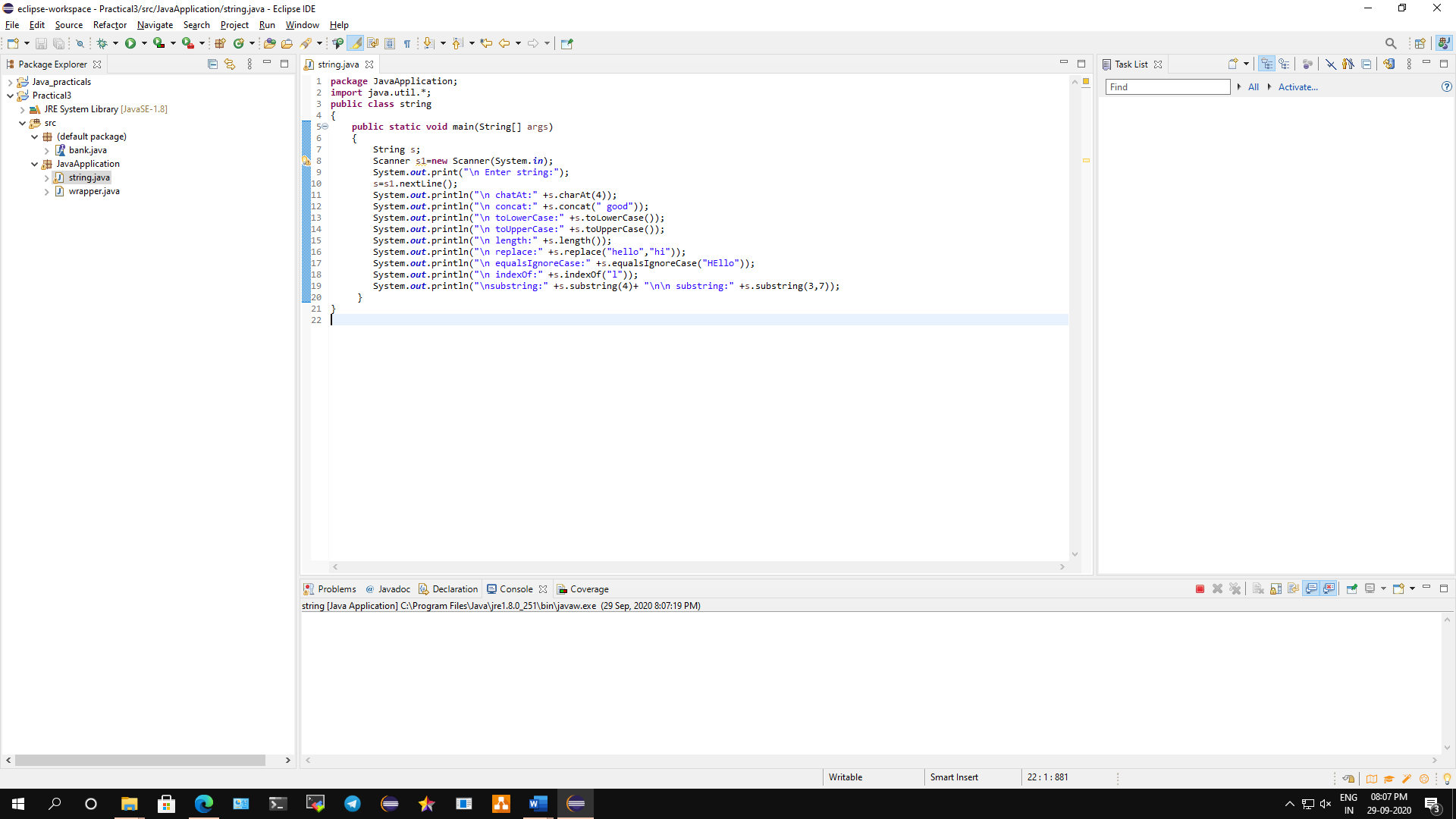
**Class.**

**Aim: Implement a java program to demonstrate basic in-built function used in String Class.**

**Description:**

Strings, which are widely used in Java programming, are a sequence of characters. In Java programming language, strings are treated as objects.

The Java platform provides the String class to create and manipulate strings. In the given code we have created a package name JavaApplication and public class string then we have defined the main method. We use string s which write the string then we have used scanner function and then we defined different string types such as uppertolower, lowertoupper, concat, length etc. Then we will input the string for e.g. “Hello World” and it will execute the code and return the output of different string types.



**Conclusion: We have implemented a java program to demonstrate basic in-built function used in String Class.**

**Code:**

**package** JavaApplication;

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

**public** **class** string

{

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

{

String s;

Scanner s1=**new** Scanner(System.***in***);

System.***out***.print("\n Enter string:");

s=s1.nextLine();

System.***out***.println("\n chatAt:" +s.charAt(4));

System.***out***.println("\n concat:" +s.concat(" good"));

System.***out***.println("\n toLowerCase:" +s.toLowerCase());

System.***out***.println("\n toUpperCase:" +s.toUpperCase());

System.***out***.println("\n length:" +s.length());

System.***out***.println("\n replace:" +s.replace("hello","hi"));

System.***out***.println("\n equalsIgnoreCase:" +s.equalsIgnoreCase("HEllo"));

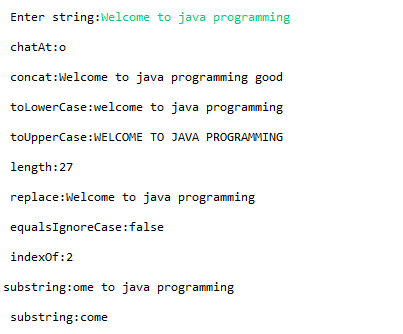
System.***out***.println("\n indexOf:" +s.indexOf("l"));

System.***out***.println("\nsubstring:" +s.substring(4)+ "\n\n substring:" +s.substring(3,7));

}

}

**Output:**

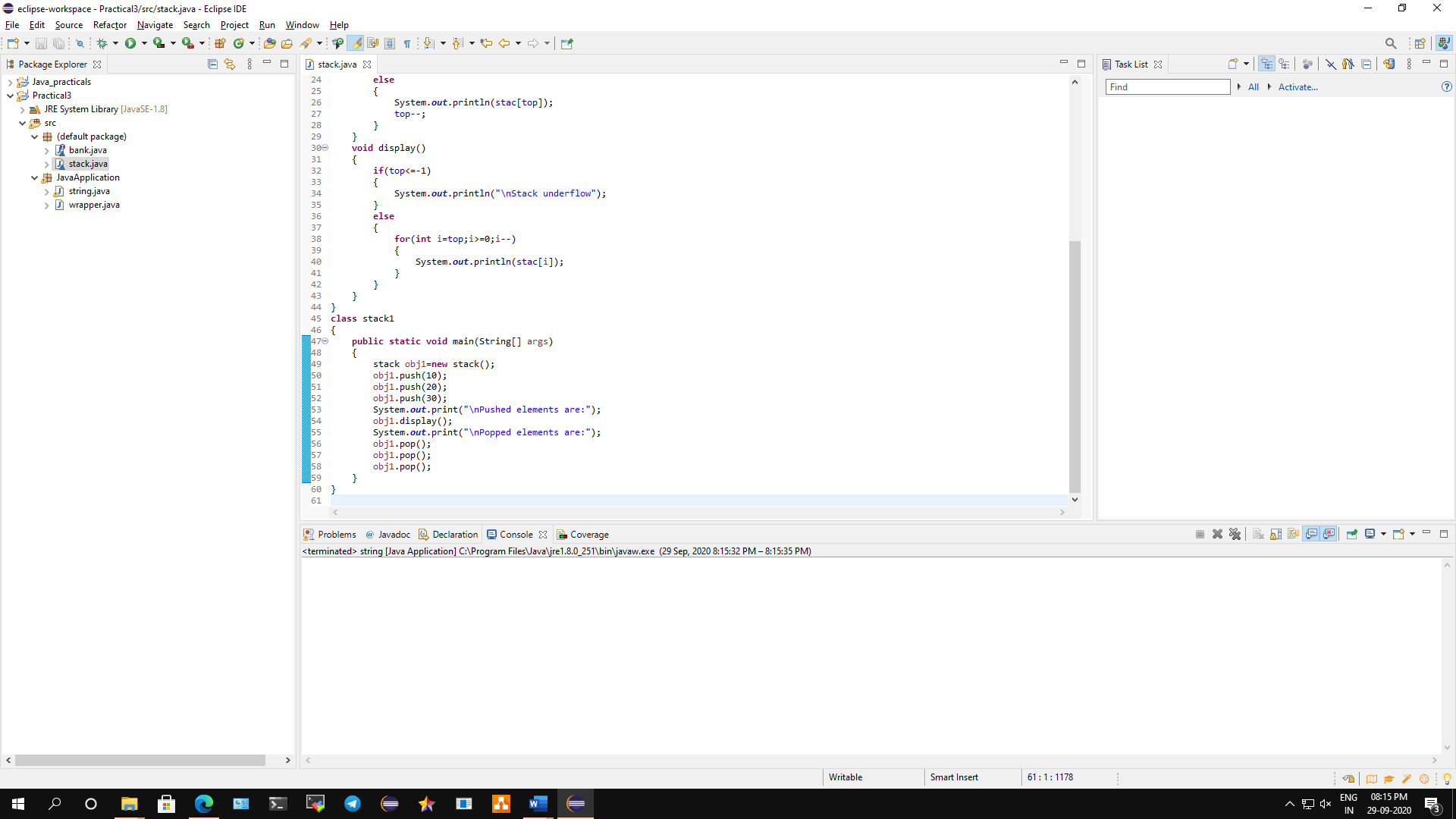


**D) To implement a program to implement Queue/Stack as ADT in JAVA.**

**Aim: Implement a java program to implement Queue/Stack as ADT.**

**Description:**

A Stack is a linear data structure which follows a particular order in which the operations are performed. The order may be LIFO (Last in First Out) or FILO (First in Last Out). There are two basic operations are performed in the stack: Push and Pop. Push Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition. Whereas Pop Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition. In the given program we defined the class stack as main class and then we have defined the pop and push functions. In the next iteration we void push which helps to push the element in stack and pop function helps to pop the element in stack and then we have extended class called as class stack1 which runs the main function to display the output.



**Conclusion: We have implemented a java program to implement Queue/Stack as ADT.**

**Code:**

**class** stack

{

**int** top,n;

**int** stac[]=**new** **int**[10];

**void** push(**int** n)

{

**this**.n=n;

**if**(top>n-1)

{

System.***out***.println("\nStack overflow");

}

**else**

{

top++;

stac[top]=n;

}

}

**void** pop()

{

**if**(top<=-1)

{

System.***out***.println("\nStack underflow");

}

**else**

{

System.***out***.println(stac[top]);

top--;

}

}

**void** display()

{

**if**(top<=-1)

{

System.***out***.println("\nStack underflow");

}

**else**

{

**for**(**int** i=top;i>=0;i--)

{

System.***out***.println(stac[i]);

}

}

}

}

**class** stack1

{

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

{

stack obj1=**new** stack();

obj1.push(20);

obj1.push(30);

obj1.push(40);

System.***out***.print("\nPushed elements are: ");

obj1.display();

System.***out***.print("\nPopped elements are: ");

obj1.pop();

obj1.pop();

obj1.pop();

}

}

**Output:**

