```
import java.util.*;
class ElectricityBill {
int consumernumber;
String consumername;
int previousreading;
int currentreading;
int reading;
double amount;
boolean domestic;
public void setConsumerNumber(int num){
this.consumernumber=num;
}
public void setConsumerName(String name){
this.consumername=name;
}
public void setPreviousMonthReading(int previous_month_reading){
this.previousreading=previous_month_reading;
}
public void setCurrentMonthReading(int currentmonthreading){
this.currentreading=currentmonthreading;
}
public void setIsDomestic(boolean isdomestic) {
this.domestic=isdomestic;
}
public float calculateBillAmount(){
reading=currentreading-previousreading;
if (domestic){
if(reading<=100){
amount=reading*1;
}
else if(reading>100 && reading<=200){
amount=100+(reading - 100)*2.50;
}
else if(reading>200 && reading<=500){
amount=100+ 250+(reading-200)*4;
}
else if(reading>500){
amount =100+250+1200+(reading-500)*6;
}
}
else{
if(reading<-100){
amount=reading*2;
}
else if(reading>100 && reading<=200){
amount=200+(reading-100)*4.50;
}
else if(reading>200 && reading langle=500){
amount =260+450+(reading - 200)*6;
}
else if(reading>500) {
amount 200+ 450+ 1800+(reading-500)*7;
}
}
return (float)amount;
}
```

Currency conerter

```
package currencyconverter;
import java.io.*;
import java.util.Scanner;
public class CurrencyConverter
{
public double dollar ToRupee(double fromVal){
return fromVal*69.00;
}
public double rupeeToDollar(double fromVal){
return fromVal/69.00;
}
public double euroToRupee (double fromVal){
return fromVal*78.00;
}
public double rupeeToEuro(double fromVal){
return fromVal/78.00;
}
public double yenToRupee (double fromVal){
return fromVal*0.64;
}
public double rupeeToYen(double fromVal){
return fromVal/0.64;
}
```

Distance converter

```
package currencyconverter;
import java.io.*;
import java.util.Scanner;
public class DistanceConverter{
public double meterTokilometer (double meters){
return meters/1000;
}
public double kilometerToMeter(double kilometer) {
return kilometer*1000:
}
public double mileToKilometer(double mile){
return mile*1.609;
}
public double kilometerToMile(double kilometer){
return kilometer/1.609;
}}
```

Time converter

```
import java.util.Scanner;
public class TimeConverter
{
public double hour ToMinute(double hours){
return hours*60;
}
public double hour ToSecond(double hour){
return hour*3600;
}
public double minuteToHour(double minutes){
return minutes/60;
}
public double secondToHour (double seconds){
return seconds/3600;
}
}
```

```
package employee;
import java.io.IOException;
import java.util.Scanner;
class Emp
{
String ename,Address,email;
int eid;
int mobile;
void getEmployeedetails()
{
Scanner in = new Scanner(System.in);
System.out.println("Enter the Emp_id. :");
eid=in.nextInt();
System.out.println("Enter the Employee Name:");
ename=in.next();
System.out.println("Enter the Employee Address:");
Address=in.next();
System.out.println("Enter the Employee Email id :");
email=in.next();
System.out.println("Enter the Mobile No:");
mobile=in.nextInt();
}
void pay_calulation(double BasicPay)
{
double DA,HRA,PF,Sfund,Gross_Salary,Netsalary;
DA=BasicPay*0.97;
HRA=BasicPay*0.10;
PF=BasicPay*0.12;
Sfund=BasicPay*0.1;
Gross_Salary=BasicPay+DA+HRA;
Netsalary=Gross_Salary-(PF+Sfund);
System.out.println("Gross salary of the Employee"+Gross_Salary);
System.out.println("Net salary of the Employee: "+Netsalary);
}
void display()
{
System.out.println("Emp_id:"+eid);
System.out.println("Employee Name:"+ename);
System.out.println("Employee Address:"+Address);
System.out.println("Employee Email id :"+email);
System.out.println("Employee Mobile No:"+mobile);
}
}
class Programmer extends Emp
{
double BasicPay;
void Programmerdetails()
{
getEmployeedetails();
Scanner in = new Scanner(System.in);
System.out.println("Enter the Basic Pay of the Programmer:");
BasicPay=in.nextInt();
display();
pay_calulation(BasicPay);
}
}
class AssistantProfessor extends Emp
{
void APDetails()
{
```

```
double BasicPay;
getEmployeedetails();
Scanner in = new Scanner(System.in);
System.out.println("Enter the Basic Pay of the AssistantProfessor:");
BasicPay=in.nextInt();
display();
pay_calulation(BasicPay);
}
}
class AssociateProfessor extends Emp
{
double BasicPay;
void ASPDetails()
{
getEmployeedetails();
Scanner in = new Scanner(System.in);
System.out.println("Enter the Basic Pay of the AssociateProfessor:");
BasicPay=in.nextInt();
display();
pay_calulation(BasicPay);
}
}
class Professor extends Emp
{
double BasicPay;
void profDetails()
{
getEmployeedetails();
Scanner in = new Scanner(System.in);

System.out.println("Enter the Basic Pay of the Professor:");
BasicPay=in.nextInt();
display();
pay_calulation(BasicPay);
}
}
public class Employee
{
public static void main(String[] args)
{
Scanner in = new Scanner(System.in);
System.out.println("Choose the type Employee");
System.out.println("1.Programmer ,2.Assistant Professor,3.Associate
Professor ,4.Professor: ");
int ch=in.nextInt();
switch(ch)
{
case 1: System.out.println("PROGRAMMER DETAILS");
Programmer p=new Programmer();
p.Programmerdetails();
break;
case 2: System.out.println("Assistant Professor DETAILS");
AssistantProfessor ap=new AssistantProfessor();
ap.APDetails();
break;
case 3: System.out.println("Associate Professor DETAILS");
AssociateProfessor asp=new AssociateProfessor();
asp.ASPDetails();
break;
case 4: System.out.println("Professor DETAILS");
```

```
Professor pf=new Professor();
pf.profDetails();
break;
}
} }
```

```
package stackadt;
import java.io.*;
interface Mystack
{
public void pop();
public void push();
public void display();
}
class Stack_array implements Mystack
{
final static int n=5;
int stack[]=new int[n];
int top=-1;
public void push()
{
try
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));

if(top==(n-1))
{
System.out.println(" Stack Overflow");
return;
}
else
{
System.out.println("Enter the element");
int ele=Integer.parseInt(br.readLine());
stack[++top]=ele;
}
}
catch(IOException e)
{
System.out.println("e");
}
}
public void pop()
{
if(top<0)
{
System.out.println("Stack underflow");
return;
}
else
{
int popper=stack[top];
top--;
System.out.println("Popped element:" +popper);
}
}
public void display()
{
if(top<0)
{
System.out.println("Stack is empty");
return;
}
else

{
```

```
String str=" ";
for(int i=0; i<=top; i++)
str=str+" "+stack[i]+" <--";
System.out.println("Elements are:"+str);
}
}
}
class StackADT
{
public static void main(String arg[])throws IOException
{
BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
System.out.println("Implementation of Stack using Array");
Stack_array stk=new Stack_array();
int ch=0;
do
{
System.out.println("1.Push 2.Pop 3.Display 4.Exit");
System.out.println("Enter your choice:");
ch=Integer.parseInt(br.readLine());
switch(ch)
{
case 1:
stk.push();
break;
case 2:
stk.pop();
break;
case 3:
stk.display();
break;
case 4:
System.exit(0);
}
}
while(ch<5);
} }
```

```
import java.util.*:

public class Hello (

public static void display(List<String> strList) {
       for (String S: Strlist) {
            System.out.print(5+ " ");
        }
}
public static void main(String[] args) {
       Scanner sc new Scanner(System.in);
       int N= Integer.parseInt(sc.nextLine());
       List<String> strList = new ArrayList<>();
       for (int query = 1; query < N; query++) {
            int queryType sc.nextInt ();
            String str = **;
            switch (queryType) (

case 1:
     strlist.add(sc.nextLine().trim());
     break;
case 2:
     int index-sc.nextInt();
     strList.add(index,sc.nextLine().trim()):
     break;
case 3:
     System.out.println(strlist.indexOf (sc.nextLine().trim()));
     break;
case 4
     int n=0;
     String startChar=sc.nextLine().trim();
     for (String element:strlist)
    {
          if(element.startsWith(startChar))
          {
             System.out.print(element+" ");
             n=n+1:0
          }
     }
     if(n==0)
     {
          System.out.println("No such string");
      }
      else{
             System.out.println();
       }
       break;
case 5:
     String element=sc.nextLine().trim():
     if (strlist.contains (element))
     {
        strlist.remove(strlist.indexOf(element));
      }
       else(
             System.out.println("No such string");
       }
```

```
        break:
case 6:
        Collections.sort(strlist);
        break:
case 7:
        System.out.println(strlist.size());
        break:
case 8:
        if (!strlist.isEmpty()) {
        display(strlist):;
        System.out.println();
     } else {
          System.out.println("List is empty");
     }
     break;
     }
   }
  }
}
```

```
package javaapplication3;

abstract class shape
{
int a 3.b 4;

abstract public void print_area();
}
class rectangle extends shape
{
public int area_rect;

       @Override

public void print_area()
{
area _rect=a*b;

          System.out.println("The area of rectangle is:"+area_rect);
}
}
class triangle extends shape

{

int area tri;

     @Override

public void print_area()
{
area_tri= (int) (0.5* a* b);
```

```
                System.out.println("The area of triangle is:"+area_tri);
}
}
class circle extends shape
{
int area_circle;

       @Override

public void print_area()
{
area_circle=(int) (3.14%aa);

               System.out.println("The area of circle is:" +area_circle);
}
}
public class JavaApplication3 {

  public static void main(String[] args) {

     rectangle r new rectangle();

     r.print_area():

     triangle t-new triangle():

     t.print_area();

     circle rl-new circle();

     rl.print_area():
     }
}
```

```
package example1;
class MyException extends Exception{
 String str1;
 MyException(String str2) {
str1=str2;
 }
 public String toString(){
return ("MyException Occurred: "+str1) ;
 }
}
public class Example1 {
 public static void main(String[] args)
 {
 try{
System.out.println("Starting of try block");
// I'm throwing the custom exception using throw
throw new MyException("This is My error Message");
}
catch(MyException exp){
System.out.println("Catch Block") ;
System.out.println(exp) ;
}
 }
}
```

```
package com.skillrack.lab;
import java.io.*;
public class FileInfoPrinter {
public static void printDetails (String filePath) throws Exception
{
File f-new File(filePath);
if(f.exists())
{
System.out.println("Present"); BufferedReader br=new BufferedReader(new
FileReader (f)); System.out.println(br.readLine()); br.close();
} else{
System.out.println("NotPresent");}}
```

```
import java.io.*;
import java.lang.*;
import java.util.*;
class even implements Runnable{
public int n;
public even(int n){
 this.n=n;
}
public void run(){
System.out.println("Even Square: "+n*n);
}}
class odd implements Runnable{
public int n;
public odd(int n){
this.n=n;
}
public void run(){
System.out.println("Odd Cube: "+n*n*n);
}}
class MultiThread extends Thread{
public void run(){
int j = 0;
Random r= new Random();
try{
for(int i=0;i<10; i++){
j=r.nextInt(100);
System.out.println("Main Thread No. "+j);
if(j%2==0){
Thread a=new Thread(new even(j));
a.start();
} else{
Thread b=new Thread(new odd(j));
b.start();
}
Thread.sleep(1000);
}
catch(Exception e) {
System.out.println(e.getMessage());
}}}
public class Main{
public static void main(String[] args)
MultiThread m= new MultiThread();
m.start();}}
```

GENERIC FUNCTION PROGRAM:

```
package genericmethodtest;
public class GenericMethodTest {
 public static <T extends Comparable<T>> T maximum(T x, T y, T z)
 {
  T max = x; // assume x is initially the largest
 if(y.compareTo(max) > 0) {
  max = y; // y is the largest so far
 }
 if(z.compareTo(max) > 0) {
  max = z; // z is the largest now
 }
 return max; // returns the largest object
}
public static void main(String args[])
{
 System.out.printf("Max of %d, %d and %d is %d\n\n",
   3, 4, 5, maximum( 3, 4, 5 ));
 System.out.printf("Max of %.1f,%.1f and %.1f is %.1f\n\n",
   6.6, 8.8, 7.7, maximum( 6.6, 8.8, 7.7 ));
 }
}
```

OUTPUT:

```
 RUN:
 Max of 3, 4 and 5 is 5
 Max of 6.6, 8.8 and 7.7 is 8.8
```