-----NEWWWWWWWWW-----

EmployeeInfo

```
import java.util.*;

import java.util.function.*;

import java.util.stream.Stream;



class Employee {

   String name;

   int salary;

   Employee(String name,int salary){

      this.name=name;

      this.salary=salary;

   }

   public void setName(String name){

      this.name=name;

   }

   public String getName(){

      return name;

   }

   public void setSalary(int salary){

      this.salary=salary;

   }

   public int getSalary(){

      return salary;

   }

   @Override

   public String toString() {
        StringBuilder sb = new StringBuilder("<");
        sb.append("name: ");
        sb.append(name);
        sb.append(" salary: ");
        sb.append("" + salary+">");
        return sb.toString();

    }
}
class EmployeeInfo{
    enum SortMethod {
        BYNAME,BYSALARY;
    };
    public List<Employee> sort(List<Employee> emps, final SortMethod method){
        if(method.equals(method.BYNAME)){

            Collections.sort(emps,(e1,e2)->{return e1.name.compareTo(e2.name);
            });
        }
            else if(method.equals(method.BYSALARY)){

              Collections.sort(emps,(e1,e2)->{
              int i =e1.salary - e2.salary;
              if(i==0) {
                  return e1.name.compareTo(e2.name);
              }
              else {
                  return i;
              }
           });
            }
            return emps;
     }
    public boolean isCharacterPresentInAllNames(Collection<Employee> entities, String character){
       Predicate<Employee> p1=s -> s.name.contains(character);
        boolean b1 = entities.stream().allMatch(p1);
        return b1;
    }
}
```

BMI

```
import java.io.*;

import java.lang.*;

import java.util.*;


class BMI {


   public float returnWeight(String str){

      // return the weight part of the string str...


      String[] a=str.split("@");

      String[] b=a[0].split("-");


      String weight1=b[0]+"."+b[1];

      float weight=Float.parseFloat(weight1);


      return weight;

   }

   public float returnHeight(String str){

      // return the height part of the string str...

      String[] a=str.split("@");

      String[] b=a[1].split("-");


      String height1=b[0]+"."+b[1];
```

```
         float height=Float.parseFloat(height1);


         return height;
    }
    public float calculateBMI(float height,float weight){
        // Calculate and return the bmi according to the description given...
        float bmi=0.0f;


        try{
            if(height==0 || weight==0){
                throw new Exception();
            }
            else{
                bmi=weight/(height*height);
                return bmi;
            }
        }catch(Exception e){
            bmi=-1;
            return bmi;
        }
    }
    public String checkStatus(float bmi){
         // return "nourished" or "malnourished" according to the codition
mention in the description...
         if(bmi>=20 && bmi<=24)
             return "nourished";
         else
```

```
            return "malnourished";
    }
}

class Source{
    public static void main(String[] args){
        //Do not Write anything here...
        //Main will be taken care...
    }
}
```

batchNumberCheck

```
import java.util.Scanner;

class batch{

        public boolean lengthCheck(String str)
        {
                int len = str.length();
                if(len==12) {
                        return true;
                }
                return false;
        }

        public boolean batchNumberCheck(String str)
        {
                boolean flag = false;

                String s1 = str.substring(0,4);

                String regex = "^[A-Z]{2}+[0-9]{1}+[A-Z]{1}+$";

                if(s1.matches(regex))
                {
                        flag=true;
```

```
		}
		return flag;

	}

	public boolean yearCheck(String str)
	{
		boolean flag = false;
		String s2 = str.substring(4,8);

		int year = Integer.parseInt(s2);
		if(year>=2015 && year<=2020)
		{
			flag = true;
		}
		return flag;
	}

	public boolean monthCheck(String str)
	{
		String s3 = str.substring(8,10);
		int month = Integer.parseInt(s3);
		boolean flag = false;
		if(month>=1 && month<=12)
		{
			flag = true;
		}
		return flag;
	}
```

```
public boolean dayCheck(String str)
{
        String s4 = str.substring(10,12);

        int day = Integer.parseInt(s4);
        boolean flag = false;
        if(day>=1 && day<=31)
        {
                flag = true;
        }
        return flag;
}

public String printBatchNumber(String str) {

        String s1 = str.substring(0,4);
        String regex = "^[A-Z]{2}+[0-9]{1}+[A-Z]{1}+$";
        String str3 = "";
        if(s1.matches(regex)) {
                str3=s1;
        }
        return str3;
}

public String printDate(String str)
{
```

```
                    String s2 = str.substring(4,8);
                    int year = Integer.parseInt(s2);
                    String s3 = str.substring(8,10);
                    int month = Integer.parseInt(s3);
                    String s4 = str.substring(10,12);
                    int day = Integer.parseInt(s4);

                    if((year>=2015 && year<=2020) && (month>=1 && month<=12) && (day>=1
&& day<=31))
                    {

                              return s2+"/"+s3+"/"+s4;
                    }
                    return null;
          }
}
public class SeedBag {

          public static void main(String[] args) {
                    // TODO Auto-generated method stub
                    Scanner scr = new Scanner(System.in);
                    batch obj = new batch();
                    String str = scr.next();
                    System.out.println(obj.lengthCheck(str));
                    System.out.println(obj.batchNumberCheck(str));
                    System.out.println(obj.yearCheck(str));
                    System.out.println(obj.monthCheck(str));
                    System.out.println(obj.dayCheck(str));
                    System.out.println(obj.printBatchNumber(str));
                    System.out.println(obj.printDate(str));      }   }
```

ScholorshipImp

```
import java.util.ArrayList;

import java.util.HashMap;


public class ScholorshipImpl {


	static HashMap<Integer,Student> hm=new
HashMap<Integer,Student>();

	static {

		hm.put(111, new Student("Alan",111,99));

		hm.put(222, new Student("jennifer",222,100));

		hm.put(333, new Student("Aarya",333,98));

		hm.put(444, new Student("Jen",444,93));

		hm.put(555, new Student("Jack",555,55));

	}

	public void addStudent(Student std) {

		hm.put(std.studentId, std);

	}

	public boolean deleteStudent(int id) {

		if(hm.remove(id)==null)

		{

			return false;

		}
```

```
        else {
        return true;
    }
    }

    public ArrayList<Student> getStudentDetails(String scholorshipScheme){
        ArrayList<Student> result=new ArrayList<Student>();
        for(Student s:hm.values()) {
            if(s.scholorshipScheme.equals(scholorshipScheme)) {
                result.add(s);
            }
        }
        return result;

    }
}
```

scholorshipScheme

```
public class Student {


	String studentName;

	Integer studentId;

	int studentScore;

	String scholorshipScheme;

	public Student(String studentName, Integer studentId, int studentScore) {

		super();

		this.studentName = studentName;

		this.studentId = studentId;

		this.studentScore = studentScore;

		if(studentScore<=90)

			scholorshipScheme="no scheme";

		else if(studentScore>=90 && studentScore<95)

			scholorshipScheme="scheme b";

		else

			scholorshipScheme="scheme a";

	}

	@Override

	public String toString() {

		return "Student{" + ", id=" + studentId + ", score=" + studentScore + ",
scholorshipScheme=" + scholorshipScheme + "}";


	}




}
```

scholorship;

```
import java.util.ArrayList;

import java.util.HashMap;


public class ScholorshipImpl {


        static HashMap<Integer,Student> hm=new
HashMap<Integer,Student>();

        static {

                hm.put(111, new Student("Alan",111,99));

                hm.put(222, new Student("jennifer",222,100));

                hm.put(333, new Student("Aarya",333,98));

                hm.put(444, new Student("Jen",444,93));

                hm.put(555, new Student("Jack",555,55));

        }

        public void addStudent(Student std) {

                hm.put(std.studentId, std);

        }

        public boolean deleteStudent(int id) {

                if(hm.remove(id)==null)

                {

                        return false;

                }

                else {
```

```
			return true;
		}
		}

		public ArrayList<Student> getStudentDetails(String scholorshipScheme){
			ArrayList<Student> result=new ArrayList<Student>();
			for(Student s:hm.values()) {
				if(s.scholorshipScheme.equals(scholorshipScheme)) {
					result.add(s);
				}
			}
			return result;

		}
}
```

Student Info

```
import java.util.ArrayList;

import java.util.List;

import static java.util.stream.Collectors.toList;

import java.util.stream.*;


class Student{


   private String firstName;

   private String lastName;

   private int score;


   public Student(String firstName,String lastName, int score){

      this.firstName=firstName;

      this.lastName=lastName;

      this.score=score;

   }

    public String getFirstName(){

       return firstName;

    }

    public String getLastName(){

       return lastName;

    }

    public long getScore(){

       return score;

    }


    public void setFirstName(){
        this.firstName=firstName;
      }
       public void setLastName(){
        this.lastName=lastName;
      }
       public void setScore(){
        this.score=score;
      }
}


class StudentImplementation{
  public long countStudents(List<Student> list){
      long count=0;
      for(Student i:list){
          if (i.getScore()>70){
              count++;
          }
          else{
          count=count+0;
          }

      }
      return count;
  }
  public List getName(List<Student> list){
      List<String> l=new ArrayList<String>();
      for(Student j: list){
          String s=j.getFirstName()+" "+j.getLastName();
          l.add(s);
      }
      return l;
  }
}
public class Source{
   public static void main(String args[]){
       List<Student> students=new ArrayList<Student>();
        List<String> studentName=new ArrayList<String>();
       students.add(new Student("Deepika","saga",100));
       students.add(new Student("Naga","Deepikasaga",90));
       students.add(new Student("Deepu","saga",98));


       StudentImplementation s=new StudentImplementation();
       long s1=s.countStudents(students);
       studentName=s.getName(students);
        System.out.println(studentName);
       System.out.println(s1);
   }
}
```

scoreCard;

```
public class Student {
  String studentName;
  Integer studentRoll;
  Integer studentScore;
public Student(String studentName, Integer studentRoll, Integer studentScore)
{
        super();
        this.studentName = studentName;
        this.studentRoll = studentRoll;
        this.studentScore = studentScore;
}
public String getStudentName() {
        return studentName;
}
public void setStudentName(String studentName) {
        this.studentName = studentName;
}
public Integer getStudentRoll() {
        return studentRoll;
}
public void setStudentRoll(Integer studentRoll) {
        this.studentRoll = studentRoll;
}
public Integer getStudentScore() {
        return studentScore;
}
public void setStudentScore(Integer studentScore) {
        this.studentScore = studentScore;
}
@Override
public String toString() {
        return "Student{" + ", name=" + studentName + ", roll=" + studentRoll +
", score=" + studentScore
                        + "}";
}
}
```

AgeToDrink

```
public class Implementation {
public String validateIntAgeToDrink(Age obj, int personAge) {
        if(personAge>=21) {
                obj.drinkingAge="legal";
        }
        else if(personAge<21) {
                try {
                        throw new IllegalAgeException("Illegal drinking age");
                }
        catch(IllegalAgeException e) {
                obj.drinkingAge="illegal";
                return "Illegal drinking age";
        }


        }
        return obj.drinkingAge;
}
public String validateStringAgeToDrink(Age obj,String personAge) {
        if(personAge.length()>2 || Integer.parseInt(personAge)<21) {
                try {
                        throw new IllegalAgeException("Invalid age detail or drinking age");
                }
                catch(IllegalAgeException e) {
                        obj.drinkingAge="illegal";
                        return "Invalid age detail or drinking age";
                }
```

```
		}
		else if(personAge.length()<3 && Integer.parseInt(personAge)>=21) {
			obj.drinkingAge="legal";
		}
		return obj.drinkingAge;
	}
	}
```

//String manipulation
public class Source {

```
//String manipulation
public class Source {
        public String concat(String str1, String str2) {
                return str1+str2;

        }
        public int getIndex(String str, char ch) {
                return str.indexOf(ch);
        }
        public String padRight(String str, int len) {
                for(int i=str.length();i<len;i++) {
                        str+=',';
                }
                return str;
        }

        public static void main(String[] args) {
                Source s=new Source();
          System.out.println(s.concat("cat","dog"));
          System.out.println(s.getIndex("cat",'a'));
          System.out.println(s.padRight("cat",5));
```

```
	}
}
```

Age Gender

```
import java.util.Arrays;
import java.util.Collection;
import java.util.Comparator;
import java.util.OptionalInt;

enum Gender
{
	MAN,WOMEN;
}
class People
{
	private String name;
	private Integer age;
	private Gender gender;
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public Integer getAge() {
		return age;
	}
	public void setAge(Integer age) {
		this.age = age;
	}
	public Gender getGender() {
		return gender;
	}
	public void setGender(Gender gender) {
		this.gender = gender;
	}
	public People(String name, Integer age, Gender gender) {
		super();
		this.name = name;
		this.age = age;
		this.gender = gender;
	}
	@Override
	public String toString() {
		return "People [name=" + name + ", age=" + age + ", gender=" + gender + "]";
	}

}
 class PeopleImplementation
{
	public static People  getMinimumAge(Collection<People> collection)
	{
		 People shortest = collection.stream()
				.min(Comparator.comparing(i -> i.getAge()))
				.get();
		return shortest;
	}
	public static Integer  getOldestAge(Collection<People> collection)
	{
		Integer max = collection.stream().mapToInt(b -> b.getAge()).max().getAsInt();
		return max;
	}




}
public class Streams {

	public static void main(String[] args) {
		Collection<People> input=Arrays.asList(
				new People("vivek",12,Gender.MAN),
				new People("kayle",23,Gender.WOMEN),
				new People("Jeremy",23,Gender.WOMEN),
				new People("Ivan",69,Gender.MAN)
				);
		People res1=PeopleImplementation.getMinimumAge(input);
	System.out.println(res1);
	Integer res2=PeopleImplementation.getOldestAge(input);
	System.out.println(res2);

	}




}
```

splitAndReverse

```
import java.util.*;
class Source{
        public int sum(ArrayList<Integer> numbers) {
                int s=numbers.stream().reduce(0, Integer::sum);
                return s;
                }
        public ArrayList<Integer> splitAndReverse(ArrayList<Integer>list){
                ArrayList<Integer> temp=new ArrayList<>();
                int mid=list.size()%2==0 ? list.size()/2 : list.size()/2+1;

                //doing left first
                for(int i =0;i<mid;i++)
                {
                        int value =list.get(i);
                        temp.add(0,value);
                }
                //doing right half

                for(int i=mid;i<list.size();i++)
                {
                        int value = list.get(i);
                        temp.add(mid, value);

                }
                return temp;
```

```
	}

public Integer getitemAtIndex(ArrayList<Integer> list, int index)
{
	if(list.size()<= index)
	{
		return null;
	}
	return list.get(index);
}
public static void main(String[] args) {
	//impl
	//dont
	Source src =new Source();
	ArrayList<Integer> numbers =new ArrayList<>();

	numbers.add(73);
	numbers.add(24);
	//numbers.add(10);
	numbers.add(15);
	numbers.add(5);

	System.out.println(src.sum(numbers));
	System.out.println(src.getitemAtIndex(numbers, 2));
	System.out.println(src.splitAndReverse(numbers));
}
}
```

speed;

```
class Speed{
 String speed;


}
class SpeedImplementation{
 String setSpeed(Speed s,int spd) {
  try {
   if(spd<30||spd>90) {
    throw new SpeedInvalidException("SpeedInvalidException");
   }
   else {
    s.speed = "Valid Speed";
   }
  }
  catch(SpeedInvalidException se) {
   s.speed = "Invalid Speed";
   return se.toString();
  }
  return s.speed;
 }
}
class SpeedInvalidException extends Exception{
 public SpeedInvalidException(String s) {
 }
}
public class Source {
 public static void main(String [] args) {
```

```
 Speed s = new Speed();

 SpeedImplementation si = new SpeedImplementation();

 System.out.println(si.setSpeed(s, 100));

}

}
```

class Contract{

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
import java.lang.*;

class Contract{
	String retailer;
	String customer;

	public Contract(String retailer, String customer) {
		this.retailer = retailer;
		this.customer = customer;
	}



}

class Receipt{
	Contract contract;
```

```
        String productQR;

        public Receipt(Contract contract, String productQR) {

                this.contract = contract;

                this.productQR = productQR;

        }



}


class PrintReceipt{


        public int partyVerification(Receipt r) {

                int count = 0 ;

                Pattern pattern = Pattern.compile("^[A-Za-z][A-Za-z\\'\\-]+([\\ A-
Za-z][A-Za-z\\'\\-]+)*", Pattern.CASE_INSENSITIVE);

           Matcher matcher = pattern.matcher(r.contract.customer);

           if(matcher.matches()) {

                count++;

           }

           matcher = pattern.matcher(r.contract.retailer);

           if(matcher.matches()) {

                count++;

           }


                return count;

        }
```

```
	public String computeGST(Receipt r) {
		String[] str = r.productQR.split("@");
		int result = 0;
		for(int i=0; i< str.length;i++) {
			int rate =
Integer.parseInt(str[i].substring(0,str[i].indexOf(',')));
			int qty =
Integer.parseInt(str[i].substring(str[i].indexOf(',')+1,str[i].length()));
			result = result + rate * qty;

		}
		result = result * 12;

		return Integer.toString(result);
	}
}

public class Source {

	public static void main(String[] args) {



	}


}
```