Contents

Shopping Cart

```
class Product{
    private int id;
    private String name;
    private int quantity;
    private float price;
    Product(int a,String b,int c,float d){
        this.id=a;
        this.name=b;
        this.quantity=c;
        this.price=d;
    }
    public void setId(int a){
        this.id=a;
    }
     public void setName(String a){
        this.name=a;
    }
     public void setQuantity(int a){
         this.quantity=a;
    }
     public void setPrice(float a){
        this.price=a;
    }
    public int getId(){
        return this.id;
    }
    public String getName(){
        return this.name;
    }
    public int getQuantity(){
        return this.quantity;
    }
    public float getPrice(){
        return this.price;
    }
}
class Cart{
    ArrayList<Product> productList=new ArrayList<Product>();
    public int totalItem(){
        int sum=0;
        for(Product i:productList){
            sum+=i.getQuantity();
        }
        return sum;

    }
   public  float netPrice(){
        float sum=0;
        for(Product i:productList){
            sum+=i.getQuantity()*i.getPrice();
        }
        return sum;

    }


}
```

Job Portal

```
class Company{
    String name;
    int requiredCandidates;
    Company(String a,int b){
        this.name=a;
        this.requiredCandidates=b;
    }
}
class JobPortal{
    public String applyJob(Company jobData,String companyName,int num){
        if(!companyName.equals(jobData.name)){
            try{
                throw new CompanyNotFoundException("no such company found");
            }
            catch(Exception e){
                return ""+e;
            }
        }
        else if(jobData.requiredCandidates<num ){
            try{
                throw new NoVacanyFoundException("no vacancy avilable");
            }
            catch(Exception e){
                return ""+e;
            }
        }
        jobData.requiredCandidates-=num;
        return "applied successfully";

    }
}

class CompanyNotFoundException extends Exception{
    public CompanyNotFoundException(String a){
        super(a);
    }
}
class NoVacanyFoundException extends Exception{
    public NoVacanyFoundException(String a){
        super(a);
    }
}
```

Lambda Assignment

```
class Employee{
    String name;
    Integer marks;
    Employee(String a,Integer b){
        this.name=a;
        this.marks=b;
    }
     public void setName(String a){
        this.name=a;
    }
     public String getName(){
        return this.name;
    }
    public void setMarks(Integer a){
         this.marks=a;
    }
     public Integer getMarks(){
         return this.marks;
    }
}
class Processor{
    public static  List<Employee> addEngToName(List<Employee> list){
          List<Employee> res=new ArrayList<Employee>();
        for(Employee e:list){


            res.add(new Employee("Eng"+e.getName(), e.getMarks()));
        }
        return list;
    }

     public static  Long countI(List<Employee> list){
         Long ans=0L;
        for(Employee e:list){

            if(e.getName().contains("i")){
               ans++;
            }
        }
        return ans;
    }
      public static  List<Employee> filterAndMultiply(List<Employee> list){
        List<Employee> res=new ArrayList<Employee>();

         for(Employee e:list){
            if(e.getName().contains("i")){
                res.add(new Employee(e.getName(), e.getMarks()*2));

            }
        }
        return res;
    }


}
```

<u>**LIST OF OPERATIONS**</u>

```
class ArrayListOps {
	public static ArrayList<Integer> makeArrayListInt(int n) {
		int array[]=new int[n];
		for (int i = 0; i < n; i++) {
			array[i]=0;
		}
		ArrayList<Integer>list=new ArrayList<>();
		for(Integer integer:array) {
			list.add(integer);
```

```
                }
                return list;
        }
        public static ArrayList<Integer> reverseList(ArrayList<Integer>list) {
        for(int k=0,j=list.size()-1;k<j;k++){
                list.add(k,list.remove(j));
        }
        return list;
        }
        public static ArrayList<Integer>changeList(ArrayList<Integer> list,int m,int n) {
                int index=list.indexOf(m);
                list.set(index,n);
                return list;
        }
}
public class Source{
        public static void main(String[] args) {
                ArrayListOps.makeArrayListInt(4);
        ArrayList<Integer>list=new ArrayList<Integer>(Arrays.asList(10,25,33,28,10,12));
        ArrayListOps.reverseList(list);
        ArrayListOps.changeList(list,100,10);
        }
}
```

<u>HIRING ON:</u>

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

class Candidate{
   private String name;
        private int id;
        private int age;
        private String gender;
        private String department;
        private int yearOfJoining;
        private double salary;
        public Candidate(int id, String name, int age, String gender, String department, int
yearOfJoining, double salary) {
                super();
                this.name = name;
```

```
		this.id = id;
		this.age = age;
		this.gender = gender;
		this.department = department;
		this.yearOfJoining = yearOfJoining;
		this.salary = salary;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public int getId() {
		return id;
	}
	public void setId(int id) {
		this.id = id;
	}
	public int getAge() {
		return age;
	}
	public void setAge(int age) {
		this.age = age;
	}
	public String getGender() {
		return gender;
	}
	public void setGender(String gender) {
		this.gender = gender;
	}
	public String getDepartment() {
		return department;
	}
	public void setDepartment(String department) {
		this.department = department;
	}
	public int getYearOfJoining() {
		return yearOfJoining;
	}
	public void setYearOfJoining(int yearOfJoining) {
		this.yearOfJoining = yearOfJoining;
	}
	public double getSalary() {
```

```
                        return salary;
            }
            public void setSalary(double salary) {
                        this.salary = salary;
            }
     @Override
    public String toString() {
        return "Employee [id=" + id + ", name=" + name + ", age=" + age + ", gender=" + gender +
", department="
                + department + ", yearOfJoining=" + yearOfJoining + ", salary=" + salary + "]";
    }
}

class Implementation{
  public static Map<String,Long> getCount(List<Candidate>list){
    Map<String,Long>count=new HashMap<>();
    long female=list.stream().filter((gender)->gender.getGender().contains("Female")).count();
    long male=list.stream().filter((gender)->gender.getGender().contains("Male")).count();
    if(female>0)
      count.put("Female",female);
    if(male>0)
      count.put("Male",male);
            return count;
  }
  public static Map<String, Double>getAverageAge(List<Candidate>list){
    Map<String,Double>average=new HashMap<>();
    OptionalDouble
averagefemale=list.stream().filter((gender)->gender.getGender().contains("Female")).mapToDou
ble((age)->age.getAge()).average();
    OptionalDouble
averagemale=list.stream().filter((gender)->gender.getGender().contains("Male")).mapToDouble((
age)->age.getAge()).average();
       if(averagefemale.isPresent())
      average.put("Female",averagefemale.getAsDouble());
       if(averagemale.isPresent())
      average.put("Male",averagemale.getAsDouble());
     return average;
    }
  public static Map<String,Long>countCandidatesDepartmentWise(List<Candidate>list){
    long productdevelop=list.stream().filter((product)->product.getDepartment().contains("Product
Development")).count();
    long s_m=list.stream().filter((sm)->sm.getDepartment().contains("Sales And
Marketing")).count();
```

```
    long s_t=list.stream().filter((st)->st.getDepartment().contains("Security And
Transport")).count();
    long hr=list.stream().filter((st)->st.getDepartment().contains("HR")).count();
    long
infra=list.stream().filter((infras)->infras.getDepartment().contains("Infrastructure")).count();
    long a_f=list.stream().filter((af)->af.getDepartment().contains("Account And
Finance")).count();
    Map<String,Long>count=new HashMap<>();
    if(productdevelop>0)
    count.put("Product Development",productdevelop);
    if(s_m>0)
    count.put("Sales And Marketing",s_m);
    if(s_t>0)
    count.put("Security And Transport",s_m);
    if(hr>0)
    count.put("HR",hr);
    if(infra>0)
    count.put("Infrastructure",infra);
    if(a_f>0)
    count.put("Account And Finance",a_f);
    return count;
  }
  public static Optional<Candidate> getYoungestCandidateDetails(List<Candidate>list){
            Optional<Candidate>candidate=list.stream().filter((male)->

male.getGender().contains("Female")).filter((department)->department.getDepartment().contain
s("Product Development")).min((p1,p2)->p1.getAge()-p2.getAge());
            if(candidate.isPresent()) {
                    candidate.get();
            }
            Optional<Candidate>candidate1=list.stream().filter((male)->
            male.getGender().contains("Male")).filter((department)->department.getDepartment()
.contains("Product Development")).min((p1,p2)->p1.getAge()-p2.getAge());
            if(candidate1.isPresent()) {
                    candidate1.get();
            }
            return candidate1;
  }
}

public class Source {
        public static void main(String args[] ) throws Exception {
                List<Candidate>list=new ArrayList<>();
```

```
			list.add(new Candidate(111,"Damon Salvatore",23,"Male","Product
Development",2009,70000));
			list.add(new Candidate(222,"Elena Gilbert",25,"Female","Product
Development",2012,50000));
			list.add(new Candidate(333,"Stefan Salvatore",30,"Male","Product
Development",2009,60000));
			list.add(new Candidate(444,"Carolyn Forbes",26,"Female","Product
Development",2010,65000));
			Implementation.getCount(list);
   Implementation.getAverageAge(list);
   Implementation.countCandidatesDepartmentWise(list);
   Implementation.getYoungestCandidateDetails(list);
	}
}
```

EMAIL OPERATION

```
class Email{
	Header header;
	String body;
	String greetings;

	public Email(Header header, String body, String greetings) {
		super();
		this.header = header;
		this.body = body;
		this.greetings = greetings;
	}
}
class Header{
	String from;
	String to;
	public Header(String from, String to) {
		super();
		this.from = from;
		this.to = to;
	}

}
class EmailOperations{
public static int emailVerify(Email e) {
		String string = "^([a-zA-Z_]{1}[a-zA-Z]+)@([a-zA-Z]+)\\.([a-zA-Z]{2,30})$";
		int value;
		boolean m1, m2;
```

```
            m1 = Pattern.matches(string, e.header.from);
            m2 = Pattern.matches(string, e.header.to);
            if (m1 && m2 == true)
                      value=2;
            else if (m1 || m2 == true)
                      value=1;
            else
                      value=0;
            //System.out.println(value);
return value;
  }
    public static String bodyEncryption(Email e) {
              StringBuffer result= new StringBuffer();

         for (int i=0; i<e.body.length(); i++)
         {
            if (Character.isUpperCase(e.body.charAt(i)))
            {
               char ch = (char)(((int)e.body.charAt(i) +
                              3 - 65) % 26 + 65);
               result.append(ch);
            }
            else if(Character.isSpace(e.body.charAt(i))) {
                      result.append(e.body.charAt(i));
            }
            else
            {
               char ch = (char)(((int)e.body.charAt(i) +
                              3 - 97) % 26 + 97);
               result.append(ch);
            }
         }
         //System.out.println(result.toString());
         return result.toString();
    }
   public static String greetingMessage(Email e) {
             String string1=e.greetings;
             String string2=e.header.from;
                      int i= string2.indexOf("@");
             StringBuffer sb=new StringBuffer();
                                sb.append(string2);
             StringBuffer sb2=sb.delete(i, sb.length());
             String concat=string1.concat(" ").concat(sb2.toString());
            // System.out.println(concat);
```

```
                    return concat;
            }

}
public class Source {
        public static void main(String args[] ) throws Exception {
                String from = "Jesirupa@gmail.com";
                String to = "jesintha@gmail.com";
                Header e = new Header(from, to);
                String body = "Hi How Are You";
                String greetings = "Regards";
                Email email = new Email(e, body, greetings);
                EmailOperations.emailVerify(email);
                EmailOperations.bodyEncryption(email);
                EmailOperations.greetingMessage(email);
        }
}
```

JOB AGENCY

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

class CompanyJobRepository {
         static String getJobPrediction(int age, String highestQualification) throws
NotEligibleException{
                String string;
                if (age >= 19){
                         if (age >= 21 && highestQualification.equals("B.E"))
                        string = "We have openings for junior developer";
                        else if (age >= 26 &&
(highestQualification.equals("M.S"))||(highestQualification.equals("PhD")))
                        string = "We have openings for senior developer";
                        else if (age >= 19 && !(highestQualification.equals("B.E"))
                                && !(highestQualification.equals("M.S")) &&
!(highestQualification.equals("PhD")))
                        throw new NotEligibleException("We do not have any job that matches
your qualifications");
                        else
                        string = "Sorry we have no openings for now";
                }
                else
                        throw new NotEligibleException("You are underage for any job");
                return string;
        }
}

public class Source {
        public static String searchForJob(int age, String highestQualification) throws
NotEligibleException {
                String string = new String();
                if (age >= 200 || age <= 0) {
                        throw new NotEligibleException("The age entered is not typical for a
human being");
                }
                 else {
                        string= CompanyJobRepository.getJobPrediction(age,
highestQualification);
                }
                return string;
        }
        public static void main(String args[] )  {
        /*try {
                        searchForJob(34, "PhD");
                } catch (NotEligibleException e) {
                        System.out.println(e);
        }*/

        }
}
class NotEligibleException extends Exception {
        public NotEligibleException(String msg) {
                super(msg);
        }
}
```

VALIDATING USER

```
import java.util.*;
import java.lang.*;
import java.util.regex.*;

class TransactionParty {
  String seller;
```

```
        String buyer;
        public TransactionParty(String seller, String buyer) {
                super();
                this.seller = seller;
                this.buyer = buyer;
        }
}

class Receipt{
  TransactionParty transactionParty;
        String productsQR;
        public Receipt(TransactionParty transactionParty, String productsQR) {
                super();
                this.transactionParty = transactionParty;
                this.productsQR = productsQR;
        }
}


class GenerateReceipt{
   public static int verifyParty(Receipt r) {
                String regex= "[A-Za-z]{1}[A-Za-z\\'\\s]+|[A-Za-z\\s-]+[A-Za-z]{1}";
                int value;
                boolean m1,m2;
                m1=Pattern.matches(regex, r.transactionParty.seller);
                m2=Pattern.matches(regex, r.transactionParty.buyer);
                if(m1&&m2==true)
                        value=2;
                else if(m1||m2==true)
                        value=1;
                else
                        value=0;
                return value;
        }
        public static String calcGST(Receipt r) {
                int gst=0; float gst_rate=0.12F;
           String[]pairs=r.productsQR.split("@");
                   for(String pair:pairs) {
                String[] rateQty=pair.split(",");
                String rate=rateQty[0];
                String quantity=rateQty[1];
                int total=(Integer.parseInt(rate))*(Integer.parseInt(quantity));
                gst=gst+total;
           }
```

```
            gst=(int)(gst*gst_rate);
               return Integer.toString(gst);
                }
}
class Source{
  public static void main(String[] args){
   /* String seller= "Jesintha";
                        String buyer= "Roopavathi";
                        TransactionParty tp=new TransactionParty(seller,buyer);
              String productQR="250,10@100,3@50,7";
                        Receipt receipt=new Receipt(tp,productQR);
                        GenerateReceipt.verifyParty(receipt);
                        GenerateReceipt.calcGST(receipt);*/
  }
}
```

HANDLING STUFF

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
 class Activity{
            String string1;
            String string2;
            String operator;
            public Activity(String string1, String string2, String operator) {
                        this.string1 = string1;
                        this.string2 = string2;
                        this.operator = operator;
  }
}

 public class Source {
  public String handleException(Activity a) {
                        String string;
            try {
            if (a.string1.equals(null) || a.string2.equals(null))
            throw new NullPointerException("Null values found");

            if (!(a.operator.equals("+")) && !(a.operator.equals("-")))
            throw new Exception("Default exception"+a.operator);
            }
```

```
        catch (NullPointerException ex) {
        string= "Null values found";
        //System.out.println(string);
        return string;
        }
        catch (Exception e) {
        string= "Default Exception"+a.operator;
        //System.out.println(string);
        return string;
        }
        return "No Exception Found";
}
        public String doOperation(Activity a){
                //String string = a.operator;
                String result= new String();
        switch (a.operator) {
        case "+":{result=a.string1.concat(a.string2);
                                //System.out.println(result);
                                return result;
                        }
        case "-":{      String regex=a.string2;
                                result=a.string1.replaceAll(regex, "");
                                //System.out.println(result);
                        return result;
                }
        }
        return result ;

}
  /*public static void main(String args[] ) throws Exception {
        Source source=new Source();
        Activity activity = new Activity("Helloworld", "world", "+");
                source.handleException(activity);
                source.doOperation(activity);
        }*/
}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////
MOBILE SHOP
//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
///////
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

class Mobile{
  // Write your code here..
        HashMap<String,ArrayList<String>> mobiles=new
HashMap<String,ArrayList<String>>();
    public String addMobile(String company,String model){
                ArrayList<String> list=new ArrayList<String>();
                if (mobiles.containsKey(company)==false){
                        list.add(model);
                        mobiles.put(company,list);
                return "model successfully added";
                }
                else{
                        list=mobiles.get(company);
                        list.add(model);
                        mobiles.put(company,list);
                        return "model successfully added";
                }
         }

        public ArrayList<String> getModels(String company){
                ArrayList<String> list1=new ArrayList<String>();
      if((mobiles.containsKey(company)==false)|(mobiles.get(company)==null)){
                return null;
                }
                else{
                list1=mobiles.get(company);
                return list1;
                }
}
public String buyMobile(String company,String model){

        ArrayList<String> list1=new ArrayList<String>();
        list1=mobiles.get(company);

if(mobiles.containsKey(company)==true&&list1.contains(model)==true){
int j=0;
if(list1.contains(model)){
j=list1.indexOf(model);
list1.remove(j);
}
mobiles.put(company,list1);
return "mobile sold successfully";
}
else{
return "item not available";
}
}
}

public class Source {
        public static void main(String args[] ) throws Exception {
                /* Enter your code here. Read input from STDIN. Print output to STDOUT */
        }
}
```

EMPLOYEE VERIFICATION

```
import java.util.*;
import java.util.function.*;
import java.util.stream.Stream;
import java.util.stream.Collectors;
class Employee {
   String name;
   int salary;
public Employee(String name,int salary){
   this.name = name;
   this.salary = salary;
}
public String getName(){
   return name;
}
public void setName(String name){
   this.name= name;
}
public int getSalary(){
   return salary;
}
public void setSalary(int salary){
   this.salary = 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){
        Comparator<Employee> comparator;
        if(method == SortMethod.BYNAME) {
                comparator = Comparator.comparing(Employee::getName);
      // System.out.println(comparator);
       }
        else {
                  comparator = Comparator.comparing(Employee::getSalary);
                  //System.out.println(comparator);
        }


            return emps.stream().sorted(comparator).collect(Collectors.toList());
 }
public boolean isCharacterPresentInAllNames(Collection<Employee> entities,String character){
   // int count=0;
    long cnt = entities.stream().filter(x -> x.name.startsWith(character)).count();
    if(cnt==1 )
        return true;
    else
        return false;
 }
/*public static void main(String[] args) {
                    // TODO Auto-generated method stub
                    List<Employee> emps = new ArrayList<>();
emps.add(new Employee("Mickey", 100000));
emps.add(new Employee("Timmy", 50000));
emps.add(new Employee("Annny", 40000));
                    EmployeeInfo EI = new EmployeeInfo();
                 EI.sort(empList,EmployeeInfo.SortMethod.BYSALARY);
                    boolean result = EI.isCharacterPresentInAllNames(empList, "K");

                    //System.out.println(empList);
            /*if(result == true)
                    System.out.println("Present");
            else
                    System.out.println("Missing");

        }*/
}
```

Question Name - Score Card

Class definitions:

Class Student:

stuName : String
stuRoll : Int
stuScore : Int

Create parameterized Constructor and getters and setters

Class Implementation:

public List<Student> sortByScore(List<Student> stu){

}
public long getScoreCountAbove35(List<Student> stu){

}

```
import java.util.*;
import java.util.stream.Collectors;

class Student{
   String stuName;
   int roll;
   int score;

   @Override
   public String toString() {
      return "Student{" +
            "stuName='" + stuName + '\'' +
            ", roll=" + roll +
            ", score=" + score +
            '}';
   }
```

```
    public Student(String stuName, int roll, int score) {
        this.stuName = stuName;
        this.roll = roll;
        this.score = score;
    }

    public String getStuName() {
        return stuName;
    }

    public void setStuName(String stuName) {
        this.stuName = stuName;
    }

    public int getRoll() {
        return roll;
    }

    public void setRoll(int roll) {
        this.roll = roll;
    }

    public int getScore() {
        return score;
    }

    public void setScore(int score) {
        this.score = score;
    }
}

class Implementation{

    public List<Student> sortByScore(List<Student> stu){

        Comparator<Student> comparator;
        comparator = Comparator.comparing(Student::getScore);
        return stu.stream().sorted(comparator).collect(Collectors.toList());
    }

    public long getScoreCountAbove35(List<Student> stu){
        return stu.stream().filter(a->a.getScore()>35).count();
    }
```

```
}

public class Source {
    public static void main(String[] args) {

        List<Student> l = new ArrayList<>();

        l.add(new Student("yokesh",101,80));
        l.add(new Student("vivek",102,30));

        Implementation i = new Implementation();
        System.out.println(i.sortByScore(l));
        System.out.println(i.getScoreCountAbove35(l));


    }
}
```

Question Name - Job Repository

(You can find this question in our practice test (4th question)) Nost similar one but requires little changes in actual exam.

```
-------------------------------------------

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

class CompanyJobRepository {

            static String getJobPrediction(int age, String highestQualification) throws
NotEligibleException{

                        String s;

                        if (age >= 19){

                                    if (age >= 21 && highestQualification.equals("B.E")){
                                                s = "We have openings for junior developer";
                                    }
```

```
                else if (age >= 26 &&( highestQualification.equals("M.S")) ||
(highestQualification.equals("PhD")))
                {
                        s = "We have openings for senior developer";
                }

                else if (age >= 19 && !(highestQualification.equals("B.E"))
                        && !(highestQualification.equals("M.S")) &&
!(highestQualification.equals("PhD")))
                        {
                                throw new NotEligibleException("We do not have any job
that matches your qualifications");
                        }

                else
                {
                        s = "Sorry we have no openings for now";
                }

        }
        else
        {
                throw new NotEligibleException("You are underage for any job");
        }
        return s;
    }
}

public class Source {
    public static String searchForJob(int age, String highestQualification) throws
NotEligibleException {

        String s = new String();

        if (age >= 200 || age <= 0) {
                throw new NotEligibleException("The age entered is not typical for a
human being");
        }
         else {
                s= CompanyJobRepository.getJobPrediction(age, highestQualification);
        }
        return s;
    }
    public static void main(String args[] )  {
        }
}
class NotEligibleException extends Exception {
        public NotEligibleException(String error) {
                super(error);
        }
}
```

Qn-List of product

```
class Product{
   List<String> productList = new ArrayList<String>();
   public void addProduct(String pName){
      productList.add(pName);
   }
   public void removeProduct(String pName){
      productList.remove(pName);
   }
   public int uniqueProduct(){
      HashSet<String> hset = new HashSet<String>(productList);
      return hset.size();
   }
}

public class Source {
   public static void main(String[] args) {
      Product p1=new Product();
      p1.addProduct("Pen");
      p1.addProduct("Shirt");
      p1.removeProduct("Shirt");
      p1.addProduct("Pen");
      int count = p1.uniqueProduct();
      System.out.println(count);
   }
}
```

Unlock with pin

```
class GetCode{
   public int getCodeThroughStrings(String s){
      s = s.replaceAll("\\s","");
      int len = s.length();
```

```
        int result=0;
        while (len > 0 || result >= 10) {
            if (len == 0) {
                len = result;
                result = 0;
            }
            result += len % 10;
            len /= 10;
        }
        return  result;
    }
}

public class Exp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        GetCode gc = new GetCode();
        int len = gc.getCodeThroughStrings(s);
        System.out.println(len);
    }
```

Qn : List of operations

```
class ArrayListOps {
            public static ArrayList<Integer> makeArrayListInt(int n) {
                        int array[]=new int[n];
                        for (int i = 0; i < n; i++) {
                                    array[i]=0;
                        }
                        ArrayList<Integer>list=new ArrayList<>();
                        for(Integer integer:array) {
                                    list.add(integer);
                        }
                        return list;
            }
            public static ArrayList<Integer> reverseList(ArrayList<Integer>list) {
            for(int k=0,j=list.size()-1;k<j;k++){
                        list.add(k,list.remove(j));
}
            return list;
            }
            public static ArrayList<Integer>changeList(ArrayList<Integer> list,int m,int n) {
                        int index=list.indexOf(m);
```

```
                    list.set(index,n);
                    return list;
          }
}
public class Source{
          public static void main(String[] args) {
                    ArrayListOps.makeArrayListInt(4);
          ArrayList<Integer>list=new ArrayList<Integer>(Arrays.asList(10,25,33,28,10,12));
          ArrayListOps.reverseList(list);
          ArrayListOps.changeList(list,100,10);
          }
}

package com.vrnu.modelcode; // All classes are inherited by default from Object
public class Employee {
 int employeeId ;
 String employeeName ;
 String Status;
public Employee(int employeeId, String employeeName, String status) {
 super(); this.employeeId = employeeId;
this.employeeName = employeeName;
 Status = status;
 }
 public int getEmployeeId() {
 return employeeId;
 }
 public void setEmployeeId(int employeeId) {
 this.employeeId = employeeId;
}
 public String getEmployeeName() {
 return employeeName;
 }
 public void setEmployeeName(String employeeName) {
 this.employeeName = employeeName;
 }
 public String getStatus() {
 return Status;
 }
 public void setStatus(String status) {
 Status = status;
 }
 @Override
 public String toString() {
```

```
 return "Employee [employeeId=" + employeeId + ", employeeName=" + employeeName + ",
Status=" + Status + "]";
 }
 }
```

Qs Name:- Exception …. //Please go through question & do//

```
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;
class Activity{
  //Implement Activity class here..
  String string1;
  String string2;
  String operator;

  public String setString1(String string1)
  {
     this.string1 = string1;
     return string1;
  }
  public String setString2(String string2)
  {
     this.string2 = string2;
     return string2;
  }
  public String getString1(String string1)
  {
    return string1;
  }
   public String getString2(String string2)
  {
    return string2;
  }
  public Activity(String string1, String string2 , String operator)
  {
    super();
    this.string1 = string1;
    this.string2 = string2;
    this.operator = operator;

  }
  @Override
  public String toString()
  {
    return string1+" "+string2+" "+operator;
  }
}



public class Source {
  //Implement the two function given in description in here...
  public String handleException(Activity a)throws Exception
  {
    String Null = null;
    try {
            if(a.string1==Null || a.string2==Null)
        {
          throw new NullPointerException("Null values found");
        }
        else if(a.operator != "+" && a.operator != "-")
        {
          throw new Exception(a.operator);
        }
        else
        {
          System.out.println("No Exception Found");

        }

            }
    catch (NullPointerException npe) {
                        // TODO: handle exception
            System.out.println(npe);
            }

    catch (Exception e) {
                        // TODO: handle exception
            System.out.println(e);
            }
    return Null;
  }

  public String  doOperation(Activity a)
  {
    String str = null;
    switch(a.operator)
    {
      case "+":
        String s1 = a.string1.concat(a.string2);
//            str = a.string1.concat(a.string2);
        System.out.println(s1);
        break;
      case "-":
        str = a.string1.replace(a.string1,a.string2);
        System.out.println(str);
        break;
    }
    return str;
  }

        public static void main(String args[] ) throws Exception {
    //Write your own main to check the program...
    Activity ac = new Activity("mondal", "priya" ,"=");
    Source so = new Source();
    String s = so.handleException(ac);
    String res = so.doOperation(ac);
    System.out.println(s);
        }
}
```

<u>**Check your Car Speed**</u>

```
import java.util.stream.Collectors;
import java.util.stream.Stream;
import java.util.*;
 class Brand {
String model;
int speed;
public Brand(String model,int speed) {
this.model=model;
 this.speed=speed;
 }
public String getModel() {
 return model;
}
public void setModel(String model) {
 this.model = model;
}
public int getSpeed() {
 return speed;
 }
public void setSpeed(int speed) {
this.speed = speed;
 }
 @Override
 public String toString() {
 return "Brand{" + "model=" + model + ", speed=" + speed + '}';
 } }
 class BrandImplementation {
 public ArrayList<Brand> Al = new ArrayList<Brand>(2);
 public long getCount(List<Brand> list){
 List<Brand> LR=list.stream().filter(b->b.getSpeed()>200).collect(Collectors.toList());
 long sum=LR.stream().mapToLong(b->b.getSpeed()).count(); return sum;
}
 public List<Brand> sortBySpeed(List<Brand> list){
 List L=list.stream().map(b -> b.getSpeed()).sorted().collect(Collectors.toList());
 return L;
 } }
 public class Source{
 public static void main(String[] args)
{
 BrandImplementation m = new BrandImplementation();
 m.Al.add(new Brand("SUV",500));
 m.Al.add(new Brand("SEDAN",800));
 System.out.println("Sorted Order:" + m.sortBySpeed(m.Al));
System.out.println("Count:" + m.getCount(m.Al));
 } }
```

Temperature

```
 class Temperature {
    double celsius;
    double fahrenheit;
    Temperature(double a,double b){
        this.celsius=a;
        this.fahrenheit=b;
    }
}
class Validator{
    public String validConversion(Temperature t) {
        Validator v=new Validator();
        double tf=v.celciusToFahrenheit(t.celsius);
        if(tf!=t.fahrenheit){
             try{
                throw new InvalidConversionException("Invalid Conversion");
             }
            catch(InvalidConversionException e){
                return ""+e;
             }
        }
        return "Valid Conversion";

    }
    public double celciusToFahrenheit(double celcius){
        return (celcius*1.8)+32;

    }
}
class InvalidConversionException extends Exception{
    public InvalidConversionException(String a){
        super(a);
    }
}
```

Binging and Streaming

```
class Product{
    private int id;
    private String name;
    private float price;
    Product(int a,String b,float c){
        this.id=a;
        this.name=b;
        this.price=c;
    }
    public void setId(int a){
        this.id=a;
    }
     public void setName(String a){
        this.name=a;
    }
     public void setPrice(float a){
        this.price=a;
    }
    public int getId(){
        return this.id;
    }
    public String getName(){
        return this.name;
    }
    public float getPrice(){
        return this.price;
    }
}
class ProductImplementation{
    public double sumOfPrices(List<Product> list){
        double sum=0;
        for(Product i:list){
            sum+=i.getPrice();
        }
        return sum;

    }
   public  float maxPrice(List<Product> list){
        float sum=0;
        for(Product i:list){
            sum=Math.max(sum,i.getPrice());
        }
        return sum;

    }
    public List<String> getProductNameList(List<Product> list){
        List<String> ans=new ArrayList<String>();
         for(Product i:list){
          if(i.getPrice()>25000){
              ans.add(i.getName());
          }
        }
        return ans;

    }

}
```

BMI Calculator

```
public float getWeight(String s){
        String[] k=s.split("\\@");
        k[0]=k[0].replaceAll("-",".");
        return Float.valueOf(k[0]);
    }
    public float getHeight(String s){
         String[] k=s.split("\\@");
        k[1]=k[1].replaceAll("-",".");
        return Float.valueOf(k[1]);
    }
```

Company Salary System

```
package com.vrnu.modelcode;
 public class ExceptionCheck {
String validateEmployee(Employee Emp) throws InvalidEmployeeException {
String st = "";
 try {
if( Emp.getEmployeeId() == 0 || Emp.getEmployeeId()<100) {
 st = "Failure"; throw new InvalidEmployeeException("Invalid Employee Id");
 }
 else if( Emp.getEmployeeName() == null || Emp.getEmployeeName().length()< 3)
{
st = "Failure";
 throw new InvalidEmployeeException("Invalid Employee Name");
}
else
 {
 st = "Success";
} }
 catch(InvalidEmployeeException iEE) {
 System.out.println(iEE.getMessage());
}
return st;
 }
 public static void main(String ar[]) throws InvalidEmployeeException
{
 Employee E1 = new Employee(100, "Muskan", null);
 Employee E2 = new Employee(101, "Mu", null);
 Employee E3 = new Employee(10, "Nalla", null);
 ExceptionCheck Obj = new ExceptionCheck();
 String S1 = Obj.validateEmployee(E1);
 E1.setStatus(S1);
 // Calling the Setter
 System.out.println(E1);
 // Object is calling toString()
 String S2 = Obj.validateEmployee(E2);
 E2.setStatus(S2);
 // Calling the Setter
 System.out.println(E2);
 // Object is calling toString()
 String S3 = Obj.validateEmployee(E3);
 E3.setStatus(S3);
 // Calling the Setter
System.out.println(E3);
 // Object is calling toString()
 } }
```

Telecom Repository

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

import static practice.TelecomRepository.Consumer.getCountry;

public class TelecomRepository {
    static String getCountryName(String code) throws InvalidCodeException {

        if (code == "90" || code == "91" || code == "92" || code == "93" || code == "94" || code ==
"95" || code == "96" || code == "97" || code == "98" || code == "99" || code == "100") {
            return "US";
        } if (code == "101") {
            return "Canada";
        } else {
            throw new InvalidCodeException("No Country with given code");


        }
    }


static class Consumer{
     public static String getCountry(String code) throws InvalidCodeException{
        String cd =code;
         if(cd.length()>3 || cd.length()<2){
            throw new InvalidCodeException("Invalid Code");
        }else {
            return getCountryName(cd);
        }
    }

}

static class InvalidCodeException extends Exception{
    public  InvalidCodeException(String message){
        super(message);
    }
}
 static class Main2{
    public static void main(String[] args) {
        try {

            System.out.println(getCountry("99"));//Output = US
            System.out.println(getCountry("101"));// Output = Canada
            System.out.println(getCountry("103"));//No Country with given code



        }catch (InvalidCodeException e){
            System.out.println(e.getMessage());
        }
    }

    }
}
```

INR Dollar

```
package practice;

import java.util.HashMap;
import java.util.Map;

public class Currency {
    public HashMap<String,String> currencyMap = new HashMap<>();
    public Currency(){
    }

    @Override
    public String toString() {
        return "Currency{" +
                "currencyMap=" + currencyMap +
                '}';
    }

    HashMap<String , String> addCountryCurrency(String country, String currency){

            currencyMap.put(country,currency);


        return currencyMap;
    }

    String getCurrency(String country){
        String s = null;
        if(currencyMap.containsKey(country)){
            s = currencyMap.get(country).toString();
        }
        return s;

    }

    String getCountry(String currency){
        String s1 = null;
        String d = getCurrency(currency);

            for (Map.Entry<String,String> entry: currencyMap.entrySet()
                 ) {
                if(entry.getValue()==currency){
                    String key = entry.getKey();
                    s1 = key;
                }
                }




        return s1;

    }
    String swapKeyValue(){
        HashMap<String,String> cMap = new HashMap<>();
        for (Map.Entry<String,String> s:currencyMap.entrySet()
             ) {
            cMap.put(s.getValue(),s.getKey());
        }
        return cMap.toString();
    }
}

class Source3{
    public static void main(String[] args) {
        Currency currency = new Currency();
        currency.addCountryCurrency("Argentina","Peso");
        currency.addCountryCurrency("Brazil", "Real");
        currency.addCountryCurrency("Cuba","Cuban Peso");
        System.out.println(currency);
        System.out.println(currency.getCurrency("Brazil"));
        System.out.println(currency.getCountry("Peso"));
        System.out.println(currency.swapKeyValue());
    }
}
```