**List of Operations:**

**import** java.util.\*; **class** ArrayListOps {

**public** ArrayList<Integer> makeArrayListInt(**int** n)

{

ArrayList<Integer> list = **new** ArrayList<>();

**for**(**int** i =0 ; i<n;i++)

{

list.add(0);

}

**return** list;

}

**public** ArrayList<Integer> reverseList(ArrayList<Integer> list)

{

Collections.*reverse*(list); **return** list;

}

**public** ArrayList<Integer> changeList(ArrayList<Integer> list, **int** m, **int** n)

{

ArrayList<Integer> list1 = **new** ArrayList<>();

**for**(Integer i : list)

{

**if**(i==m) list1.add(n);

**else**

list1.add(i);

}

**return** list1;

}

}

**public** **class** Source{

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

ArrayList<Integer> list = **new** ArrayList<Integer>(Arrays.*asList*(10, 25, 33, 28, 10, 12));

ArrayListOps obj = **new** ArrayListOps();

ArrayList<Integer> list1 =obj.makeArrayListInt(4); ArrayList<Integer> list2 =obj.reverseList(list);

ArrayList<Integer> list3 =obj.changeList(list2,28,20);

System.***out***.println(list1);

System.***out***.println(list2);

System.***out***.println(list3);

}

}

**Mobile Shop:**

**class** Mobile{

// Write your code here..

HashMap<String, ArrayList<String>> mobiles = **new** HashMap<>();

**public** String addMobile(String company, String model)

{

ArrayList<String> list=**null**;

**if**(mobiles.containsKey(company))

{

list=mobiles.get(company);

}

**else**

{

list =**new** ArrayList<String>();

}

list.add(model);

mobiles.put(company,list);

**return** "model successfully added";

}

**public** ArrayList<String> getModels(String company)

{

**if**(mobiles.containsKey(company))

{

**return** mobiles.get(company);

}

**else**

{

**return** **null**;

}

}

**public** String buyMobile(String company, String model)

{

**if**(mobiles.containsKey(company))

{

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

**boolean** flag = **false**; **for**(String s : list )

{

**if**(s.equals(model))

{

list.remove(s);

flag =**true**; **break**;

}

}

mobiles.put(company,list);

**if**(flag)

**return** "mobile sold successfully";

**else**

**return** "item not available";

}

**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 \*/

Mobile obj = **new** Mobile();

System.***out***.println(obj.addMobile("Oppo", "K3"));

System.***out***.println(obj.getModels("Oppo"));

System.***out***.println(obj.buyMobile("Oppo", "K3"));

}

}

**Email Operation:**

**class** Email{

// Implement Email Class according to the specifiaction.

Header header;

String body;

String greetings;

Email(Header header,String body,String greetings)

{

**this**.header=header; **this**.body=body;

**this**.greetings=greetings;

}

}

**class** Header{

// Implemet the Header Class according to the specifiaction.

String from;

String to;

Header(String from,String to)

{

**this**.from=from; **this**.to=to;

}

}

**class** EmailOperations{

// Implemet the Three methods specified in the specified.

**public** **int** emailVerify(Email e)

{

String s1=e.header.from; String s2=e.header.to;

**int** i=0;

**if**(s1.matches("[a-zA-Z\_]\*[@]{1}[a-z]\*[/.][a-

z]\*")&&s2.matches("[a-zA-Z\_]\*[@]{1}[a-z]\*[/.][a-z]\*"))

i=2;

**else** **if**(s1.matches("[a-zA-Z\_]\*[@]{1}[a-z]\*[/.][a-

z]\*")||s2.matches("[a-zA-Z\_]\*[@]{1}[a-z]\*[/.][a-z]\*"))

i=1;

**return** i;

}

**public** String bodyEncryption(Email e)

{

String s=e.body; String s1="";

**for**(**int** i=0;i<s.length();i++)

{

**if**(s.charAt(i)==' ')

s1+=" ";

**else**

{

**int** k=s.charAt(i);

**if**((k>=88&&k<=90)||(k>=120&&k<=122))

k-=26; k+=3;

s1+=(**char**)k;

}

}

**return** s1;

}

**public** String greetingMessage(Email e)

{

String s1=e.greetings; String s2=e.header.from; **int** k = s2.indexOf("@"); String s3=s2.substring(0,k);

**return** s1+" "+s3;

}

}

**Handling Stuff:**

**class** Activity{

//Implement Activity class here..

String string1;

String string2;

String operator;

Activity(String string1,String string2,String operator)

{

**this**.string1=string1; **this**.string2=string2; **this**.operator=operator;

}

}

**class** MyException **extends** Exception

{

MyException(String msg)

{ **super**(msg);

}

}

**public** **class** Source {

//Implement the two function given in description in here... **public** String handleException(Activity a)

{

String s1=a.string1;

String s2=a.string2;

String s3=a.operator;

**try**

{

**if**(s1==**null**||s2==**null**)

**throw** **new** NullPointerException("Null values found"); **else** **if**(s3!="+"&&s3!="-") **throw** **new** MyException(s3);

}

**catch**(NullPointerException e)

{

**return** e.getMessage();

}

**catch**(MyException ex)

{

**return** ex.getMessage();

}

**return** "No Exception Found";

}

**public** String doOperation(Activity a)

{

String s1=a.string1;

String s2=a.string2;

String s3=a.operator;

String s = "";

**switch**(s3) {

**case** "+":s=s1+s2; **break**; **case** "-":s=s1.replace(s2,""); **break**;

} **return** s;

} **public** **static** **void** main(String args[] ) **throws** Exception { //Write your own main to check the program...

}

}

**Job Agency:**

**class** CompanyJobRepository {

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

{

**if**(age<19)

**throw** **new** NotEligibleException("You are underage for any job"); **else** **if**(age>=21&&highestQualification.equals("B.E"))

{

**return** "We have openings for junior developer";

}

**else**

**if**(age>=26&&(highestQualification.equals("M.S")||highestQualification.equals("PhD" )))

{

**return** "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");

}

**return** "Sorry we have no openings for now";

}

}

**public** **class** Source {

String searchForJob(**int** age, String highestQualification) **throws**

NotEligibleException

{

String s= "";

**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[] ) {

/\* Enter your code here. Read input from STDIN. Print output to STDOUT \*/

}

}

**class** NotEligibleException **extends** Exception {

NotEligibleException(String msg)

{

**super**(msg);

}

}

**Validating Users:**

**class** TransactionParty {

String seller;

String buyer;

**public** TransactionParty(String seller, String buyer)

{

**this**.seller=seller; **this**.buyer=buyer;

}

}

**class** Receipt{

TransactionParty transactionParty;

String productsQR;

**public** Receipt(TransactionParty transactionParty, String productsQR)

{

**this**.transactionParty=transactionParty; **this**.productsQR=productsQR;

}

}

**class** GenerateReceipt{

**public** **int** verifyParty(Receipt r)

{

String s1=r.transactionParty.seller; String s2=r.transactionParty.buyer; **int** i=0;

**if**(s1.matches("^[a-zA-Z][a-zA-Z' -]\*[a-zA-Z]$")&&s2.matches("^[a-zA-Z][a-zA-Z'

-]\*[a-zA-Z]$")) i=2;

**else** **if**(s1.matches("^[a-zA-Z][a-zA-Z' -]\*[a-zA-Z]$")||s2.matches("^[a-zA-Z][azA-Z' -]\*[a-zA-Z]$")) i=1;

**return** i;

}

**public** String calcGST(Receipt r)

{

String s=r.productsQR;

String s1[] = s.split("@"); **float** sum=0; **for**(String i:s1)

{

String[] s2=i.split(",");

**int** k=1;

**for**(String j:s2)

{

k\*=Integer.*parseInt*(j);

}

sum=sum+k;

}

sum=(sum/100)\*12;

String z=String.*format*("%d",(**int**)sum);

**return** z;

}

} **class** Source{

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

} }

**BMI Calculator:**

**class** BMICalculator{

//Implement the methods here.. **public** **float** getWeight (String str){

String[] temp = str.split("\\&");

String rel= temp[0].replace("/",".");

**float** res1 = Float.*valueOf*(rel);

**return** res1;

}

**public** **float** getHeight (String str){

String[] temp1 = str.split("\\&");

**float** res2 = Float.*valueOf*(res2);

**return** res2;

}

}

----------------------------------------------------------------------------------

**class** BMICalculator{

// Implement the methods here..

**float** getWeight(String str) {

**float** res;

|  |
| --- |
| String  String |

strl = str.replaceAll("/", ".");

[] sepStr = str1.split("&");

**try** {

res = Float.*parseFloat*(sepStr[0]);

}

**catch**(NumberFormatException|ArrayIndexOutOfBoundsException e)

{

**throw** e;

}

**return** res;

}

**float** getHeight (String str) {

**float** res;

|  |
| --- |
| String  String |

str1=str.replaceAll("/",">");

[] sepStr= str1.split("&");

**try**{

res= Float.*parseFloat*(sepStr[1]);

}

**catch**(NumberFormatException | ArrayIndexOutOfBoundsException e){

**throw** e;

}

**return** res;

}

}

**Hiring Challenge:**

**class** Candidate {

String name; **int** totalRating;

**int** totalContest;

**public** Candidate(String name, **int** totalRating, **int** totalContest) { **super**();

**this**.name = name; **this**.totalRating = totalRating; **this**.totalContest = totalContest;

}

}

**class** Validator {

**public** String eligible(Candidate details) **throws** Exception {

**try** {

**if** (details.totalRating < 1000) {

**throw** **new** CriteriaMismatchException("minimum 1000 total rating is required");

}

**if** (details.totalContest < 10) {

**throw** **new** CriteriaMismatchException("minimum 10 contest participation is required");

}

**return** "eligible candidate";

} **catch** (CriteriaMismatchException c) { **return** c.getMessage();

}

}

**public** String sendInvite(Candidate deatils) **throws** Exception {

**try** {

eligible(deatils); **return** "invitation send";

} **catch** (CriteriaMismatchException c) { **return** "candidate is not eligible";

} **catch** (Exception e) {

**return** "other exception";

}

}

} **class** CriteriaMismatchException **extends** Exception {

**public** CriteriaMismatchException(String msg) {

**super**(msg);

}

}

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

Candidate cd = **new** Candidate("steve", 1020, 20);

Validator v = **new** Validator();

System.***out***.println(v.eligible(cd));

System.***out***.println(v.sendInvite(cd));

}

}

**Bentley Car:**

**class** Vehicle {

// Implement the Vehicle with constructor and getter setter method

String name;

Double price;

Vehicle (String name, Double price) { **this**.name = name; **this**.price = price;

}

**public** **void** setName(String name) { **this**.name = name;

}

**public** **void** setPrice (Double price) { **this**.price = price;

}

**public** String getName() { **return** **this**.name;

}

**public** Double getPrice() { **return** **this**.price;

}

**class** VehicleImplementation {

// Implement the VehicleImplementation method... **public** Double averageCost (List<Vehicle> list){

Double avg=list.stream().mapToDouble(x-

>x.getPrice()).average().getAsDouble();

**return** avg;

}

**public** List<String> getVehicleList (List<Vehicle> list){

**return** list.stream().filter (x->x.price(750008).map(x-

>x.name).collect(Collectors.toList()));

}

**public** **double** minPrice (List<Vehicle> list) {

Double sum= list.stream().mapToDouble(x-

>x.price).min().orElse(0.0);

**return** sum;

}

}

**class** Source{

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

List<Vehicle> list = **new** ArrayList<Vehicle>(); list.add(**new** Vehicle ("Alfa Romeo", 768008d)); list.add(**new** Vehicle ("Bugatti", 95000d));

}

}

}

**CVV Validation:**

**class** GFG {

**public** **static** **boolean** isValidCVVNumber(String str)

{

String regex = "^[0-9]{2,3}$"; Pattern p = Pattern.*compile*(regex); **if** (str == **null**)

{

**return** **false**;

}

Matcher m = p.matcher(str); **return** m.matches();

}

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

{

String str1 = "561";

System.***out***.println(*isValidCVVNumber*(str1)); String str2 = "5061";

System.***out***.println(*isValidCVVNumber*(str2)); String str3 = "50614";

System.***out***.println(*isValidCVVNumber*(str3)); String str4 = "5a#1";

System.***out***.println(*isValidCVVNumber*(str4));

} }

**DNA:**

**public** **class** Source {

**public** String dnaComplement(String dna)

{

StringBuilder sb = **new** StringBuilder(); **for** (**int** i = 0; i < dna.length(); i++) { **if** (dna.charAt(i) == 'A') { sb = sb.append("T"); System.***out***.println(dna);

System.***out***.println(sb);

}

**if** (dna.charAt(i) == 'T') { sb = sb.append("A");

}

**if** (dna.charAt(i) == 'C') {

sb = sb.append("G");

}

**if** (dna.charAt(i) == 'G') {

sb = sb.append("C");

}

}

**return** sb.toString();

}

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

Source obj = **new** Source();

obj.dnaComplement("A");

}

}

**Bus Seat Availability:**

**class** Ticket {

**int** availableSeat = 30; **int** seat[];

**public** String bookshow(**int** seatNumber) **throws** Exception { **int** seat[] = **new** **int**[30];

**if** (availableSeat == 0) {

**throw** **new** NoSeatAvailableException("No Seat Number");

}

**if** (seatNumber < 1 || seatNumber > 30) {

**throw** **new** InvalidSeatNumberException("Invalid Seat Number");

}

**if** (seat[seatNumber] == 1) {

**throw** **new** SeatIsAlreadyTakenException("Seat Number is already booked");

}

**else**

availableSeat = availableSeat - 1;

**return** "Your reservation is confirmed";

}

}

**class** InvalidSeatNumberException **extends** Exception {

**public** InvalidSeatNumberException(String str) {

**super**(str);

}

}

**class** NoSeatAvailableException **extends** Exception {

**public** NoSeatAvailableException(String str) {

**super**(str);

}

}

**class** SeatIsAlreadyTakenException **extends** Exception {

**public** SeatIsAlreadyTakenException(String str) {

**super**(str);

}

} **public** **class** A **extends** Ticket {

**public** **static** **void** main(String args[]) **throws** Exception {

A a1 = **new** A(); a1.bookshow(31);

}

}

**Bandeja Paisa:**

**class** Product{

// Write Your Code Here..

Integer id;

String name;

Double price;

**public** Product(Integer id, String name, Double price) {

**this**.id = id; **this**.name = name;

**this**.price = price;

}

**public** Integer getId() {

**return** id;

}

**public** **void** setId(Integer id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** Double getPrice() {

**return** price;

}

**public** **void** setPrice(Double price) { **this**.price = price;

}

@Override

**public** String toString() { **return** "Product ["

+ "id=" + id

+ ", name=" + name

+ ", price=" + price + "]";

}

**class** Implementation{

**public** Long getProductCount(List<Product> list, String productName) {

**long** l=0;

**for** (Product p:list){ **if**(p.getName().equals(productName))

l++;

}

**return** **new** ~~Long~~ (l);

}

**public** Product getProductDetails(List<Product> list, String

productName, **int** id) {

**long** l=0;

**for** (Product p:list){ **if**((p.getId()).intValue()==id &&

p.getName().equals(productName))

**return** p;

}

**return** **null**;

}

**public** **class** Source {

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

}

}

}

}

**Vowel Manipulation:**

**import** java.util.Scanner;

**public** **class** ReplaceVowels {

**public** StringBuilder manipulateVowels(String str) {

StringBuilder sb = **new** StringBuilder();

**char** ch[] = str.toCharArray(); **for** (**int** i = 0; i < ch.length; i++) {

**if** (ch[i] == 'a' || ch[i] == 'e' || ch[i] == 'i' || ch[i] ==

'o' || ch[i] == 'u') {

ch[i] = 'b';

}

}

sb.append(ch);

**return** sb;

}

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

System.***out***.println("enter the string");

ReplaceVowels rv = **new** ReplaceVowels();

StringBuilder sb = rv.manipulateVowels(scan.next());

System.***out***.println(sb.toString());

scan.close();

}

}

**Employee Information Extraction:**

**class** Employee{

String name; String ssn; String dept; **int** salary;

**public** Employee(String name, String ssn, String dept, **int** salary){ **this**.name = name; **this**.ssn = ssn; **this**.dept = dept; **this**.salary = salary;

} **public** String toString(){ **return** "Employee{" +

"name='" + name + '\'' +

", ssn='" + ssn + '\'' +

", dept='" + dept + '\'' +

", salary=" + salary +

'}';

} }

**class** EmployeeImplementation{

**public** Employee getEmployeeInfo(String str){

**return** **new** Employee(

str.substring(0,str.indexOf("@")),

str.substring(str.indexOf("@")+1,str.indexOf("-")), str.substring(str.indexOf("-")+1,str.indexOf("#")),

Integer.*valueOf*(str.substring(str.indexOf("#")+1))

);

}

**public** String getEmployeeLevel(Employee e){

**int** ssn = Integer.*parseInt*(e.ssn.substring(e.ssn.length()-3));

**if**(ssn>0 && ssn<=60) **return** "L1"; **if**(ssn>60 && ssn<=120) **return** "L2"; **if**(ssn>120 && ssn<=180) **return** "L3"; **return** "L4";

}

}

**public** **class** Source {

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

EmployeeImplementation emp = **new** EmployeeImplementation();

Employee e = emp.getEmployeeInfo("Alex David@1PC16CS046-SDE#8");

System.***out***.println(e);

System.***out***.println(emp.getEmployeeLevel(e));

}

}

**Airing TV Show:**

**public** **class** Source {

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

}

|  |  |
| --- | --- |
|  | **public** String printIndex(ArrayList<String> list,**int** ind ) { |
|  | **return** (list.get(ind)); |
|  | } |
|  |  |
| String n){ | **public** ArrayList<String> addAfter(ArrayList<String> a, String m, |
|  | a.add(a.indexOf(m)+1,n); |
|  | **return** a; |
| } | } |

==================================================================================

**public** String printIndex(ArrayList<String> list, **int** ind) { **return** list.get(ind);

}

**public** ArrayList<String> addAfter(ArrayList<String> a, String m, String n) {

a.add(a.indexOf(m) + 1, n); **return** a;

}

**Harry's assignment:**

**class** StringPlay {

**int** convert;

**int** max;

**public** StringPlay() { **super**();

}

}

**class** StringMethods {

**public** **int** convertToInt(StringPlay sp, String str) { sp.convert = Integer.*parseInt*(str); **return** Integer.*parseInt*(str);

}

**public** **int** getMax(StringPlay sp, String str, **char** ch) { **int** count = 0;

**for** (**int** i = 0; i < str.length(); i++) {

**if** (str.charAt(i) == ch) {

count++;

}

}

sp.max = count;

**return** count;

}

}

**public** **class** Source { **public** **static** **void** main(String args[]) { StringMethods sm = **new** StringMethods();

StringPlay sp = **new** StringPlay();

System.***out***.println(sm.getMax(sp, "fgfgfgf", 'g'));

System.***out***.println(sm.convertToInt(sp, "123"));

}

}

**Retailer:**

**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\\'\\-]+([\\ AZa-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) {

}

}

**Sack Expiry:**

**public** **class** ExpiryCheck {

**static** **boolean** lengthCheck(String s) { **if** (s.length() == 12)

**return** **true**;

**return** **false**;

}

**static** **boolean** batchNumberCheck(String s) { String batch = s.substring(0, 4);

String checkone = batch.substring(0, 1);

String checktwo = batch.substring(1, 2); String checkthree = batch.substring(3, 4);

**char** x = checkone.charAt(0); **char** y = checktwo.charAt(0);

**char** z = checkthree.charAt(0);

**if** ((Character.*isUpperCase*(x) && Character.*isUpperCase*(y)) &&

Character.*isUpperCase*(z)) {

**try** {

**int** intValue = Integer.*parseInt*(batch.substring(2, 3));

**return** **true**;

} **catch** (NumberFormatException e) {

System.***out***.println("Input String cannot be parsed to

Integer.");

}

}

**return** **false**;

}

**static** **boolean** yearCheck(String s) {

String batch = s.substring(4, 8); **int** year = Integer.*parseInt*(batch);

**if** (year >= 2015 && year <= 2020) {

**return** **true**; } **else** { **return** **false**;

}

}

**static** **boolean** monthCheck(String s)

{

String batch=s.substring(8,10); **int** month =Integer.*parseInt*(batch);

**if**(month>=1 && month<=12) {**return** **true**;} **else**{**return** **false**;}

} }

**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;

}

}

**Students In Class Room:**

**import** java.util.ArrayList; **import** java.util.List;

**class** Implementation {

|  |  |
| --- | --- |
| { | **public** List<String> changeOccurence(List<String> list, String a, String b) |
|  | **for** (**int** i = 0; i < list.size(); i++) { |
|  | **if** (list.get(i).equalsIgnoreCase(a)) { |
|  | list.set(i, b); |
|  | } |
|  | } |
|  | **return** list; |
|  | } |
|  | **public** String getIndex(List<String> list) { |
|  | **if** (list.size() > 0) { |
|  | **return** list.get(0); |
|  | } **else** |
|  | **return** **null**; |
|  | } |
|  | **public** List<String> addAfter(List<String> list, String a, String b) { |
|  | **int** index = list.indexOf(a); |
|  | **if** (index != -1) { |
|  | list.add(index + 1, b); |
|  | **return** list.subList(index + 1, list.size()); |
|  | } |
|  | **return** list; |
| } | } |

**public** **class** Source {

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

List<String> list = **new** ArrayList(); list.add("A"); list.add("B"); list.add("C"); list.add("D");

Implementation implementation = **new** Implementation();

System.***out***.println(implementation.changeOccurence(list, "B", "S"));

System.***out***.println(implementation.getIndex(list));

System.***out***.println(implementation.addAfter(list, "B", "S"));

}

}

**Validate Coupon:**

**class** Validator {

// Write Your Code Here..

**public** String validateCoupon(Product p) **throws** Exception { String[] coupon = p.coupon.split("-"); **int** discount = integer.parseint(coupon[1]);

**if** (coupon[0].contentEquals(p.name) && (discount >= 10 && discount

<= 25)) {

**return** "valid coupon";

} **else**

**throw** **new** InvalidCouponException("invalid coupon");

}

**public** **double** netPrice(Product p) **throws** Exception { **double** netPrice = 0;

**try** {

validateCoupon(p);

**int** discount = integer.parseint(p.coupon.split("-")[1]); netPrice = p.price - ((p.price8discount) / 100);

} **catch** (InvalidCouponException e) {

}

**return** netPrice;

}

} **class** InvalidCouponException **extends** Exception { // Write Your Code Here..

**public** InvalidCouponException(String msg) { **super**(msg);

}

}

**Work Force Validation:**

**class** Workforce {

|  |
| --- |
| String  String |

firstName;

lastName;

}

**class** WorkforceValidation { **public** String nameValidation(Workforce w, String firstName, String lastName) {

**try** {

**if** (firstName == **null** || lastName == **null**) {

|  |  |  |
| --- | --- | --- |
|  |  | **throw** **new** NullPointerException("Entry Missing"); |
| { |  | } **else** **if** (firstName.length() == 0 || lastName.length() == 0) |
| bound"); |  | **throw** **new** StringIndexOutOfBoundsException("Index out of |
|  |  | } **else** **if** (Character.*isDigit*(firstName.charAt(0)) || |

Character.*isDigit*(lastName.charAt(0))) {

**throw** **new** InvalidNameException("First Character is

Invalid");

}

} **catch** (Exception e) {

}

w.firstName = firstName;

w.lastName = lastName; **return** firstName + lastName;

}

}

**class** InvalidNameException **extends** Exception {

**public** InvalidNameException(String message) {

**super**(message);

}

}

**public** **class** Source {

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

}

}

**Depository:**

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

**class** Repository { **static** String getCountryName(String countryCode) **throws** InvalidCodeException {

**int** num = Integer.*parseInt*(countryCode);

**if** ((num >= 70) && (num <= 99)) **return** ("India");

**else** **if** (countryCode.equals("011")) **return** ("Dial somewhere else outside of USA");

**else** **if** (num == 908) **return** ("US"); **else**

**throw** **new** InvalidCodeException("No country with the given code found");

}

}

**class** RepositoryImplementation { **static** String getCountry(String countryCode) **throws** InvalidCodeException { **if** ((countryCode.length() > 3) || (countryCode.length() < 2)) **throw** **new** InvalidCodeException("Invalid code detail found");

**else**

**return** (Repository.*getCountryName*(countryCode));

// throw new InvalidCodeException("No country with the given code found");

}

}

**class** InvalidCodeException **extends** Exception { **public** InvalidCodeException(String errorMessage) { **super**(errorMessage);

}

}

**public** **class** Source **extends** RepositoryImplementation { **public** **static** **void** main(String[] args) **throws** Exception{

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

String s1=scan.nextLine();

String cc=*getCountry*(s1);

System.***out***.print(*getCountry*(s1));

}

}

**Sherlock Needs Help:**

**import** java.util.regex.\*; **import** java.util.ArrayList;

**class** IdentifyWords {

**public** String getPossibleWords(String s1, String s2) {

String regex = s1.replace("\_", ".");

String check = regex.toUpperCase();

String[] possibleStrings = s2.split(":", -1); ArrayList<String> result = **new** ArrayList<String>(); **for** (String s : possibleStrings) {

**if** (Pattern.*matches*(check, s.toUpperCase())) { result.add(s.toUpperCase());

}

}

**if** (!result.isEmpty()) {

String joinedString = String.*join*(":", result); **return** joinedString;

} **else**

**return** "Code\_Error";

}

}

**Catch Me If You Can:**

**import** java.io.File;

**import** java.io.FileNotFoundException; **import** java.util.ArrayList; **import** java.util.List; **import** java.util.Scanner;

**public** **class** ExceptionCheck {

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

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

System.***out***.println("Enter the input string");

String inputStr = a.nextLine();

List<String> outputList = **new** ArrayList<>(); Implementation impl = **new** Implementation(); outputList = impl.numberCheck(inputStr);

System.***out***.println(outputList);

System.***out***.println("Enter the file path");

String inputPath = a.nextLine();

String fileCheckOutpiut = impl.fileCheck(inputPath);

System.***out***.println(fileCheckOutpiut);

}

}

**class** Implementation {

**public** String fileCheck(String inputPath) {

**try** {

File file = **new** File(inputPath); **if** (file.exists()) {

**return** "File Found";

} **else** {

**throw** **new** FileNotFoundException();

}

} **catch** (FileNotFoundException e) { e.printStackTrace(); **return** e.getMessage();

}

}

**public** List<String> numberCheck(String inputStr) {

String a = **null**;

List<String> chars = **new** ArrayList<>();

;

**for** (**int** i = 0; i < inputStr.length(); i++) {

**try** {

**if** (Character.*isDigit*(inputStr.charAt(i))) {

chars.add(String.*valueOf*(inputStr.charAt(i)));

} **else** {

**throw** **new** NumberFormatException();

}

} **catch** (NumberFormatException e) { a = "For input string " + '"' + inputStr.charAt(i) +

'"';

chars.add(a);

}

}

**return** chars;

}

}

**List of Series:**

**import** java.util.ArrayList;

**public** **class** Implementations {

**public** String getIndex(ArrayList<String> seriesList, **int** index) {

String s = seriesList.get(index);

// System.out.println(s);

**return** s;

}

**public** ArrayList<String> addAfterSeries(ArrayList<String> seriesList, String p, String q) {

**int** s = seriesList.indexOf(p); seriesList.add(s + 1, q);

**return** seriesList;

}

**public** **static** **void** main(String[] ar) { ArrayList<String> list = **new** ArrayList<>(); Implementations s = **new** Implementations();

list.add("sunil"); list.add("sai"); list.add("kumar");

s.getIndex(list, 1);

ArrayList<String> list1 = s.addAfterSeries(list, "sai", "satya");

**for** (String l : list1) {

System.***out***.println(l);

}

}

}

**Merit Scholarship:**

**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;

}

}

**Population:**

**import** java.util.ArrayList; **import** java.util.Collections; **import** java.util.HashMap; **import** java.util.List; **import** java.util.Map; **import** java.util.Map.Entry;

**class** Population {

Map<String, Integer> populationData = **new** HashMap<>();

String maxPopulation() {

List<String> countries = **new** ArrayList<>(); **int** max = Collections.*max*(populationData.values());

**for** (Entry<String, Integer> entry : populationData.entrySet()) { **if** (entry.getValue() == max) { countries.add(entry.getKey());

}

} **if** (countries.size() > 0) { **return** countries.get(0);

} **else** { **return** "";

}

}

**long** totalPopulation() {

**long** totalPopulation = 0;

**for** (**int** population : populationData.values()) { totalPopulation = totalPopulation + population;

}

**return** totalPopulation;

}

}

**Salary:**

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

**class** Salary {

HashMap<String, Integer> empList = **new** HashMap<>();

**public** **int** totalSalary() {

**int** total = 0;

**for** (Map.Entry<String, Integer> set : empList.entrySet()) {

total = total + set.getValue();

}

**return** total;

}

**public** String getSalary(String designation) {

**return** String.*valueOf*(empList.get(designation));

}

**public** **void** updateSalary(String designation, **int** newSalary) {

empList.replace(designation, newSalary);

}

}

**public** **class** Source {

**public** **static** **void** main(String args[]) { Salary obj = **new** Salary(); obj.empList.put("CEO", 20000); obj.empList.put("Developer", 50000); System.***out***.println(obj.totalSalary()); obj.updateSalary("Developer", 60000);

System.***out***.println("salary is " + obj.getSalary("Developer"));

System.***out***.println("salary is " + obj.totalSalary());

}

}

**Score Card:**

**import** java.util.ArrayList; **import** java.util.List; **class** Implementation {

|  |  |
| --- | --- |
| { | **public** List<String> changeOccurence(List<String> list, String a, String b) |
|  | **for** (**int** i = 0; i < list.size(); i++) { |
|  | **if** (list.get(i).equalsIgnoreCase(a)) { |
|  | list.set(i, b); |
|  | } |
|  | } |
|  | **return** list; |
|  | } |
|  | **public** String getIndex(List<String> list) { |
|  | **if** (list.size() > 0) { |
|  | **return** list.get(0); |
|  | } **else** |
|  | **return** **null**; |
|  | } |
|  | **public** List<String> addAfter(List<String> list, String a, String b) { |
|  | **int** index = list.indexOf(a); |
|  | **if** (index != -1) { |
|  | list.add(index + 1, b); |
|  | **return** list.subList(index + 1, list.size()); |
|  | } |
|  | **return** list; |
| } | } |

**public** **class** Source {

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

List<String> list = **new** ArrayList(); list.add("A"); list.add("B"); list.add("C"); list.add("D");

Implementation implementation = **new** Implementation();

System.***out***.println(implementation.changeOccurence(list, "B", "S"));

System.***out***.println(implementation.getIndex(list));

System.***out***.println(implementation.addAfter(list, "B", "S"));

}

}

**Unlock with Pin:**

**public** **class** Source { **public** **int** getCodeThroughStrings (String str) {

**int** digit=0, sum = 0, i=0;

String withoutspace;

**if**(str.contains(" ")) {

withoutspace= str.replaceAll("\\s", ""); i=withoutspace.length();

}

**else** { i=str.length();

}

**while**(i > 0) { digit = i%10;

sum=sum+digit; i=i/10;

}

**return** sum;

}

}

**Coupon Dunia:**

**import** java.util.Scanner;

**class** Source{

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

Scanner sc=**new** Scanner(System.***in***); String name=sc.next(); **double** p=sc.nextDouble();

String c=sc.next();

Product prod=**new** Product(name,c,p);

Validator val=**new** Validator(); String valCop="";

**try** {

valCop=val.validateCoupon(prod);

}

**catch**(Exception e){ // **TODO**: handle exception

}

**if**(valCop=="Valid Coupon"){

System.***out***.println(valCop);

System.***out***.println(val.netPrice(prod));

}

} **class** Product {

String name; **double** price; String coupon;

**public** Product(String name, String coupon, **double** price) {

**this**.name = name; **this**.coupon = coupon; **this**.price = price;

}

}

**class** Validator {

**public** String validateCoupon(Product p) **throws** Exception { **int** disc = Integer.*parseInt*(p.coupon.split("-")[1]);

String name = p.coupon.split("-")[0]; **if** (name == p.name && disc <= 25 && disc >= 10)

**return** "Valid Coupon";

**else** {

**throw** **new** InvalidCouponException("Invalid Coupon");

}

}

**public** **double** netPrice (Product p) { **int** disc = Integer.*parseInt*(p.coupon.split("-")[1]); **double** res = p.price- ((p.price\* disc) / 100);

**return** res;

}

} **class** InvalidCouponException **extends** Exception {

**public** InvalidCouponException(String message) {

**super** (message);

System.***out***.println(message);

}

}

} **Map Filter:**

**class** User {

**private** String firstName; **private** String lastName;

**private** **int** age;

User(String firstName, String lastName, **int** age) {

**this**.firstName = firstName; **this**.lastName = lastName; **this**.age = age;

}

**public** String getFirstName() { **return** firstName;

}

**public** **void** setFirstName(String firstName) {

**this**.firstName = firstName;

}

**public** String getLastName() { **return** lastName;

}

**public** **void** setLastName(String lastName) { **this**.lastName = lastName;

}

**public** **int** getAge() {

**return** age;

}

**public** **void** setAge(**int** age) { **this**.age = age;

}

}

**class** Implementation {

**public** List<User> filterAge(List<User> list) { List<User> list1 = **new** ArrayList<User>();

**for** (User u : list) { **if** (u.getAge() > 40) {

list1.add(u);

}

}

**return** list1;

}

**public** User findYoungest(List<User> list) {

**int** age = 0; User u = **null**;

**for** (User i : list) { **if** (age > i.getAge()) {

age = i.getAge();

u = i;

}

}

**return** u;

}

}

**Transaction Party:**

**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-Zaz]{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** String calcGST(Receipt r) {

String inputString = r.productsQR;

String ratePriceArray[] = inputString.split("@"); **int** sumOfAll = 0;

**for** (String current : ratePriceArray) { String splitIndividual[] = current.split(",");

String Rate = splitIndividual[0]; String Quantity = splitIndividual[1]; **int** rate = Integer.*parseInt*(Rate); **int** quantity = Integer.*parseInt*(Quantity);

sumOfAll = sumOfAll + rate \* quantity;

}

**int** GST = (sumOfAll \* 12 / 100); String s = Integer.*toString*(GST);

**return** s;

}

} **class** Source { **public** **static** **void** main(String[] args) { Scanner sc = **new** Scanner(System.***in***);

String seller = sc.nextLine();

String buyer = sc.nextLine();

String productQR = sc.nextLine();

TransactionParty t1 = **new** TransactionParty(seller, buyer);

Receipt r1 = **new** Receipt(t1, productQR);

GenerateReceipt g1 = **new** GenerateReceipt();

System.***out***.println(g1.*verifyParty*(r1)); System.***out***.println(g1.calcGST(r1));

sc.close();

}

}

**Shopping Cart:**

**class** Product { **private** String id; **private** String name; **private** **int** quantity; **private** **float** price;

**public** Product(String id, String name, **int** quantity, **float** price) {

**super**(); **this**.id = id; **this**.name = name;

**this**.quantity = quantity;

**this**.price = price;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) { **this**.name = name;

}

**public** **int** getQuantity() { **return** quantity;

}

**public** **void** setQuantity(**int** quantity) { **this**.quantity = quantity;

}

**public** **float** getPrice() {

**return** price;

}

**public** **void** setPrice(**float** price) { **this**.price = price;

}

}

**public** **class** Source {

ArrayList<Product> productList = **new** ArrayList<>();

**public** **float** netPrice() {

**float** sum = 0;

**for** (Product p : productList) {

sum = sum + p.getQuantity() \* p.getPrice();

}

**return** sum;

}

**public** **int** totalItem()

{

**int** sum = 0; **for**(Product p: productList)

{

sum = sum+p.getQuantity();

}

**return** sum;

}

**Holiday Package:**

**import** java.util.ArrayList; **import** java.util.Collections; **import** java.util.HashMap; **import** java.util.List;

**public** **class** Source {

HashMap<String, Integer> holidayPkg = **new** HashMap<>();

**public** **int** cheapestPackage(**int** numberOfPlaces) {

List<Integer> list = **new** ArrayList<>(); **for** (String s : holidayPkg.keySet()) { list.add(holidayPkg.get(s));

}

Collections.*sort*(list); **int** result = 0;

**int** index = 0;

**while** (numberOfPlaces > 0) {

result += list.get(index);

index++;

numberOfPlaces--;

}

**return** result;

}

**public** **int** maximumPlace(**int** budget) { List<Integer> list = **new** ArrayList<>();

**for** (String s : holidayPkg.keySet()) { list.add(holidayPkg.get(s));

}

Collections.*sort*(list);

**int** result = 0;

**int** index = 0;

**while** (budget > 0) { **if** ((budget - list.get(index)) > 0) { budget -= list.get(index);

result++; } **else** { budget = -1;

}

index++;

}

**return** result;

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

Source obj = **new** Source();

obj.holidayPkg.put("Delhi", 5000);

obj.holidayPkg.put("Jaipur", 4000);

obj.holidayPkg.put("Agra", 2500); obj.holidayPkg.put("Goa", 7000);

System.***out***.println(obj.cheapestPackage(2));

System.***out***.println(obj.maximumPlace(3000));

}

}