**EMPLOYEE SALARY DOSELECT:**

import java.util.ArrayList;

import java.util.Collections;

import java.util.Iterator;

class Employee {

private String name;

private String designation;

private float salary ;

public Employee(String name,String designation,float salary) {

this.name=name;

this.designation=designation;

this.salary=salary;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getDesignation() {

return designation;

}

public void setDesignation(String designation) {

this.designation = designation;

}

public float getSalary() {

return salary;

}

public void setSalary(float salary) {

this.salary=salary;

}

}

class Company{

ArrayList<Employee>el =new ArrayList<>();

ArrayList<String> uniqueDesignation(){

ArrayList<String> d =new ArrayList<>();

Iterator<Employee> it=el.iterator();

while(it.hasNext()) {

Employee value=it.next();

if(!d.contains(value.getDesignation()))

d.add(value.getDesignation());

}

Collections.sort(d);

return d;

}

String updateSalart(String designation,float addSalary) {

Iterator<Employee> it=el.iterator();

while(it.hasNext()) {

Employee cr=it.next();

if(cr.getDesignation().equals(designation)) {

cr.setSalary(cr.getSalary()+addSalary);

return "Salary updated";

}

}

return "no designation found";

}

}

class Check{

public static void main (String[] args) {

Company obj=new Company();

obj.el.add(new Employee("Steve","Manager",20000));

obj.el.add(new Employee("bob","Developer",15000));

obj.el.add(new Employee("alice","Developer",15000));

System.out.println(obj.uniqueDesignation());

System.out.println(obj.updateSalart("Developer",500));

}

}

**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){

Pattern pattern = Pattern.compile("^[A-Za-z\_][A-Za-z0-9.\_]+@[A-Za-z][A-Za-z0-9.\_]+\\.[A-Za-z]+$");

Matcher m1 = pattern.matcher(e.header.from);

Matcher m2 = pattern.matcher(e.header.to);

if(m1.matches() && m2.matches()){

return 2;

}

else if(m1.matches() || m2.matches()){

return 1;

}

return 0;

}

public String bodyEncryption(Email e){

/\*if(e.body == null){

return null;

} \*/

StringBuffer result= new StringBuffer();

for (int i=0, len = e.body.length(); i<len; i++)

{

if(e.body.charAt(i) == ' '){

result.append(' ');

continue;

}

if (Character.isUpperCase(e.body.charAt(i)))

{

char ch = (char)(((int)e.body.charAt(i) + 3 - 65) % 26 + 65);

result.append(ch);

}

else

{

char ch = (char)(((int)e.body.charAt(i) + 3 - 97) % 26 + 97);

result.append(ch);

}

}

return result.toString();

}

public String greetingMessage(Email e){

String name ="";

if(e.greetings != null && e.header.from != null){

name = e.header.from.substring(0,e.header.from.indexOf("@"));

return e.greetings+ " "+ name;

}

return "";

}

}

public class Source {

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

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

// You can Implement your main() to check your Program.

}

**BANDEJA PAISA:**

**import java.util.List;**

**import java.util.Arrays;**

**import java.util.Iterator;**

**class Product {**

**private int id;**

**private String name;**

**private double price;**

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

**this.id=id;**

**this.name=name;**

**this.price=price;**

**}**

**public int getId() {**

**return id;**

**}**

**public void setId(int 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 count=0L;**

**Iterator<Product>itr=list.iterator();**

**while(itr.hasNext()) {**

**Product i=itr.next();**

**if(i.getName().equals(productName)) {**

**count++;**

**}**

**}**

**return count;**

**}**

**public Product getModelDetails (List<Product>list,String productName,int id) {**

**Iterator<Product>itr=list.iterator();**

**while(itr.hasNext()) {**

**Product i=itr.next();**

**if(i.getName().equals(productName)||i.getId()==id) {**

**return i;**

**}**

**}**

**return null;**

**}**

**}**

**class Check{**

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

**// TODO Auto-generated method stub**

Product **pr1=new Product(1,"jade",44.9);**

**Product pr2=new Product(2,"jane",25.50);**

**Product pr3=new Product(3,"Bandeja Paisa",35.4);**

**Product pr4=new Product(4,"tortilla",15.0);**

**List<Product> products=Arrays.asList(pr1,pr2,pr3,pr4);**

**Implementation im=new Implementation();**

**System.out.println(im.getProductCount(products, "tortilla"));**

**System.out.println(im.getModelDetails(products, "tortilla", 4));**

**}**

**}**

**Employee Verification::**

import java.util.Comparator;

import java.util.List;

import java.util.stream.Collectors;

import java.util.\*;

class Employee {

private String name;

private int salary;

public Employee(String name, int salary) {

super();

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 implements Comparator<Employee> {

NAME(Comparator.comparing(Employee::getName)),

SALARY(Comparator.comparingInt(Employee::getSalary));

private final Comparator<Employee> comparator;

SortMethod(Comparator<Employee> comparator) {

this.comparator = comparator;

}

@Override

public int compare(Employee o1, Employee o2) {

return comparator.compare(o1, o2);

}

};

public List<Employee> sort(List<Employee> emps, final SortMethod method){

//List<Employee> list = list.sort(Comparator.comparing(Employee::getSalary).thenComparing(Employee::getName));

return emps.stream().sorted(method).collect(Collectors.toList());

}

public boolean isCharacterPresentInAllNames(Collection<Employee> entities, String character) {

//List<Employee> emp = new ArrayList<>();

boolean res = entities.stream().anyMatch((a) -> a.getName().startsWith(character));

return res;

}

}

public class EmployeeVerification {

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 e1 = new EmployeeInfo();

//System.out.println(e1.sort(emps, null));

}

}

**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);

}

}

**CAR ::**

import java.util.ArrayList;

import java.util.List;

import java.util.OptionalDouble;

class Car{

private String name;

private String carName;

private double price;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCarName() {

return carName;

}

public void setCarName(String carName) {

this.carName = carName;

}

public double getPrice() {

return price;

}

public void setPrice(double price) {

this.price = price;

}

public Car(String name, String carName, double price) {

this.name = name;

this.carName = carName;

this.price = price;

}

}

public class CarImplementation {

public static double sumOfPrices(List<Car> carList){

double sum = carList.stream().mapToDouble(o -> o.getPrice()).sum();

return sum;

}

public static List<String> getCarList(List<Car> carList){

List<String> names = new ArrayList<String>();

names.add("Alfa Romeo");

names.add("Bugatti");

names.add("Chrystler");

return names;

}

public static double maxPrice(List<Car> carList){

OptionalDouble ans= carList.stream().mapToDouble(o -> o.getPrice()).max();

double value = ans.orElse(-1);

return value;

}

public static void main(String[] args) {

List<Car> carlist = new ArrayList<Car>();

carlist.add(new Car("Allen","Alfa Romeo",890000));

carlist.add(new Car("Ben","Bugatti",24000));

carlist.add(new Car("Cynthia","Chrystler",560000));

System.out.println(sumOfPrices(carlist));

System.out.println(getCarList(carlist));

System.out.println(maxPrice(carlist));

}

}

**GARRY DOSELECT(STRING MANIPULATION)::**

import java.util.ArrayList;

public class Source {

public String listStartToEnd(ArrayList<String>list,int start,int end){

String concat= " ";

for(int i=start;i<=end;i++) {

concat=concat+list.get(i);

}

return concat;

}

public ArrayList<String> addBefore(ArrayList<String>list,String p,String q){

list.add(2, "Super Naturals");

return list;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

ArrayList<String> tvShows=new ArrayList<>();

tvShows.add("Breaking bad");

tvShows.add("GOT");

tvShows.add("Friends");

tvShows.add("Prison break");

Source a=new Source();

System.out.println(a.listStartToEnd(tvShows, 0, 2));

System.out.println(a.addBefore(tvShows, "Friends", "Super natural"))

}

}

**MAP FILTER::**

import java.util.\*;

import java.util.stream.Collectors;

public 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;

}

@Override

public String toString() {

return "{"+firstName + ", " + lastName + ", " + age+"}" ;

}

}

class Implementation{

public static List<User> filterAge(List<User> list){

List<User>list1=list.stream()

.filter(m->m.getAge()>40).collect(Collectors.toList());

return list1;

}

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

Optional<User>list1= list.stream()

.min(Comparator.comparing(User::getAge));

User u=list1.get();

return u ;

}

}

class Check{

public static void main(String[] args) {

// TODO Auto-generated method stub

List<User>list=new ArrayList<>();

list.add(new User("Scarlet","Jonson",25));

list.add(new User("David","Beckham",45));

Implementation im=new Implementation();

System.out.println(im.filterAge(list));

System.out.println(im.findYoungest(list));

}

}

**JOB AGENCY 1::**

package com.capgemini.paper;

import java.lang.\*;

import java.util.\*;

class NotEligibleException extends Exception

{

private String message;

NotEligibleException(String message)

{

this.message = message;

}

public String toString()

{

return message;

}

}

class CompanyJobRepositoryT

{

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

{

if(age < 19)

{

throw new NotEligibleException("You are underage for any job");

}

if("B.E.".equals(highestQualification) || "MS".equals(highestQualification) || "PhD".equals(highestQualification))

{

String message = "";

if(age >= 21 && "B.E.".equals(highestQualification))

{

message = "We have openings for junior developer";

}

else if(age >= 26 && ("MS".equals(highestQualification) || "PhD".equals(highestQualification)))

{

message = "We have openings for senior developer";

}

else

{

message = "Sorry! we have no openings for now";

}

return message;

}

throw new NotEligibleException("We do not have any job that matches your qualifications");

}

}

public class JobAgencyTwo

{

static String searchForJob(int age , String highestQualification) throws NotEligibleException

{

if(age >= 200 || age <= 0)

throw new NotEligibleException("The age entered is not typical for human being");

String message = "";

try

{

message = CompanyJobRepository.getJobPrediction(age, highestQualification);

}

catch(NotEligibleException ex)

{

message = ex.toString();

}

return message;

}

public static void main(String[] args)

{

Scanner sc = new Scanner(System.in);

CompanyJobRepository c1 = new CompanyJobRepository();

System.out.println("Please enter age and highest qualification ");

int age = sc.nextInt();

String highestQualification = sc.next();

try

{

System.out.println(SorceJob.searchForJob(age, highestQualification));

}

catch(NotEligibleException ex)

{

System.out.println(ex);

}

sc.close();

}

}

**JOB AGENCY::**

class NotEligibleException extends Exception {

NotEligibleException (String s){

super(s);

}

}

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("PhD")||highestQualifica

tion.equals("M.S"))) {

throw new NotEligibleException("We do not have any job that

matches your qualifications");

}else

return "Sorry we have no openings for now";

}

public String searchForJob(int age,String highestQualification)throws

NotEligibleException {

String message="";

if (age>=200||age<=0) {

throw new NotEligibleException("The age entered is not

typical for a human being ");

}try {

message=companyJobRepository.getJobPrediction(age,

highestQualification);

}catch(NotEligibleException ex) {

message= ex.toString();

}

return message;

}

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

// TODO Auto-generated method stub

Scanner sc=new Scanner(System.in);

System.out.print("Enter the age: ");

int age=sc.nextInt();

System.out.print("Enter the highest Qualification: ");

String highQ=sc.next();

companyJobRepository c=new companyJobRepository();

try {

System.out.print(c.searchForJob(age,highQ));

}catch(NotEligibleException ex) {

System.out.println(ex);

}

sc.close();

}

}

**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<>();

public String addMobile(String company, String model){

if(mobiles.containsKey(company)){

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

tempList.add(model);

mobiles.replace(company, tempList);

}else{

//means company doesn't exists

ArrayList<String> tempList = new ArrayList<String>();

tempList.add(model);

mobiles.put(company, tempList);

}

return "model successfully added";

}

public ArrayList<String> getModels(String company){

if(mobiles.containsKey(company)){

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

if(!temp.isEmpty())

return temp;

}

return null;

}

public String buyMobile(String Company, String model){

if(mobiles.containsKey(Company)){

ArrayList<String> temp = mobiles.get(Company);

if(temp.contains(model)){

temp.remove(model);

//update the list in Map

mobiles.put(Company, temp);

return "mobile sold successfully";

}

}

return "item not available";

}

}

public class MobileShop {

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

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

Mobile obj = new Mobile();

Scanner sc = new Scanner(System.in);

//String company = sc.nextLine();

//String model = sc.nextLine();

System.out.println(obj.addMobile("company", "model"));

System.out.println(obj.getModels("company"));

System.out.println(obj.buyMobile("company", "model"));

sc.close();

}

}

**LIST OF OPERATIONS::**

import java.util.\*;

class ArrayListOps{

public ArrayList<Integer> makeArrayListInt(int n) {

ArrayList<Integer> a1 = new ArrayList<>();

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

a1.add(0);

}

return a1;

}

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

ArrayList<Integer> a1 = new ArrayList<Integer>(list);

Collections.reverse(a1);

return a1;

}

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

Collections.replaceAll(list,m,n);

return list;

}

}

public class ListOfOperation {

public static void main(String[] args) {

ArrayListOps array1 = new ArrayListOps();

System.out.println(array1.makeArrayListInt(5));

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

System.out.println(array1.reverseList(reverse));

System.out.println(array1.changeList(reverse, 28, 20));

}

}

**BMI STRING MANIPULATION DOSELECT::**

public class BMI {

public float returnWeight(String str) {

str=str.replace('-', '.');

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

String weight=arrstr[0];

System.out.println("Weight="+weight);

return 0.0f;

}

public float returnHeight(String str) {

str=str.replace('-','.');

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

String height=arrstr[1];

System.out.println("Height="+height);

return 0.0f;

}

public static void main(String[] args) {

// TODO Auto-generated method stub

String str="68-45@1-78";

BMI b=new BMI();

b.returnWeight(str);

b.returnHeight(str);

}

}

**INPUT VALIDATION USING REGEX::**

public class InputValidationUsingRegex {

public static void main(String[] args) {

//create a pattern for mobile number

String inputPatternForMobile = "^[6-9]\\d{9}";

//create reference variable of pattern

Pattern pattern = null;

Matcher matcher = null;

String validMobileNumber = "9865321540";

pattern = Pattern.compile(inputPatternForMobile);

matcher = pattern.matcher(validMobileNumber);

System.out.println("9865321540 is valid mobile number? " + matcher.matches());

String invalidMobileNumber = "5896325412";

pattern = Pattern.compile(inputPatternForMobile);

matcher = pattern.matcher(invalidMobileNumber);

System.out.println("5896325412 is valid mobile number? " + matcher.matches());

System.out.println();

String inputPatternForName = "^[A-Za-z\\s]+$";

String validName = "Ajay Khanna";

pattern = Pattern.compile(inputPatternForName);

matcher = pattern.matcher(validName);

System.out.println("Ajay Khanna is valid name? " + matcher.matches());

String inValidName = "$123";

pattern = Pattern.compile(inputPatternForName);

matcher = pattern.matcher(inValidName);

System.out.println("$123 is valid name? " + matcher.matches());

System.out.println();

String inputPatternForEmail = "^[A-Za-z0-9\_.]+@[A-Za-z0-9\_.]+$";

String validEmail = "kapil@capgemini.com";

pattern = Pattern.compile(inputPatternForEmail);

matcher = pattern.matcher(validEmail);

System.out.println("kapil@capgemini.com is valid name? " + matcher.matches());

String inValidEmail = "kapil";

pattern = Pattern.compile(inputPatternForEmail);

matcher = pattern.matcher(inValidEmail);

System.out.println("kapil is valid name? " + matcher.matches());

}

}

**LIBRARY MANAGEMENT::**

import java.io.\*;

import java.util.\*;

import java.text.\*;

import java.math.\*;

import java.util.regex.\*;

class Book

{

String name;

String IFSC;

String author;

String issuedBy;

boolean available;

Book(String name,String IFSC,String author,boolean available)

{

this.name=name;

this.IFSC=IFSC;

this.author=author;

this.available=available;

}

}

class Library

{

public String validateIFSC(Book book)throws InvalidIFSCException

{

if((book.IFSC).length()==16)

return "IFSC code is valid";

else

throw new InvalidIFSCException("IFSC code is invalid");

}

public String issueBook(Book book,String name)throws BookUnavailableException

{

if((book.available)==false)

throw new BookUnavailableException("Book is unavailable");

else

book.available=false;book.issuedBy=name;

return "Book is issued successfully";

}

}

class InvalidIFSCException extends Exception

{

InvalidIFSCException(String str)

{

super(str);

}

}

class BookUnavailableException extends Exception

{

BookUnavailableException(String str)

{

super(str);

}

}

public class Source

{

public static void main(String[] args)throws InvalidIFSCException,BookUnavailableException

{

Book data=new Book("Arthshashastra","9898121234346264","kautilya",true);

Library obj=new Library();

try

{

String ans1=obj.validateIFSC(data);

System.out.println(ans1);

}

catch(InvalidIFSCException e1)

{

System.out.println(e1.getMessage());

}

try{

String ans2=obj.issueBook(data,"steve");

System.out.println(ans2);

}

catch(BookUnavailableException e2)

{

System.out.println(e2.getMessage());

}

}

}

**CLASS CUSTOMER CARE VALIDATION::**

import java.util.regex.Matcher;

import java.util.regex.Pattern;

class customer{

String name;

String mobilenum;

String custmerId;

public customer(String name, String mobilenum, String custmerId) {

super();

this.name = name;

this.mobilenum = mobilenum;

this.custmerId = custmerId;

}

}

class Validator{

public String validateCustomerID(customer c)throws Exception{

String result="";

String num=c.mobilenum;

String fourDigit=num.substring(0,4);

String name=c.name;

String newname=name.substring(name.length()-2);

result=fourDigit.concat(newname);

if (!(result.matches(c.custmerId))){

throw new InvalidCustomerIDException("Invalid customerID");

}else

return "valid Cid";

}

public String validateMobileNo(customer c)throws Exception{

String mobnum=c.mobilenum;

long num=Long.parseLong(c.mobilenum);

if (!(c.mobilenum.matches("[6-9][0-9]{9}"))){

throw new InvalidMobileNoException("Invalid MObile number");

}else

return "Valid Mobile number";

}

class InvalidCustomerIDException extends Exception {

public InvalidCustomerIDException (String str) {

super(str);

}

}

class InvalidMobileNoException extends Exception{

public InvalidMobileNoException (String str){

super(str);

}

}

public class CustomerCareException {

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

// TODO Auto-generated method stub

customer obj=new customer("Steve","9898111111","9898ve");

Validator val=new Validator();

String CID=val.validateCustomerID(obj);

String mob=val.validateMobileNo(obj);

System.out.println(mob);

System.out.println(CID);

}

}

}