1)Given a string and an int n, return a string made of the first n characters of the string, followed by the first n-1 characters of the string, and so on. You may assume that n is between 0 and the length of the string, inclusive (i.e. n >= 0 and n <= str.length()).

repeatFront("Chocolate", 4) → "ChocChoChC" repeatFront("Chocolate", 3) → "ChoChC" repeatFront("Ice Cream", 2) → "IcI"

**package** com.techment.test;

**import** java.util.Scanner;

**public** **class** Solution1 {

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

Solution1 solution1 = **new** Solution1();

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

System.***out***.println("enter number:");

**int** num = scanner.nextInt();

**try** {

System.***out***.println("enter any string:");

solution1.repeatFront(scanner.next(), num);

}

**catch** (Exception e) {

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

}

}

**public** String repeatFront(String str, **int** num) {

**int** size = str.length();

String newstr ="";

**if**(num>0)

{

**for** (**int** i = num; num>0 ; num--) {

newstr+= str.substring(0,num);

}

System.***out***.print(newstr);

}

**return** "wrong input";

}

}

2) Given a string and an int n, return a string made of n repetitions of the last n characters of the string. You may assume that n is between 0 and the length of the string, inclusive.

repeatEnd("Hello", 3) → "llollollo" repeatEnd("Hello", 2) → "lolo" repeatEnd("Hello", 1) → "o"

**package** com.techment.test;

**import** java.util.Scanner;

**public** **class** Solution2 {

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

Solution2 solution2 = **new** Solution2();

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

System.***out***.println("enter number:");

**int** num = scanner.nextInt();

**try** {

System.***out***.println("enter any string:");

solution2.repeatEnd(scanner.next(), num);

}

**catch** (Exception e) {

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

}

}

**public** String repeatEnd(String str, **int** num) {

**int** size = str.length();

String newstr ="";

**if**(num>0)

{

**for** (**int** i = 0; i<num ; i++) {

newstr+= str.substring(size-num , size);

}

System.***out***.print(newstr);

}

**return** "wrong input";

}

}

3) ArrayList containing the firstName,lastName and weight of the wrestler that will be used to create a stream Task: create a WWE class which has the following private member firstName String, lastName String, weight int ->Define parameterized Constructor ->Define Setter and getter

Create a WWEWrestlerImplementation class which performs operation as per given requirement using stream api

a)count the number of wrestlers

b)sum of all weight of all wrestlers whose weight is above 200

c) create a method printFirstName(List list) that returns the first name of the all wrestler.

d)write a method to find minimum weight of the wrestler

**public** **class** WWE {

**private** String firstName,lastName ;

**private** **int** weight;

**public** WWE(String firstName, String lastName, **int** weight) {

**super**();

**this**.firstName = firstName;

**this**.lastName = lastName;

**this**.weight = weight;

}

**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** getWeight() {

**return** weight;

}

**public** **void** setWeight(**int** weight) {

**this**.weight = weight;

}

}

**import** java.util.ArrayList;

**import** java.util.List;

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

**public** **class** WWESolution {

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

ArrayList<WWE> wwe = **new** ArrayList<WWE>();

wwe.add(**new** WWE("oswald", "osd", 200));

wwe.add(**new** WWE("vini", "vnd", 100));

wwe.add(**new** WWE("joy", "lew", 208));

wwe.add(**new** WWE("henry", "penguin", 130));

wwe.add(**new** WWE("daisy", "flower", 250));

WWESolution solution = **new** WWESolution();

**int** minweight = solution.minWeight(wwe);

System.***out***.println("Min weight="+minweight);

**int** maxweight = solution.maxWeight(wwe);

System.***out***.println("Max weight="+maxweight);

**int** count = solution.countWrestlers(wwe);

System.***out***.println("Total Wrestlers="+count);

**long** totalweight = solution.totalWeight(wwe);

System.***out***.println("Total weight of all Wrestlers whose weight>200="+totalweight);

solution.printFirstName(wwe);

}

**int** minWeight( ArrayList<WWE> wwe )

{

**int** minweight = wwe.stream().collect(Collectors.*summarizingInt*(WWE::getWeight)).getMin();

**return** minweight;

}

**int** maxWeight( ArrayList<WWE> wwe )

{

**int** maxweight = wwe.stream().collect(Collectors.*summarizingInt*(WWE::getWeight)).getMax();

**return** maxweight;

}

**int** countWrestlers(ArrayList<WWE> wwe)

{

**int** total = (**int**) wwe.stream().count();

**return** total;

}

**long** totalWeight(ArrayList<WWE> wwe)

{

**long** totalweight = wwe.stream().filter(e->e.getWeight()>200).collect(Collectors.*summarizingInt*(WWE :: getWeight)).getSum();

**return** totalweight;

}

**void** printFirstName(List<WWE> wwe)

{

//wwe.stream().forEach(i->System.out.println(getFirstName()));

//return fname;

}

}

4) : Write a java program that simulates a traffic light. The program lets the user select one of three lights: red, yellow, or green with radio buttons. On entering the choice, an appropriate message with “stop” or “ready” or “go” should appear in the console .Initially there is no message shown.

**package** com.techment.solution4;

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** java.applet.\*;

**public** **class** TrafficLight **extends** Applet **implements** ItemListener{

String msg="";

Checkbox red , green ,yellow;

CheckboxGroup cbg;

**public** **void** init() {

cbg = **new** CheckboxGroup();

red = **new** Checkbox("Red",cbg,**false**);

green = **new** Checkbox("Green",cbg,**false**);

yellow = **new** Checkbox("Yellow",cbg,**false**);

add(red);

add(green);

add(yellow);

red.addItemListener(**this**);

green.addItemListener(**this**);

yellow.addItemListener(**this**);

}

@Override

**public** **void** itemStateChanged(ItemEvent e) {

repaint();

}

**public** **void** paint(Graphics g)

{

msg = "";

msg += cbg.getSelectedCheckbox().getLabel();

g.drawString(msg, 50, 100);

}

}

5) Create a method which accepts the id and the age of people as a Map and decide if they are eligible for vote. A person is eligible for vote if his age is greater than 18. Add the IDs of all the eligible persons to list and return the list.

Method Name votersList

Method Description Generate the list of voters based on the ages of the people Argument Map

Return Type List

Logic Accept a map with ID as key and Date of Birth as value and check if the person is eligible to vote. A person is eligible for vote for if his age is greater than 18. If the person is eligible add his ID to the list.

**package** com.techment.solution5;

**import** java.util.ArrayList;

**import** java.util.HashMap;

**import** java.util.Map;

**class** Voters

{

String name;

**int** id,age;

**public** Voters(String name, **int** id, **int** age) {

**super**();

**this**.name = name;

**this**.id = id;

**this**.age = age;

}

@Override

**public** String toString() {

**return** "Voters [name=" + name + ", id=" + id + ", age=" + age + "]";

}

}

**public** **class** Voting {

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

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

list.add(**new** Voters("raji", 1001, 23));

list.add(**new** Voters("shejal", 1002, 18));

list.add(**new** Voters("sunita", 1003, 50));

list.add(**new** Voters("pkd", 1004, 55));

list.stream().forEach(s->System.***out***.println(s));

Voting voting = **new** Voting();

voting.votesrList(Map<id , age>);

}

}