```
import java.util.LinkedList;

/*
1. Attivare una SIM

2. Disattivare una SIM

3. Acquisto SIM da parte del negozio. Le SIM acquistate dal negozio andranno in
magazzino.

4. Ricaricare una SIM: si dovrà chiedere l'operatore, il numero di telefono e
l'importo della ricarica

5. Richiedere la portabilità del numero: si dovrà chiedere l'ICCID della sim,
il numero di telefono di cui effettuare la portabilità, l'operatore attuale e
il nuovo operatore. Nella fase di portabilità si dovrà chiedere all'utente se
trasferire o meno il credito residuo.

6. Visualizzare le SIM attive/disattive in negozio in base alla scelta
dell'utente.
*/
public class Main {
    public static void main(String[] args) {
        Negozio negozio = new Negozio();
        char sc;
        do {
            menu();
            sc = Input.leggiChar('a','g',"Scelta: ");
            switch (sc) {
                case 'a': {
                    attivaSim(negozio);
                    break;
                }
                case 'b': {
                    disattivaSim(negozio);
                    break;
                }
                case 'c': {
                    aggiungiMagazzino(negozio);
                    break;
                }
                case 'd': {
                    ricaricaSim(negozio);
                    break;
                }
                case 'e': {
                    portabilitaSim(negozio);
                    break;
                }
                case 'f':{

                    break;
                }
            }
```

```
        }while(sc!='g');
    }

    private static void aggiungiMagazzino(Negozio negozio) {
        int num = Input.leggiInt(0,Integer.MAX_VALUE,"Inserire il numero di Sim
con il quale rifornirsi");
        Sim sim;
        for(int i=0;i<num;i++){
            String operatore = Input.leggiStr("Inserire nome operatore");
            String Iccid = Input.leggiStr("Inserire Iccid");
            String numero = Input.leggiStr("Inserire numero");
            sim = new Sim("NULL",operatore,numero,Iccid);
            negozio.addSimToStorage(sim);
        }
    }
    private static void disattivaSim(Negozio negozio) {
        String iccid = Input.leggiStr("Inserire l'iccid della sim da
disattivare");
        if(negozio.deactivateSim(iccid)){
            System.out.println("Sim disattivata con successo");
        }else
            System.out.println("Sim non disattivata, l' Iccid inserito non
esiste");
    }


    public static void attivaSim(Negozio negozio){
        String msg, nome;
        nome = Input.leggiStr("Inserisci il nome dell'intestatario sim");
        msg = negozio.sellSim(nome);
        System.out.println(msg);
    }
    public static void ricaricaSim(Negozio negozio){
        char sc;
        String numero = Input.leggiStr("Inserisci il numero di cui effettuare
la ricarica");
        String operatore = Input.leggiStr("Inserisci l'operatore del numero
telefonico");
        Sim sim = negozio.findNumeroOperatore(numero,operatore);
        if(sim == null){
            System.out.println("Scegliere il taglio della ricarica");
            System.out.println("a) 5 €");
            System.out.println("b) 10 €");
            System.out.println("c) 20 €");
            sc = Input.leggiChar('a','c',"Scelta:");
            switch (sc){
                case 'a':{
                    sim.addCredito(5f);
                    break;
                }
                case 'b':{
                    sim.addCredito(10f);
                    break;
                }
```

```
                case 'c':{
                    sim.addCredito(20f);
                    break;
                }
            }
        }else{
            System.out.println("Il numero inserito è inesistente, quindi è
impossibile effettuare la ricarica");
        }
    }

    private static void portabilitaSim(Negozio negozio) {
        String iccid = Input.leggiStr("Inserisci iccid scheda sim");
        String numero = Input.leggiStr("Inserisci numero di telefono da
portare");
        String operatoreAttuale = Input.leggiStr("Inserisci l'operatore
attuale");
        String nuovoOperatore = Input.leggiStr("Inserire Il nuovo operatore");
        Sim sim = negozio.findNumeroOperatoreIccid(numero,
operatoreAttuale,iccid);
        Sim newSim;
        if(sim!=null){
            int sc = Input.leggiInt(0,1,"Vuoi trasferire il credito (1 si/0
no)");
            switch (sc){
                case 0:{
                    newSim = new Sim(sim);
                    newSim.setOperatore(nuovoOperatore);
                    newSim.setCredito(0);
                    negozio.addSimToSold(newSim);
                }
                case 1:{
                    newSim = new Sim(sim);
                    newSim.setOperatore(nuovoOperatore);
                    negozio.addSimToSold(newSim);
                }
            }
        }else
            System.out.println("Non è stata trovata la sim");
    }


    public static void visualizzaSim(Negozio negozio) {
        char scelta = Input.leggiChar('a', 'b', "a) Visualizza SIM attive\nb)
Visualizza SIM disattive\nScelta: ");

        LinkedList<Sim> simsDaVisualizzare = negozio.getSimToVisualize(scelta
== 'a');

        if (simsDaVisualizzare.isEmpty()) {
            System.out.println("Nessuna SIM trovata.");
        } else {
            for (Sim sim : simsDaVisualizzare) {
                System.out.println(sim);
```

```
            }
        }
    }




    public static void menu(){
        System.out.println("a) Attiva SIM");
        System.out.println("b) Disattiva SIM");
        System.out.println("c) Aggiungi SIM a magazzino");
        System.out.println("d) Ricarica SIM");
        System.out.println("e) Richiedi portabilità numero");
        System.out.println("f) Visualizza SIM attive/disattive");
        System.out.println("g) Termina programma");
    }

}
////////////////////////////////////////////////////////////////////////////
/////////

import java.util.LinkedList;

public class Negozio {
    private String pIva,nome,indirizzo;
    private LinkedList<Sim> simSold,simBuy;

    public Negozio(){
        setNome("Mario Rossi");
        setpIva("00000000000");
        setIndirizzo("Via Romano Nicolo' 18");
    }

    public Negozio(String nome,String pIva,String indirizzo){
        setNome(nome);
        setIndirizzo(indirizzo);
        setpIva(pIva);
    }

    public void setNome(String nome) {
        if(nome.trim().isEmpty()){
            throw new IllegalArgumentException("Non è stato inserito nulla");
        }else
            this.nome = nome;
    }

    public void setIndirizzo(String indirizzo) {
        if(indirizzo.trim().isEmpty()){
            throw new IllegalArgumentException("Non è stato inserito nulla");
        }else
            this.indirizzo = indirizzo;
    }
```

```
    public void setpIva(String pIva) {
        if(pIva.trim().isEmpty()){
            throw new IllegalArgumentException("Non è stato inserito nulla");
        }else
            this.pIva = pIva;
    }

    public String getNome() {
        return nome;
    }

    public String getIndirizzo() {
        return indirizzo;
    }

    public String getpIva() {
        return pIva;
    }

    public boolean addSimToStorage(Sim sim){
           boolean check = false;
           if(checkIccid(sim.getIccId()) == null &&
checkNumero(sim.getNumero()) == null)
           {
               check = true;
               this.simBuy.add(sim);
           }
           return check;
    }

    public boolean addSimToSold(Sim sim){
        boolean check = false;
        if(checkIccid(sim.getIccId()) == null && checkNumero(sim.getNumero())
== null)
        {
            check = true;
            this.simSold.add(sim);
        }
        return check;
    }

    private Sim checkIccid(String iccid,LinkedList<Sim> lista){
        Sim find = null;
        for(int i=0; i<lista.size() && find==null;i++){
            Sim sim = lista.get(i);
            if(sim.getIccId().equals(iccid)){
                find = sim;
            }
        }
        return find;
    }

    private Sim checkNumero(String numero,LinkedList<Sim> lista){
```

```
        Sim find = null;
        for(int i=0; i<lista.size() && find==null;i++){
            Sim sim = lista.get(i);
            if(sim.getNumero().equals(numero)){
                find = sim;
            }
        }
        return find;
    }

    public String sellSim(String nome){
        String msg = "SIM venduta";
        Sim sim = this.simBuy.getLast();
        try{
            sim.setNome(nome);
        }catch (IllegalArgumentException e){
            msg = e.getMessage();
        }
        if(msg.equals("SIM venduta")){
            this.simBuy.remove(sim);
            this.simSold.add(sim);
        }
        return msg;
    }

    public boolean deactivateSim(String iccid){
        boolean deleted = false;
        Sim sim = checkIccid(iccid,simSold);
        if(sim!=null){
            sim.setActive(false);
            sim.setCredito(0);
            sim.setOperatore("NULL");
            sim.setNome("NULL");
            deleted = true;
            this.simSold.remove(sim);
        }
        return deleted;
    }

    public Sim findNumeroOperatore(String numero,String operatore){
        Sim find = null;
        for(int i=0; i<this.simSold.size() && find==null;i++){
            Sim sim =this.simSold.get(i);
            if(sim.getNumero().equals(numero) &&
sim.getOperatore().equals(operatore)){
                find = sim;
            }
        }
        return find;
    }

    public Sim findNumeroOperatoreIccid(String numero,String operatore,String
iccid){
        Sim find = null;
```

```
        for(int i=0; i<this.simSold.size() && find==null;i++){
            Sim sim =this.simSold.get(i);
            if(sim.getNumero().equals(numero) &&
sim.getOperatore().equals(operatore)){
                find = sim;
            }
        }
        return find;
    }

    public LinkedList<Sim> getSimToVisualize(boolean attive) {
        LinkedList<Sim> simsToDisplay = new LinkedList<>();
        if (attive) {
            for (Sim sim : simSold) {
                if (sim.isActive()) {
                    simsToDisplay.add(sim);
                }
            }
        } else {
            for (Sim sim : simBuy) {
                simsToDisplay.add(sim);
            }
        }
        return simsToDisplay;
    }


}




}
////////////////////////////////////////////////////////////
public class Sim implements Comparable<Sim> {
    private String nome;
    private String operatore;
    private String iccId;
    private String numero;
    private float credito;
    private int minuti;
    private boolean isActive;

    //COSTRUTTORI
    public Sim(){
        setActive(false);
        setOperatore("Tim");
        setCredito(5);
        setNome("NULL");
        setNumero("1235468790");
        setIccId("12345678901234567890");
        setMinuti(0);
    }
```

```
    public Sim(String nome,String operatore,String numero,String iccId){
        setOperatore("Postemobile");
        setActive(false);
        setCredito(0);
        setNome(nome);
        setNumero(numero);
        setIccId(iccId);
        setMinuti(0);
    }

    public Sim(Sim a){
        setActive(a.isActive());
        setCredito(a.getCredito());
        setNome(a.getNome());
        setNumero(a.getNumero());
        setIccId(a.getIccId());
        setMinuti(a.getMinuti());
    }
    //SETTER

    public void setOperatore(String operatore) {
        if (!(operatore.trim().isEmpty())) {
            this.operatore = operatore;
        } else
            this.operatore = "NULL";
    }

    public void setActive(boolean active) {
        this.isActive = active;
    }

    public void setCredito(float credito) {
        if(credito>=0){
            this.credito = credito;
        }
        else this.credito = 5;
    }

    public void setNome(String nome) {
        if (!(nome.trim().isEmpty())) {
            this.nome = nome;
        } else
            throw new IllegalArgumentException("Errore - nome intestatario
errato - SIM NON ATTIVA");
    }
    public void setNumero(String numero) {
        if (!(numero.trim().isEmpty())) {
            this.numero = numero;
        } else
            this.numero = "1234567890";
    }

    public void setMinuti(int minuti) {
```

```
    if(minuti>=0){
        this.minuti = minuti;
    }
    else this.minuti = 0;
}

public void setIccId(String iccId) {
    if(!(iccId.trim().isEmpty())){
        this.iccId = iccId;
    }else{
        this.iccId = "12354678901234568790";
    }
}

//GETTER
public float getCredito() {
    return credito;
}

public String getOperatore() { return operatore; }

public boolean isActive() {
    return isActive;
}

public int getMinuti() {
    return minuti;
}

public String getIccId() {
    return iccId;
}

public String getNome() {
    return nome;
}
public String getNumero() {
    return numero;
}
//METODI
public void addCredito(float credito){
    if(credito>0){
        this.credito=+credito;
    }
}
public int call(int minuti){
    int n;
    float creditoEnough;
    creditoEnough = this.credito * 0.32f;
    if(isActive()){
        if(creditoEnough>this.credito){
            n=1;
            this.credito=-creditoEnough;
        }else{
```

```
                n=2;
            }
        }else
            n=0;
        return n;
    }

   public String oreChiamata() {
       float ore = minuti / 60;
       float minutiCall = minuti % 60;
       String str = "Ore :" + ore + "Minuti :" + minutiCall;
       return str;
   }
    @Override
    public String toString() {
        return "Sim{" + "nome='" + nome + '\'' +
                ", iccId='" + iccId + '\'' +
                ", numero='" + numero + '\'' +
                ", credito=" + credito +
                ", minuti=" + minuti +
                ", isActive=" + isActive +
                '}';
    }

    @Override
    public int compareTo(Sim o) {
        int n;
        if(o.getIccId().compareTo(this.iccId)==0){
            n = 0;
        }
        if(o.getIccId().compareTo(this.iccId)>0){
            n = 1;
        }else
            n = -1;
        return n;
    }

}
//////////////////////////////////////////
import java.util.InputMismatchException;
import java.util.Scanner;
public class Input {
    public static int leggiInt (int vmin, int vmax,String msg){
        int n=0;
        boolean err;
        do{
            System.out.println(msg);
            Scanner input = new Scanner(System.in);
            err = false;
            try {
                n = input.nextInt();
                input.nextLine();
            }catch (InputMismatchException e){
                System.out.println("Ciò che hai inserito non è un numero");
```

```
                err = true;
            }
            if(n<vmin || n>vmax)
                System.out.println("Errore");
        }while((n<vmin || n>vmax) || err);
        return n;
    }

    public static char leggiChar (char vmin, char vmax,String msg){
        char c='.';
        boolean err;
        do{
            System.out.println(msg);
            Scanner input = new Scanner(System.in);
            err = false;
            try {
                c = input.nextLine().charAt(0);
                input.nextLine();
            }catch (InputMismatchException e){
                System.out.println("Ciò che hai inserito non è un carattere");
                err = true;
            }
            if(c<vmin || c>vmax)
                System.out.println("Errore");
        }while((c<vmin || c>vmax) || err);
        return c;
    }

    public static float leggiFloat (float vmin, float vmax,String msg){
        float n=0;
        boolean err;
        do{
            System.out.println(msg);
            Scanner input = new Scanner(System.in);
            err = false;
            try {
                n = input.nextFloat();
                input.nextLine();
            }catch (InputMismatchException e){
                System.out.println("Ciò che hai inserito non è un numero
decimale");
                err = true;
            }
            if(n<vmin || n>vmax)
                System.out.println("Errore");
        }while((n<vmin || n>vmax) || err);
        return n;
    }

    public static Double leggiDouble (double vmin, double vmax,String msg){
        double n=0;
        boolean err;
        do{
            System.out.println(msg);
```

```
            Scanner input = new Scanner(System.in);
            err = false;
            try {
                n = input.nextFloat();
                input.nextLine();
            }catch (InputMismatchException e){
                System.out.println("Ciò che hai inserito non è un numero
decimale");
                err = true;
            }
            if(n<vmin || n>vmax)
                System.out.println("Errore");
        }while((n<vmin || n>vmax) || err);
        return n;
    }

    public static String leggiStr(String msg) {
        String s="";
        boolean err;
        do{
            System.out.println(msg);
            Scanner input = new Scanner(System.in);
            err = false;
            try {
                s = input.nextLine();
                input.nextLine();
            }catch (InputMismatchException e){
                System.out.println("Ciò che hai inserito non è una Stringa");
                err = true;
            }
            if(s.trim().isEmpty())
                System.out.println("Errore");
        }while(s.trim().isEmpty() || err);
        return s;
    }

}
```