```
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.NoSuchElementException;
@SuppressWarnings("all")
public class Azienda {
    private MyList<Dipendente> dipendenti;
    private String nome;
    private String responsabile;
    private String indirizzo;
    private String sitoWeb;
    private String email;

    public Azienda() {
        this.dipendenti = new MyList<>();
    }

    public String getNome() {
        return nome;
    }

    public String getIndirizzo() {
        return indirizzo;
    }

    public String getEmail() {
        return email;
    }

    public String getResponsabile() {
        return responsabile;
    }

    public String getSitoWeb() {
        return sitoWeb;
    }

    public void setNome(String nome) {
        if(!nome.trim().isEmpty())
            this.nome = nome;
        else
            throw new IllegalArgumentException("Nome non valido");
    }

    public void setIndirizzo(String indirizzo) {
        if(!indirizzo.trim().isEmpty())
            this.indirizzo = indirizzo;
        else
            throw new IllegalArgumentException("Indirizzo non valido");
    }

    public void setEmail(String email) {
        if(!email.trim().isEmpty())
            this.email = email;
```

```
    else
        throw new IllegalArgumentException("Email non valida");
}

public void setResponsabile(String responsabile) {
    if(!responsabile.trim().isEmpty())
        this.responsabile = responsabile;
    else
        throw new IllegalArgumentException("Responsabile non valido");
}

public void setSitoWeb(String sitoWeb) {
    if(!sitoWeb.trim().isEmpty())
        this.sitoWeb = sitoWeb;
    else
        throw new IllegalArgumentException("Sito web non valido");
}

public void addImpiegato(Impiegato imp) {
    dipendenti.add(imp);
}

public void addOperario(Operaio op) {
    dipendenti.add(op);
}

public String stampaDipendenti(){
    String s="";
    Dipendente current = dipendenti.getFirst();
    while (current != null) {
        s += current.stampaDati()+"\n";
    }
    return s;
}

public String stampaDipendenti(double salario){
    String s="";
    Dipendente current = dipendenti.getFirst();
    while (current != null) {
        if(current.calcolaPaga() > salario)
            s += current.stampaDati()+"\n";
    }
    return s;
}

public void licenziaDipendente(int mat){
    int pos=-1;
    Dipendente current = dipendenti.getFirst();
    while (current != null && pos==-1) {
        if(current.getMat() == mat) {
            dipendenti.remove(current);
            pos=1;
        }
    }
```

```
    }

    public double calcolaStipendi(){
        double tot=0;
        Dipendente current = dipendenti.getFirst();
        while (current != null) {
            tot += current.calcolaPaga();
        }
        return tot;
    }

    public void esportaOrdinato(String filename, String t) throws IOException {
        Dipendente d;
        PrintWriter fout;
        StringBuilder s = new StringBuilder();
        for (int k = 0; k < dipendenti.size(); k++) {
            d = dipendenti.get(k);
            if (d.getClass().toString().contains("Impiegato") &&
(t.compareTo("Impiegato") == 0 || t.compareTo("impiegato")==0))  {
                s.append(d.toCsv());
                s.append('\n');
            }else if (d.getClass().toString().contains("Operaio") &&
(t.compareTo("Operaio") == 0 || t.compareTo("operaio")==0)) {
                s.append(d.toCsv());
                s.append('\n');
            }
            if(!s.isEmpty()){
                fout = new PrintWriter(new FileWriter(filename));
                fout.print(s);
                fout.close();
            }
            else
                throw new NoSuchElementException("Nessun dipendente di tale tipo
trovato");


        }
    }
}
//////////////////////////////////////////
public abstract class Dipendente implements Comparable<Dipendente> , FileCSV{
    protected int mat;
    protected String nome,cog,data;

    public abstract String stampaDati();
    public abstract double calcolaPaga();

    public Dipendente(int mat, String nome, String cog, String data) throws
IllegalArgumentException {
        setMat(mat);
        setNome(nome);
        setCog(cog);
        setData(data);
    }
```

```
public Dipendente() {
    this.mat=0;
    this.nome="";
    this.cog="";
    this.data="";
}
public void setMat(int mat) {
    if(mat >= 0)
        this.mat = mat;
    else
        throw new IllegalArgumentException("Matricola non valida");
}

public void setNome(String nome) {
    if(nome != null)
        this.nome = nome;
    else
        throw new IllegalArgumentException("Nome non valido");
}

public void setCog(String cog) {
    if(cog != null)
        this.cog = cog;
    else
        throw new IllegalArgumentException("Cognome non valido");
}

public void setData(String data) {
    if(data != null)
        this.data = data;
    else
        throw new IllegalArgumentException("Data di nascita non valida");
}

public String getNome() {
    return nome;
}

public int getMat() {
    return mat;
}

public String getCog() {
    return cog;
}

public String getData() {
    return data;
}

@Override
public int compareTo(Dipendente o) {
    int n;
```

```
        if(this.mat < o.mat)
            n = -1;
        else
            if(this.mat > o.mat)
                n = 1;
            else
                n = 0;
        return n;
    }
}
//////////////////////////////////////////
public interface FileCSV {
    public void fromCsv(String csv);
    public String toCsv();
}
import java.util.StringTokenizer;

//per ogni ora vengono decurtati 20€ dallo stipendio
public class Impiegato extends Dipendente implements FileCSV{
    private double salario;
    private int orePermesso;
    private final double dec = 20d;


    public Impiegato(int mat, String nome, String cog, String data, double
salario, int orePermesso) throws IllegalArgumentException {
        super(mat, nome, cog, data);
        setSalario(salario);
        setOrePermesso(orePermesso);
    }

    @Override
    public String stampaDati() {
        return "Matricola: "+mat+" Nome: "+nome+" Cognome: "+cog+" Salario:
"+salario;
    }

    @Override
    public double calcolaPaga() {
        double paga = salario;
        if(orePermesso > 0)
            paga = salario - orePermesso*dec;
        return paga;
    }

    public void setSalario(double salario) {
        if(salario >= 0)
            this.salario = salario;
        else
            throw new IllegalArgumentException("Salario non valido");
    }

    public void setOrePermesso(int orePermesso) {
        if(orePermesso >= 0)
```

```
            this.orePermesso = orePermesso;
        else
            throw new IllegalArgumentException("Ore permesso non valide");
    }

    public double getSalario() {
        return salario;
    }

    public int getOrePermesso() {
        return orePermesso;
    }

    @Override
    public void fromCsv(String csv) {
        StringTokenizer strtok = new StringTokenizer(csv, ",");
        mat = Integer.parseInt(strtok.nextToken());
        nome = strtok.nextToken();
        cog = strtok.nextToken();
        data = strtok.nextToken();
        salario = Double.parseDouble(strtok.nextToken());
        orePermesso = Integer.parseInt(strtok.nextToken());
    }

    @Override
    public String toCsv() {
        return mat+";"+nome+";"+cog+";"+data+";"+salario+";"+orePermesso;
    }
}
////////////////////////////////
import java.io.IOException;
import java.io.RandomAccessFile;
import java.util.InputMismatchException;
import java.util.Scanner;
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;
    }

    public static void write(RandomAccessFile raf, String str, int len) throws
IOException {
        StringBuffer buf = new StringBuffer(str);
        buf.setLength(len);
        raf.writeChars(buf.toString());
    }
    public static String readString(RandomAccessFile raf, int len) throws
IOException{
        char str[] = new char[len];
        for(int k = 0; k < len; k++)
            str[k] = raf.readChar();

        String s = new String(str);
        return s.trim();
    }

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

import java.io.IOException;

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

```
        Azienda azienda = new Azienda();
        char sc;
        do{
            menu();
            sc = Input.leggiChar('a','g',"Scelta: ");
            switch (sc){
                case 'a': {
                    addDipendente(azienda);
                    break;
                }
                case 'b': {
                    stampaDipendenti(azienda);
                    break;
                }
                case 'c': {
                    stampaPerSalario(azienda);
                    break;
                }
                case 'd': {
                    licenziaDipendente(azienda);
                    break;
                }
                case 'e': {
                    stampaTotaleStipendi(azienda);
                    break;
                }
                case 'f': {
                    esportaOrdinato(azienda);
                    break;
                }
            }
        }while(sc!='g');
    }

    public static void menu(){
        System.out.println("1. Inserimento di un nuovo dipendente");
        System.out.println("2. Stampa dell’elenco dei dipendenti dell’azienda");
        System.out.println("3. Stampa di tutti di dipendenti aventi uno
stipendio mensile inferiore a 1350€");
        System.out.println("4. Licenziamento di un dipendete, data la
matricola");
        System.out.println("5. Stampa del totale stipendi che l’azienda dovrà
versare ai dipendenti");
        System.out.println("6. Esportazione su file csv di tutti i dipendenti di
un tipo indicato ordinati per cognome (Operaio/Impiegato)");
        System.out.println("7. Esci");
    }

    public static void addDipendente(Azienda azienda){
        char sc;
        do{
            System.out.println("a. Aggiungi Impiegato");
            System.out.println("b. Aggiungi Operaio");
            System.out.println("c. Esci");
```

```
            sc = Input.leggiChar('a','c',"Scelta: ");
            switch (sc){
                case '1': {
                    int mat=Integer.parseInt(Input.leggiStr("Matricola: "));
                    String nome=Input.leggiStr("Nome: ");
                    String cog=Input.leggiStr("Cognome: ");
                    String data=Input.leggiStr("Data di nascita: ");
                    double
salario=Input.leggiDouble(0,Double.MAX_VALUE,"Salario: ");
                    int orePermesso=Input.leggiInt(0,Integer.MAX_VALUE,"Ore
permesso: ");
                    try {
                        Impiegato imp = new
Impiegato(mat,nome,cog,data,salario,orePermesso);
                        azienda.addImpiegato(imp);
                    } catch (IllegalArgumentException e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                }
                case '2': {
                    int mat=Integer.parseInt(Input.leggiStr("Matricola: "));
                    String nome=Input.leggiStr("Nome: ");
                    String cog=Input.leggiStr("Cognome: ");
                    String data=Input.leggiStr("Data di nascita: ");
                    double pagaOra=Input.leggiDouble(0,Double.MAX_VALUE,"Paga
oraria: ");
                    int orePermesso=Input.leggiInt(0,Integer.MAX_VALUE,"Ore
permesso: ");
                    int oreStraordinario=Input.leggiInt(0,Integer.MAX_VALUE,"Ore
straordinario: ");
                    try {
                        Operaio op = new
Operaio(mat,nome,cog,data,pagaOra,orePermesso,oreStraordinario);
                        azienda.addOperario(op);
                    } catch (IllegalArgumentException e) {
                        System.out.println(e.getMessage());
                    }
                    break;
                }
            }
        }while(sc!='c');
    }
    public static void stampaDipendenti(Azienda azienda){
        System.out.println(azienda.stampaDipendenti());
    }

    public static void stampaPerSalario(Azienda azienda){
        Double salario = Input.leggiDouble(0,Double.MAX_VALUE,"Salario: ");
        System.out.println(azienda.stampaDipendenti(salario));
    }

    public static void licenziaDipendente(Azienda azienda){
        int mat = Input.leggiInt(0,Integer.MAX_VALUE,"Matricola: ");
```

```
        azienda.licenziaDipendente(mat);
    }

    public static void stampaTotaleStipendi(Azienda azienda){
        System.out.println(azienda.calcolaStipendi());
    }

    public static void esportaOrdinato(Azienda azienda){
        char sc;
        do{
            System.out.println("a. Esporta Impiegati");
            System.out.println("b. Esporta Operai");
            System.out.println("c. Esci");
            sc = Input.leggiChar('a','c',"Scelta: ");
            switch (sc){
                case 'a': {
                    try{azienda.esportaOrdinato("impiegati.csv","Impiegato");
                    }catch (IOException e){
                        System.out.println(e.getMessage());
                    }
                    break;
                }
                case 'b': {
                    try {
                        azienda.esportaOrdinato("operai.csv","Operaio");
                    }catch (IOException e){
                        System.out.println(e.getMessage());
                    }
                    break;
                }
            }
        }while(sc!='c');
    }
}

//////////////////////////////////////

import java.util.NoSuchElementException;
@SuppressWarnings("all")
public class MyList<T extends Comparable<T> & FileCSV> {
    private Nodo<T> first;

    public MyList() {
        this.first = null;
    }

    public void add(T obj)
    {
        if(obj != null){
            Nodo<T> toAdd = new Nodo<>(obj);
            if(first== null)
                this.first = toAdd;
            else{
                Nodo<T> aus = first;
```

```
            Nodo<T> prec = null;
            boolean tro = false;
            while(aus != null && !tro){
                if (obj.compareTo(aus.getDato()) < 0)
                    tro = true;
                else{
                    prec = aus;
                    aus = aus.getNext();
                }
            }
            if(prec == null){
                Nodo<T> exFirst = first;
                first = toAdd;
                toAdd.setNext(exFirst);
            }
            else if(aus == null){
                prec.setNext(toAdd);
                toAdd.setNext(null);
            }
            else{
                toAdd.setNext(aus);
                prec.setNext(toAdd);
            }
        }
    }
    else
        throw new NullPointerException("Oggetto null non consentito");
}

public boolean contains(T obj) {
    boolean exists = false;
    Nodo<T> current = first;
    while (current != null) {
        if (current.getDato().equals(obj)) {
            exists = true;
        }
        current = current.getNext();
    }
    return exists;
}

public int size() {
    int size = 0;
    Nodo<T> current = first;
    while (current != null) {
        size++;
        current = current.getNext();
    }
    return size;
}
public void addFirst(T obj) {
    Nodo<T> newNode = new Nodo<>(obj);
    if (first != null) {
        newNode.setNext(first);
```

```
    }
    first = newNode;
}

public void addLast(T obj) {
    add(obj);
}

public void clear(){
    first = null;
}

public T element() {
    if (first == null) {
        throw new NoSuchElementException("List is empty");
    }
    return first.getDato();
}

public T getFirst() {
    if (first == null) {
        throw new NoSuchElementException("List is empty");
    }
    return first.getDato();
}

public T getLast() {
    if (first == null) {
        throw new NoSuchElementException("List is empty");
    }
    Nodo<T> current = first;
    while (current.getNext() != null) {
        current = current.getNext();
    }
    return current.getDato();
}

public T remove(){
    if (first == null) {
        throw new NoSuchElementException("List is empty");
    }
    T removedElement = first.getDato();
    first = first.getNext();
    return removedElement;
}

public T removeLast(){
    if (first == null) {
        throw new NoSuchElementException("List is empty");
    }
    if (first.getNext() == null) {
        T removedElement = first.getDato();
        first = null;
        return removedElement;
```

```
        }
        Nodo<T> current = first;
        while (current.getNext().getNext() != null) {
            current = current.getNext();
        }
        T removedElement = (T) current.getNext().getDato();
        current.setNext(null);
        return removedElement;
    }

    public void remove(T dato){
        if (first == null) {
            throw new NoSuchElementException("List is empty");
        }
        if (first.getDato().equals(dato)) {
            first = first.getNext();
        } else {
            Nodo<T> current = first;
            while (current.getNext() != null &&
!current.getNext().getDato().equals(dato)) {
                current = current.getNext();
            }
            if (current.getNext() == null) {
                throw new NoSuchElementException("Element not found");
            }
            current.setNext(current.getNext().getNext());
        }
    }

    public T get(int index) {
        if (index < 0) {
            throw new IndexOutOfBoundsException("Index is negative");
        }
        Nodo<T> current = first;
        for (int i = 0; i < index; i++) {
            if (current == null) {
                throw new IndexOutOfBoundsException("Index is greater than list
size");
            }
            current = current.getNext();
        }
        if (current == null) {
            throw new IndexOutOfBoundsException("Index is greater than list
size");
        }
        return current.getDato();
    }
}

////////////////////////////////////////////////

public class Nodo <T> {
    private T dato;
    private Nodo next;
```

```
    public Nodo(T obj) {
        setDato(obj);
        setNext(null);
    }

    private void setDato(T dato) {
        this.dato = dato;
    }

    public Nodo getNext() {
        return next;
    }

    public T getDato() {
        return dato;
    }

    public void setNext(Nodo next) {
        this.next = next;
    }


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

import java.util.StringTokenizer;

public class Operaio extends Dipendente implements FileCSV{
    private double pagaOra;
    private int orePermesso;
    private int oreStraordinario;

    public Operaio(int mat, String nome, String cog, String data, double
pagaOra, int orePermesso, int oreStraordinario) throws IllegalArgumentException
{
        super(mat, nome, cog, data);
        setPagaOra(pagaOra);
        setOrePermesso(orePermesso);
        setOreStraordinario(oreStraordinario);
    }

    @Override
    public String stampaDati() {
        return "Matricola: "+mat+" Nome: "+nome+" Cognome: "+cog+" Ore lavorate:
"+(160-orePermesso+oreStraordinario);
    }

    @Override
    public double calcolaPaga() {
        double paga;
        if(orePermesso == oreStraordinario)
            paga = 160*pagaOra;
        else
```

```
        if(orePermesso > oreStraordinario)
            paga = 160*pagaOra - orePermesso*pagaOra;
        else
            paga = 160*pagaOra + (oreStraordinario-orePermesso)*pagaOra*1.3;
    return paga;
}

public void setOrePermesso(int orePermesso) {
    if(orePermesso >= 0)
        this.orePermesso = orePermesso;
    else
        throw new IllegalArgumentException("Ore permesso non valide");
}

public void setOreStraordinario(int oreStraordinario) {
    if(oreStraordinario >= 0)
        this.oreStraordinario = oreStraordinario;
    else
        throw new IllegalArgumentException("Ore straordinario non valide");
}

public void setPagaOra(double pagaOra) {
    if(pagaOra >= 0)
        this.pagaOra = pagaOra;
    else
        throw new IllegalArgumentException("Paga oraria non valida");
}

public double getPagaOra() {
    return pagaOra;
}

public int getOrePermesso() {
    return orePermesso;
}

public int getOreStraordinario() {
    return oreStraordinario;
}

@Override
public void fromCsv(String csv) {
    StringTokenizer strtok = new StringTokenizer(csv, ",");
    mat = Integer.parseInt(strtok.nextToken());
    nome = strtok.nextToken();
    cog = strtok.nextToken();
    data = strtok.nextToken();
    pagaOra = Double.parseDouble(strtok.nextToken());
    orePermesso = Integer.parseInt(strtok.nextToken());
    oreStraordinario = Integer.parseInt(strtok.nextToken());
}

@Override
public String toCsv() {
```

```
        return
mat+";"+nome+";"+cog+";"+data+";"+pagaOra+";"+orePermesso+";"+oreStraordinario;
    }
}
```