

My Project

Generated by Doxygen 1.9.8

1 Hierarchical Index	1
1.1 Class Hierarchy	1
2 Class Index	3
2.1 Class List	3
3 File Index	5
3.1 File List	5
4 Class Documentation	7
4.1 Angajat Class Reference	7
4.2 AngajatFactory Class Reference	7
4.3 CerereReparatie Class Reference	7
4.4 CNPValidator Class Reference	8
4.5 Data Class Reference	8
4.6 Electrocasnic Class Reference	9
4.7 ElectrocasnicFactory Class Reference	9
4.8 Frigider Class Reference	10
4.8.1 Member Function Documentation	10
4.8.1.1 afisare()	10
4.8.1.2 cloneaza()	11
4.9 MasinaDeSpalat Class Reference	11
4.9.1 Member Function Documentation	12
4.9.1.1 afisare()	12
4.9.1.2 cloneaza()	12
4.10 Receptioner Class Reference	12
4.10.1 Member Function Documentation	13
4.10.1.1 afisare()	13
4.10.1.2 calculeazaSalariu()	13
4.10.1.3 getTipAngajat()	13
4.11 ServiceManager Class Reference	14
4.12 Supervizor Class Reference	15
4.12.1 Member Function Documentation	16
4.12.1.1 afisare()	16
4.12.1.2 calculeazaSalariu()	16
4.12.1.3 getTipAngajat()	16
4.13 Tehnician Class Reference	16
4.13.1 Member Function Documentation	17
4.13.1.1 afisare()	17
4.13.1.2 calculeazaSalariu()	17
4.13.1.3 getTipAngajat()	18
4.14 TV Class Reference	18
4.14.1 Member Function Documentation	19

4.14.1.1 afisare()	19
4.14.1.2 cloneaza()	19
5 File Documentation	21
5.1 Angajat.h	21
5.2 AngajatFactory.h	22
5.3 CerereReparatie.h	22
5.4 CNPValidator.h	23
5.5 Data.h	23
5.6 Electrocasnic.h	24
5.7 ElectrocasnicFactory.h	24
5.8 Frigider.h	24
5.9 MasinaDeSpalat.h	25
5.10 Receptioner.h	25
5.11 ServiceManager.h	25
5.12 Supervizor.h	27
5.13 Tehnician.h	27
5.14 TV.h	28
Index	29

Chapter 1

Hierarchical Index

1.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Angajat	7
Receptioner	12
Supervizor	15
Tehnician	16
AngajatFactory	7
CerereReparatie	7
CNPValidator	8
Data	8
Electrocasnic	9
Frigider	10
MasinaDeSpalat	11
TV	18
ElectrocasnicFactory	9
ServiceManager	14

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Angajat	7
AngajatFactory	7
CerereReparatie	7
CNPValidator	8
Data	8
Electrocasnic	9
ElectrocasnicFactory	9
Frigider	10
MasinaDeSpalat	11
Receptioner	12
ServiceManager	14
Supervizor	15
Tehnician	16
TV	18

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Angajat.h	21
AngajatFactory.h	22
CerereReparatie.h	22
CNPValidator.h	23
Data.h	23
Electrocasnic.h	24
ElectrocasnicFactory.h	24
Frigider.h	24
MasinaDeSpalat.h	25
Receptioner.h	25
ServiceManager.h	25
Supervizor.h	27
Tehnician.h	27
TV.h	28

Chapter 4

Class Documentation

4.1 Angajat Class Reference

Inheritance diagram for Angajat:

4.2 AngajatFactory Class Reference

Static Public Member Functions

- static std::unique_ptr< [Angajat](#) > **creeazaAngajat** (const std::string &tip, const std::string &nume, const std::string &prenume, const std::string &cnp, const [Data](#) &dataAngajarii, const std::string &orasDomiciliu, const std::map< std::string, std::set< std::string > > &specializari={})
- static std::unique_ptr< [Angajat](#) > **creeazaDinCSV** (const std::string &linie)

The documentation for this class was generated from the following files:

- [AngajatFactory.h](#)
- [AngajatFactory.cpp](#)

4.3 CerereReparatie Class Reference

Public Member Functions

- **CerereReparatie** (std::unique_ptr< [Electrocasnic](#) > _aparat, const time_t _timestamp, int _nivelComplexitate)
- **CerereReparatie** (const [CerereReparatie](#) &)
- **CerereReparatie** & **operator=** (const [CerereReparatie](#) &)
- int **getId** () const
- int **getNivelComplexitate** () const
- const [Electrocasnic](#) * **getAparat** () const
- time_t **getTimeStamp** () const
- int **getDurataEstimata** () const
- int **getDurataRamasa** () const

- double **getPretReparatie** () const
- StareCerere **getStare** () const
- int **getIdTehnician** () const
- void **setStare** (const StareCerere)
- void **setIdTehnician** (int)
- void **proceseaza** ()
- bool **esteFinalizata** () const
- bool **esteValida** () const
- void **afisare** (std::ostream &) const
- std::string **getTimeStampString** () const
- bool **operator<** (const CerereReparatie &) const

The documentation for this class was generated from the following files:

- CerereReparatie.h
- CerereReparatie.cpp

4.4 CNPValidator Class Reference

Static Public Member Functions

- static bool **esteValid** (const std::string &cnp)
- static Data **getDataNasterii** (const std::string &cnp)
- static char **getSex** (const std::string &cnp)

The documentation for this class was generated from the following files:

- CNPValidator.h
- CNPValidator.cpp

4.5 Data Class Reference

Public Member Functions

- Data (int, int, int)
- Data (const std::string &)
- bool **esteValida** () const
- int **getVarsta** () const
- int **getZi** () const
- int **getLuna** () const
- int **getAn** () const
- bool **operator<** (const Data &)
- bool **operator>** (const Data &)
- bool **operator==** (const Data &)

Static Public Member Functions

- static Data **dataCurenta** ()

Friends

- std::ostream & **operator<<** (std::ostream &, const Data &)

The documentation for this class was generated from the following files:

- Data.h
- Data.cpp

4.6 Electrocasicnic Class Reference

Inheritance diagram for Electrocasicnic:

Public Member Functions

- Electrocasicnic (const std::string &, const std::string &, const std::string &, int, double)
- std::string **getTip** () const
- std::string **getMarca** () const
- std::string **getModel** () const
- int **getAnFabricatie** () const
- double **getPretCatalog** () const
- int **getVechime** () const
- virtual void **afisare** (std::ostream &) const =0
- virtual std::unique_ptr< Electrocasicnic > **cloneaza** () const =0
- bool **operator==** (const Electrocasicnic &altul) const

Protected Attributes

- std::string **tip**
- std::string **marca**
- std::string **model**
- int **anFabricatie**
- double **pretCatalog**

The documentation for this class was generated from the following files:

- Electrocasicnic.h
- Electrocasicnic.cpp

4.7 ElectrocasicnicFactory Class Reference

Static Public Member Functions

- static std::unique_ptr< Electrocasicnic > **creeazaElectrocasicnic** (const std::string &tip, const std::string &marca, const std::string &model, int anFabricatie, double pretCatalog, const std::map< std::string, std::string > &dateSpecifice)
- static std::map< std::string, std::string > **parseazaDateSpecifice** (const std::string &linie)

The documentation for this class was generated from the following files:

- ElectrocasicnicFactory.h
- ElectrocasicnicFactory.cpp

4.8 Frigider Class Reference

Inheritance diagram for Frigider:

Collaboration diagram for Frigider:

Public Member Functions

- **Frigider** (const std::string &, const std::string &, int, double, bool)
- bool **getAreCongelator** () const
- void **afisare** (std::ostream &) const override
- std::unique_ptr< [Electrocasnic](#) > **cloneaza** () const override

Public Member Functions inherited from [Electrocasnic](#)

- **Electrocasnic** (const std::string &, const std::string &, const std::string &, int, double)
- std::string **getTip** () const
- std::string **getMarca** () const
- std::string **getModel** () const
- int **getAnFabricatie** () const
- double **getPretCatalog** () const
- int **getVechime** () const
- bool **operator==** (const [Electrocasnic](#) &altul) const

Additional Inherited Members

Protected Attributes inherited from [Electrocasnic](#)

- std::string **tip**
- std::string **marca**
- std::string **model**
- int **anFabricatie**
- double **pretCatalog**

4.8.1 Member Function Documentation

4.8.1.1 **afisare()**

```
void Frigider::afisare (
    std::ostream & dev ) const [override], [virtual]
```

Implements [Electrocasnic](#).

4.8.1.2 cloneaza()

```
unique_ptr< Electrocasic > Frigider::cloneaza ( ) const [override], [virtual]
```

Implements [Electrocasic](#).

The documentation for this class was generated from the following files:

- Frigider.h
- Frigider.cpp

4.9 MasinaDeSpalat Class Reference

Inheritance diagram for MasinaDeSpalat:

Collaboration diagram for MasinaDeSpalat:

Public Member Functions

- **MasinaDeSpalat** (const std::string &, const std::string &, int, double, double)
- double **getCapacitate** () const
- void **afisare** (std::ostream &) const override
- std::unique_ptr< Electrocasic > **cloneaza** () const override

Public Member Functions inherited from [Electrocasic](#)

- **Electrocasic** (const std::string &, const std::string &, const std::string &, int, double)
- std::string **getTip** () const
- std::string **getMarca** () const
- std::string **getModel** () const
- int **getAnFabricatie** () const
- double **getPretCatalog** () const
- int **getVechime** () const
- bool **operator==** (const [Electrocasic](#) &altul) const

Additional Inherited Members

Protected Attributes inherited from [Electrocasic](#)

- std::string **tip**
- std::string **marca**
- std::string **model**
- int **anFabricatie**
- double **pretCatalog**

4.9.1 Member Function Documentation

4.9.1.1 afisare()

```
void MasinaDeSpalat::afisare (
    std::ostream & ) const [override], [virtual]
```

Implements [Electrocasnic](#).

4.9.1.2 cloneaza()

```
unique_ptr< Electrocasnic > MasinaDeSpalat::cloneaza () const [override], [virtual]
```

Implements [Electrocasnic](#).

The documentation for this class was generated from the following files:

- [MasinaDeSpalat.h](#)
- [MasinaDeSpalat.cpp](#)

4.10 Receptioner Class Reference

Inheritance diagram for Receptioner:

Collaboration diagram for Receptioner:

Public Member Functions

- **Receptioner** (const std::string &, const std::string &, const std::string &, const [Data](#) &, const std::string &)
- void **adaugaCerere** (int)
- const std::vector< int > & **getCereri** () const
- double **calculeazaSalariu** () const override
- void **afisare** (std::ostream &) const override
- std::string **getTipAngajat** () const override

Public Member Functions inherited from [Angajat](#)

- **Angajat** (const std::string &, const std::string &, const std::string &, const [Data](#) &, const std::string &)
- int **getId** () const
- std::string **getNume** () const
- std::string **getPrenume** () const
- std::string **getCNP** () const
- [Data](#) **getDataAngajarii** () const
- std::string **getOrasDomiciliu** () const
- void **setNume** (const std::string &)
- void **setPrenume** (const std::string &)
- void **validareNume** (const std::string &, const std::string &) const
- double **calculeazaBonusFidelitate** () const
- bool **primestePrimaTransport** () const

Additional Inherited Members

Protected Attributes inherited from [Angajat](#)

- const int **id**
- std::string **nume**
- std::string **prenume**
- std::string **cnp**
- [Data](#) **dataAngajarii**
- std::string **orasDomiciliu**

Static Protected Attributes inherited from [Angajat](#)

- static int **nextID** =1000
- static constexpr double **SALARIU_BAZA** =4000.0
- static constexpr double **BONUS_DE_FIDELITARE_PROCENT** =5.0
- static constexpr int **ANI_FIDELITATE** =3
- static constexpr double **PRIMA_TRANS** =400.0

4.10.1 Member Function Documentation

4.10.1.1 [afisare\(\)](#)

```
void Receptioner::afisare (
    std::ostream & ) const [override], [virtual]
```

Reimplemented from [Angajat](#).

4.10.1.2 [calculeazaSalariu\(\)](#)

```
double Receptioner::calculeazaSalariu ( ) const [override], [virtual]
```

Implements [Angajat](#).

4.10.1.3 [getTipAngajat\(\)](#)

```
string Receptioner::getTipAngajat ( ) const [override], [virtual]
```

Implements [Angajat](#).

The documentation for this class was generated from the following files:

- Receptioner.h
- Receptioner.cpp

4.11 ServiceManager Class Reference

Public Member Functions

- **ServiceManager** (const **ServiceManager** &)=delete
- **ServiceManager** & **operator=** (const **ServiceManager** &)=delete
- bool **arePersonalMinim** () const
- int **getNumarTehnicieni** () const
- int **getNumarReceptioneri** () const
- int **getNumarSupervizori** () const
- void **adaugaAngajat** (std::unique_ptr<**Angajat**> angajat)
- bool **stergeAngajat** (const std::string &cnp)
- bool **modificaNumeAngajat** (const std::string &cnp, const std::string &numeNou, const std::string &prenumeNou)
- **Angajat** * **cautaAngajatDupaCNP** (const std::string &cnp)
- **Angajat** * **cautaAngajatDupaID** (int id)
- void **afiseazaAngajat** (const std::string &cnp) const
- void **afiseazaTotiAngajatii** () const
- void **incarcaAngajatiDinCSV** (const std::string &numeFisier)
- void **adaugaModelInCatalog** (std::unique_ptr<**Electrocasnic**> electrocasnic)
- bool **stergeModelInCatalog** (const std::string &tip, const std::string &marca, const std::string &model)
- bool **poateFiReparat** (const std::string &tip, const std::string &marca, const std::string &model) const
- const **Electrocasnic** * **getDetaliiModel** (const std::string &tip, const std::string &marca, const std::string &model) const
- void **afiseazaCatalogComplet** () const
- void **afiseazaAparateNereparabile** () const
- void **incarcaCatalogDinCSV** (const std::string &numeFisier)
- void **primesteCerere** (std::unique_ptr<**CerereReparatie**> cerere)
- void **incarcaCereriDinCSV** (const std::string &numeFisier)
- bool **esteCerereValida** (const **CerereReparatie** &cerere)
- **Tehnician** * **gasesteTehnicianPotrivit** (const **CerereReparatie** &cerere)
- void **alocaCerere** (**CerereReparatie** *cerere, **Tehnician** *tehnician)
- void **procesareCereriInAsteptare** ()
- void **startSimulare** ()
- void **executeazaTic** ()
- void **afiseazaStatusTic** () const
- void **stopSimulare** ()
- void **finalizeazaCerere** (**CerereReparatie** *cerere)
- void **genereazaRaportTop3Salarii** (const std::string &numeFisier) const
- void **genereazaRaportCeaMaiLungaReparatie** (const std::string &numeFisier) const
- void **genereazaRaportCereriInAsteptare** (const std::string &numeFisier) const
- void **afiseazaStatistici** () const
- void **reset** ()

Static Public Member Functions

- static **ServiceManager** * **getInstance** ()
- static void **deleteInstance** ()

The documentation for this class was generated from the following files:

- ServiceManager.h
- ServiceManager.cpp

4.12 Supervizor Class Reference

Inheritance diagram for Supervizor:

Collaboration diagram for Supervizor:

Public Member Functions

- **Supervizor** (const std::string &, const std::string &, const std::string &, const [Data](#) &, const std::string &)
- double **calculeazaSalariu** () const override
- void **afisare** (std::ostream &) const override
- std::string **getTipAngajat** () const override

Public Member Functions inherited from [Angajat](#)

- **Angajat** (const std::string &, const std::string &, const std::string &, const [Data](#) &, const std::string &)
- int **getId** () const
- std::string **getNum** () const
- std::string **getPrenume** () const
- std::string **getCNP** () const
- [Data](#) **getDataAngajarii** () const
- std::string **getOrasDomiciliu** () const
- void **setNum** (const std::string &)
- void **setPrenume** (const std::string &)
- void **validareNum** (const std::string &, const std::string &) const
- double **calculeazaBonusFidelitate** () const
- bool **primestePrimaTransport** () const

Additional Inherited Members

Protected Attributes inherited from [Angajat](#)

- const int **id**
- std::string **nume**
- std::string **prenume**
- std::string **cnp**
- [Data](#) **dataAngajarii**
- std::string **orasDomiciliu**

Static Protected Attributes inherited from [Angajat](#)

- static int **nextID** =1000
- static constexpr double **SALARIU_BAZA** =4000.0
- static constexpr double **BONUS_DE_FIDELITARE_PROCENT** =5.0
- static constexpr int **ANI_FIDELITATE** =3
- static constexpr double **PRIMA_TRANS** =400.0

4.12.1 Member Function Documentation

4.12.1.1 afisare()

```
void Supervizor::afisare (
    std::ostream & ) const [override], [virtual]
```

Reimplemented from [Angajat](#).

4.12.1.2 calculeazaSalariu()

```
double Supervizor::calculeazaSalariu ( ) const [override], [virtual]
```

Implements [Angajat](#).

4.12.1.3 getTipAngajat()

```
string Supervizor::getTipAngajat ( ) const [override], [virtual]
```

Implements [Angajat](#).

The documentation for this class was generated from the following files:

- [Supervizor.h](#)
- [Supervizor.cpp](#)

4.13 Tehnician Class Reference

Inheritance diagram for Tehnician:

Collaboration diagram for Tehnician:

Public Member Functions

- **Tehnician** (const std::string &_nume, const std::string &_prenume, const std::string &_cnp, const [Data](#) &_dataAngajarii, const std::string &_orasDomiciliu)
- void **adaugaSpecializare** (const std::string &tip, const std::string &marca)
- bool **areSpecializare** (const std::string &tip, const std::string &marca) const
- bool **poatePrimiCerere** () const
- void **adaugaldCerereActivă** (int id)
- void **finalizareCerere** (int idCerere, double valoareReparatie, int durataEfectiva)
- double **getDurataTotalaLucrata** () const
- int **getNrCereriActive** () const
- double **calculeazaSalariu** () const override
- void **afisare** (std::ostream &) const override
- std::string **getTipAngajat** () const override

Public Member Functions inherited from [Angajat](#)

- **Angajat** (const std::string &, const std::string &, const std::string &, const [Data](#) &, const std::string &)
- int **getId** () const
- std::string **getNum** () const
- std::string **getPrenume** () const
- std::string **getCNP** () const
- [Data](#) **getDataAngajarii** () const
- std::string **getOrasDomiciliu** () const
- void **setNum** (const std::string &)
- void **setPrenume** (const std::string &)
- void **validareNum** (const std::string &, const std::string &) const
- double **calculeazaBonusFidelitate** () const
- bool **primestePrimaTransport** () const

Additional Inherited Members

Protected Attributes inherited from [Angajat](#)

- const int **id**
- std::string **nume**
- std::string **prenume**
- std::string **cnp**
- [Data](#) **dataAngajarii**
- std::string **orasDomiciliu**

Static Protected Attributes inherited from [Angajat](#)

- static int **nextID** =1000
- static constexpr double **SALARIU_BAZA** =4000.0
- static constexpr double **BONUS_DE_FIDELITARE_PROCENT** =5.0
- static constexpr int **ANI_FIDELITATE** =3
- static constexpr double **PRIMA_TRANS** =400.0

4.13.1 Member Function Documentation

4.13.1.1 [afisare\(\)](#)

```
void Tehnician::afisare (
    std::ostream & ) const [override], [virtual]
```

Reimplemented from [Angajat](#).

4.13.1.2 [calculeazaSalariu\(\)](#)

```
double Tehnician::calculeazaSalariu () const [override], [virtual]
```

Implements [Angajat](#).

4.13.1.3 `getTipAngajat()`

```
string Tehnician::getTipAngajat ( ) const [override], [virtual]
```

Implements [Angajat](#).

The documentation for this class was generated from the following files:

- Tehnician.h
- Tehnician.cpp

4.14 TV Class Reference

Inheritance diagram for TV:

Collaboration diagram for TV:

Public Member Functions

- **TV** (const std::string &, const std::string &, int, double, double, bool)
- double **getDiagonala** () const
- bool **getEstelnCm** () const
- void **afisare** (std::ostream &) const override
- std::unique_ptr<[Electrocasnic](#)> **cloneaza** () const override

Public Member Functions inherited from [Electrocasnic](#)

- **Electrocasnic** (const std::string &, const std::string &, const std::string &, int, double)
- std::string **getTip** () const
- std::string **getMarca** () const
- std::string **getModel** () const
- int **getAnFabricatie** () const
- double **getPretCatalog** () const
- int **getVechime** () const
- bool **operator==** (const [Electrocasnic](#) &altul) const

Additional Inherited Members

Protected Attributes inherited from [Electrocasnic](#)

- std::string **tip**
- std::string **marca**
- std::string **model**
- int **anFabricatie**
- double **pretCatalog**

4.14.1 Member Function Documentation

4.14.1.1 afisare()

```
void TV::afisare (
    std::ostream & ) const [override], [virtual]
```

Implements [Electrocasnic](#).

4.14.1.2 cloneaza()

```
unique_ptr< Electrocasnic > TV::cloneaza ( ) const [override], [virtual]
```

Implements [Electrocasnic](#).

The documentation for this class was generated from the following files:

- TV.h
- TV.cpp

Chapter 5

File Documentation

5.1 Angajat.h

```
00001 #pragma once
00002 #include <string>
00003 #include <iostream>
00004 #include "Data.h"
00005
00006 class Angajat{
00007
00008 protected:
00009
00010     //TODO : gandeste te la o denumire mai buna pentru id urile de la angajat si cerere
00011     static int nextID;
00012     const int id;
00013     std::string nume, prenume, cnp;
00014     Data dataAngajarii;
00015     std::string orasDomiciliu;
00016
00017     static constexpr double SALARIU_BAZA=4000.0;
00018     static constexpr double BONUS_DE_FIDELITARE_PROCENT=5.0;
00019     static constexpr int ANI_FIDELITATE=3;
00020     static constexpr double PRIMA_TRANS=400.0;
00021
00022 public:
00023
00024     Angajat():id(0){}
00025     Angajat(const std::string& ,const std::string& , const std::string& , const Data& , const
00026     std::string& );
00027     virtual ~Angajat()=default;
00028
00029     // getteri
00030
00031     int getId() const;
00032     std::string getNume() const;
00033     std::string getPrenume() const;
00034     std::string getCNP() const;
00035     Data getDataAngajarii() const;
00036     std::string getOrasDomiciliu() const;
00037
00038     //setteri
00039
00040     void setNume(const std::string& );
00041     void setPrenume(const std::string& );
00042
00043     //validare
00044
00045     void validareNume(const std::string& , const std::string& ) const;
00046
00047     void calculSalariu-- VIRTUAL PURE
00048     virtual double calculeazaSalariu() const=0;
00049     virtual void afisare(std::ostream& ) const;
00050
00051     //tip angajat-- VIRTUAL PURE
00052     virtual std::string getTipAngajat() const=0;
00053
00054
00055
00056
00057
```

```

00058     // auxiliare
00059     double calculeazaBonusFidelitate() const;
00060     bool primestePrimaTransport() const;
00061
00062 };

```

5.2 AngajatFactory.h

```

00001 #pragma once
00002 #include <string>
00003 #include <memory>
00004 #include <vector>
00005 #include <map>
00006 #include <set>
00007 #include <iostream>
00008
00009 #include "Angajat.h"
00010 #include "Data.h"
00011
00012 class AngajatFactory {
00013 public:
00014
00015     static std::unique_ptr<Angajat> creeazaAngajat(const std::string& tip, const std::string&
00016         nume, const std::string& prenume, const std::string& cnp, const Data& dataAngajarii, const std::string&
00017         orasDomiciliu, const std::map<std::string, std::set<std::string>>& specializari = {});
00018
00019     static std::unique_ptr<Angajat> creeazaDinCSV(const std::string& linie);
00020
00021     static std::vector<std::string> splitLine(const std::string& linie, char delimitator);
00022     static std::map<std::string, std::set<std::string>> parseazaSpecializari(const std::string& text);
00023 };

```

5.3 CerereReparatie.h

```

00001 #pragma once
00002 #include "Electrocasnic.h"
00003 #include "Tehnician.h"
00004 #include <string>
00005 #include <iostream>
00006 #include <memory>
00007 #include <ctime>
00008
00009 enum class StareCerere{
00010
00011     IN_ASTEPTARE,
00012     REPARTIZATA,
00013     IN_LUCRU,
00014     FINALIZATA,
00015     INVALIDA
00016 };
00017
00018
00019 class CerereReparatie {
00020     //TODO la fel! alta denumire pentru id uri!!
00021     static int nextID;
00022     int id;
00023     std::unique_ptr<Electrocasnic> aparat;
00024     time_t timestamp;
00025     int nivelComplexitate; // 0-5 (0- nu se poate repara! REMAT!!)
00026     int durataEstimata;
00027     int durataRamasa;
00028     double pretReparatie;
00029     StareCerere stare;
00030     int idTehnician; // -1 daca e nerepartizata
00031
00032     public:
00033         CerereReparatie()=default;
00034         CerereReparatie(std::unique_ptr<Electrocasnic> _aparat, const time_t _timestamp, int
00035             _nivelComplexitate);
00036
00037         // constructor de copiere si operatorul egal pentru unique_ptr ???????
00038
00039         CerereReparatie(const CerereReparatie&);
00040         CerereReparatie& operator=(const CerereReparatie& );
00041
00042         //getteri

```

```

00043     int getId() const;
00044     int getNivelComplexitate() const;
00045     const Electrocasnic* getAparat() const;
00046     time_t getTimeStamp() const;
00047     int getDurataEstimata() const;
00048     int getDurataRamasa() const;
00049     double getPretReparatie() const;
00050     StareCerere getStare() const;
00051     int getIdTehnician() const;
00052
00053
00054     // setters
00055
00056     void setStare(const StareCerere);
00057     void setIdTehnician(int);
00058
00059
00060     // Procesare
00061
00062     void proceseaza(); // reduce durata ramasa
00063     bool esteFinalizata() const;
00064     bool esteValida() const;
00065
00066     void afisare(std::ostream&) const;
00067     std::string getTimeStampString() const;
00068
00069
00070
00071     //Comparare pentru sortare?????
00072     bool operator<(const CerereReparatie& ) const;
00073
00074 };
00075

```

5.4 CNPValidator.h

```

00001 #pragma once
00002 #include "Data.h"
00003 #include <string>
00004
00005 class CNPValidator {
00006 public:
00007     static bool esteValid(const std::string& cnp);
00008     static Data getDataNasterii(const std::string& cnp);
00009     static char getSex(const std::string& cnp);
00010
00011 private:
00012     static bool verificaCifreControl(const std::string& cnp);
00013     static bool verificaData(int an, int luna, int zi);
00014
00015     CNPValidator() = delete;
00016 };
00017

```

5.5 Data.h

```

00001 #pragma once
00002 #include <iostream>
00003 #include <string>
00004 #include <ostream>
00005 #include <istream>
00006
00007 class Data {
00008     int zi, luna, an;
00009
00010 public:
00011     Data()=default;
00012     Data(int, int, int);
00013     Data(const std::string& ); // "DD.MM.YYYY"
00014
00015     // VALIDARI
00016     bool esteValida() const;
00017
00018
00019     //getteri
00020     int getVarsta() const;
00021     int getZi() const;
00022     int getLuna() const;
00023     int getAn() const;

```

```

00024
00025 //operatori
00026 bool operator<(const Data&);
00027 bool operator>(const Data&);
00028 bool operator==(const Data&);
00029 friend std::ostream& operator<<(std::ostream&, const Data&);
00030
00031 //static
00032 static Data dataCurenta();
00033
00034 // string toString() const
00035
00036
00037
00038 };

```

5.6 Electrocasnic.h

```

00001 #pragma once
00002 #include <string>
00003 #include <memory>
00004 #include "Data.h"
00005
00006 class Electrocasnic {
00007 protected:
00008     std::string tip, marca, model;
00009     int anFabricatie;
00010     double pretCatalog;
00011
00012 public:
00013     Electrocasnic() = default;
00014     Electrocasnic(const std::string&, const std::string&, const std::string&, int, double );
00015
00016     virtual ~Electrocasnic() = default;
00017
00018     // getteri
00019     std::string getTip() const;
00020     std::string getMarca() const;
00021     std::string getModel() const;
00022     int getAnFabricatie() const;
00023     double getPretCatalog() const;
00024
00025     int getVechime() const;
00026
00027     virtual void afisare(std::ostream&) const = 0;
00028     virtual std::unique_ptr<Electrocasnic> cloneaza() const = 0;
00029
00030     bool operator==(const Electrocasnic& altul) const;
00031
00032 };

```

5.7 ElectrocasnicFactory.h

```

00001 #pragma once
00002 #include <string>
00003 #include <memory>
00004 #include <map>
00005 #include <vector>
00006
00007 #include "Electrocasnic.h"
00008
00009 class ElectrocasnicFactory {
00010 public:
00011
00012     static std::unique_ptr<Electrocasnic> creeazaElectrocasnic(const std::string& tip, const
00013         std::string& marca, const std::string& model, int anFabricatie, double pretCatalog, const
00014         std::map<std::string, std::string>& dateSpecifice);
00015
00016     static std::map<std::string, std::string> parseazaDateSpecifice(const std::string& linie);
00017
00018 };

```

5.8 Frigider.h

```

00001 #include "Electrocasnic.h"
00002 #include <string>

```

```

00003 #include <memory>
00004
00005
00006 class Frigider: public Electrocasnic{
00007     bool areCongelator;
00008
00009
00010 public:
00011     Frigider()=default;
00012     Frigider(const std::string&, const std::string&, int, double, bool);
00013     bool getAreCongelator() const;
00014     void afisare(std::ostream &) const override;
00015     std::unique_ptr<Electrocasnic> cloneaza() const override;
00016
00017 };

```

5.9 MasinaDeSpalat.h

```

00001 #pragma once
00002 #include "Electrocasnic.h"
00003
00004 #include <string>
00005 #include <memory>
00006
00007 class MasinaDeSpalat: public Electrocasnic {
00008     double capacitate; // in kg
00009
00010
00011 public:
00012     MasinaDeSpalat()=default;
00013     MasinaDeSpalat(const std::string&, const std::string&, int,double,double );
00014     double getCapacitate() const;
00015     void afisare(std::ostream&) const override;
00016     std::unique_ptr<Electrocasnic> cloneaza() const override;
00017
00018 };

```

5.10 Receptioner.h

```

00001 #pragma once
00002
00003 #include "Angajat.h"
00004 #include <string>
00005 #include <vector>
00006
00007 class Receptioner: public Angajat{
00008     std::vector<int> idCereriInregistrate;
00009
00010
00011 public:
00012     Receptioner()=default;
00013     Receptioner(const std::string&, const std::string&, const std::string&, const Data& ,const std::string& );
00014     void adaugaCerere(int );
00015     const std::vector<int>& getCereri() const;
00016     double calculeazaSalariu() const override;
00017     void afisare(std::ostream&) const override;
00018     std::string getTipAngajat() const override;
00019
00020 };

```

5.11 ServiceManager.h

```

00001 #pragma once
00002 #include <vector>
00003 #include <map>
00004 #include <set>
00005 #include <queue>
00006 #include <memory>
00007 #include <string>
00008 #include <fstream>
00009 #include <algorithm>
00010
00011 #include "Angajat.h"
00012 #include "Receptioner.h"

```

```

00013 #include "Tehnician.h"
00014 #include "Supervizor.h"
00015 #include "Electrocasnic.h"
00016 #include "CerereReparatie.h"
00017 #include "AngajatFactory.h"
00018 #include "ElectrocasnicFactory.h"
00019
00020 class ServiceManager {
00021 private:
00022     // Singleton instance
00023     static ServiceManager* instance;
00024
00025     // Constructor privat pentru Singleton
00026     ServiceManager();
00027
00028     // Colecții principale
00029     std::vector<std::unique_ptr<Angajat>> angajati;
00030
00031     // Catalog: map<tip, map<marca, map<model, unique_ptr<Electrocasnic>>>
00032     std::map<std::string, std::map<std::string, std::map<std::string, std::unique_ptr<Electrocasnic>>>
00033     catalogElectrocasnice;
00034
00035     // Cereri
00036     std::vector<std::unique_ptr<CerereReparatie>> toateCererile; // toate cererile primite
00037     std::vector<CerereReparatie*> cereriActive; // pointeri la cereri in lucru
00038     std::queue<CerereReparatie*> cereriInAsteptare;
00039     std::vector<CerereReparatie*> cereriFinalizate;
00040
00041     // Pentru statistici
00042     std::map<std::string, int> aparateNereparabile; // "tip_marca_model" -> nr_aparitii
00043
00044     // Simulare
00045     int timpCurent;
00046     bool esteInSimulare;
00047 public:
00048     // Singleton
00049     static ServiceManager* getInstance();
00050     static void deleteInstance();
00051
00052     // Interzice copierea
00053     ServiceManager(const ServiceManager&) = delete;
00054     ServiceManager& operator=(const ServiceManager&) = delete;
00055
00056     ~ServiceManager();
00057
00058     // === GESTIUNE ANGAJATI ===
00059
00060     // Verificări
00061     bool arePersonalMinim() const;
00062     int getNumarTehnicieni() const;
00063     int getNumarReceptioneri() const;
00064     int getNumarSupervizori() const;
00065
00066     // CRUD
00067     void adaugaAngajat(std::unique_ptr<Angajat> angajat);
00068     bool stergeAngajat(const std::string& cnp);
00069     bool modificaNumeAngajat(const std::string& cnp, const std::string& numeNou, const std::string&
00070     prenumeNou);
00071     Angajat* cautaAngajatDupaCNP(const std::string& cnp);
00072     Angajat* cautaAngajatDupaID(int id);
00073
00074     // Afisare
00075     void afiseazaAngajat(const std::string& cnp) const;
00076     void afiseazaTotiAngajatii() const;
00077
00078     // Încărcare din fișier
00079     void incarcaAngajatiDinCSV(const std::string& numeFisier);
00080
00081     // === GESTIUNE ELECTROCASNICE ===
00082
00083     // CRUD Catalog
00084     void adaugaModelInCatalog(std::unique_ptr<Electrocasnic> electrocasnic);
00085     bool stergeModelDinCatalog(const std::string& tip, const std::string& marca, const std::string&
00086     model);
00087
00088     // Verificări
00089     bool poateFiReparat(const std::string& tip, const std::string& marca, const std::string& model)
00090     const;
00091     const Electrocasnic* getDetaliiModel(const std::string& tip, const std::string& marca, const
00092     std::string& model) const;
00093
00094     // Afisare
00095     void afiseazaCatalogComplet() const;
00096     void afiseazaAparateNereparabile() const;
00097
00098     // Încărcare

```

```

00095     void incarcaCatalogDinCSV(const std::string& numeFisier);
00096
00097     // === GESTIUNE CERERI ===
00098
00099     // Primire cereri
00100     void primesteCerere(std::unique_ptr<CerereReparatie> cerere);
00101     void incarcaCereriDinCSV(const std::string& numeFisier);
00102
00103     // Verificare validitate
00104     bool esteCerereValida(const CerereReparatie& cerere);
00105
00106     // === SIMULARE ===
00107
00108     // Alocare automată
00109     Tehnician* gasesteTehnicianPotrivit(const CerereReparatie& cerere);
00110     void alocaCerere(CerereReparatie* cerere, Tehnician* tehnician);
00111     void procesareCereriInAsteptare();
00112
00113     // Simulare în timp real
00114     void startSimulare();
00115     void executeazaTic();
00116     void afiseazaStatusTic() const;
00117     void stopSimulare();
00118
00119     // Finalizare cereri
00120     void finalizeazaCerere(CerereReparatie* cerere);
00121
00122     // === RAPORTĂRI ===
00123
00124     void genereazaRaportTop3Salarii(const std::string& numeFisier) const;
00125     void genereazaRaportCeaMaiLungaReparatie(const std::string& numeFisier) const;
00126     void genereazaRaportCereriInAsteptare(const std::string& numeFisier) const;
00127
00128     // === UTILITĂȚI ===
00129
00130     void afiseazaStatistici() const;
00131     void reset(); // pentru testare
00132 };

```

5.12 Supervizor.h

```

00001 #pragma once
00002
00003 #include "Angajat.h"
00004 #include <string>
00005 #include <vector>
00006
00007
00008 class Supervizor: public Angajat{
00009
0010     static constexpr double SPOR_CONDUCERE_PROCENT=20.0;
0011
0012     public:
0013
0014         Supervizor()=default;
0015         Supervizor(const std::string&, const std::string&, const std::string&, const Data&, const
0016             std::string&);
0016         double calculeazaSalariu() const override;
0017         void afisare(std::ostream&) const override;
0018         std::string getTipAngajat() const override;
0019
0020
0021
0022
0023
0024
0025 };

```

5.13 Tehnician.h

```

00001 #pragma once
00002 #include "Angajat.h"
00003 #include <map>
00004 #include <set>
00005 #include <vector>
00006
00007 class Tehnician : public Angajat {
00008     std::map<std::string, std::set<std::string>> specializari;

```

```

00010
00011     std::vector<int> idCereriActive;
00012     double valoareaTotalaReparatii;
00013     double durataTotalaLucrata;
00014
00015     static constexpr double BONUS_REPARATII_PROCENT = 2.0;
00016     static constexpr int MAX_CERERI_ACTIVE = 3;
00017
00018 public:
00019     Tehnician() = default;
00020     Tehnician(const std::string& _nume, const std::string& _prenume, const std::string& _cnp,
00021     const Data& _dataAngajarii, const std::string& _orasDomiciliu);
00022
00023     // Specializari
00024     void adaugaSpecializare(const std::string& tip, const std::string& marca);
00025     bool areSpecializare(const std::string& tip, const std::string& marca) const;
00026
00027     // Gestiune Cereri Active
00028     bool poatePrimiCerere() const;
00029     void adaugaIdCerereActivă(int id);
00030     void finalizareCerere(int idCerere, double valoareReparatie, int durataEfectiva);
00031
00032     // Getteri pentru alocare automata
00033     double getDurataTotalaLucrata() const;
00034     int getNrCereriActive() const ;
00035
00036     // Salariu si Afisare
00037     double calculeazaSalariu() const override;
00038     void afisare(std::ostream& ) const override;
00039     std::string getTipAngajat() const override;
00040 };

```

5.14 TV.h

```

00001 #pragma once
00002 #include "Electrocasnic.h"
00003
00004 #include <string>
00005 #include <memory>
00006 #include <iostream>
00007
00008 class TV: public Electrocasnic{
00009
00010     double diagonala;
00011     bool esteInCm; // true=cm si false=incii
00012
00013 public:
00014     TV()=default;
00015     TV(const std::string&, const std::string&, int, double, double, bool);
00016
00017     double getDiagonala() const;
00018     bool getEsteInCm() const;
00019     void afisare(std::ostream&) const override;
00020
00021     std::unique_ptr<Electrocasnic> cloneaza() const override;
00022
00023 };

```

Index

afisare
 Frigider, 10
 MasinaDeSpalat, 12
 Receptioner, 13
 Supervizor, 16
 Tehnician, 17
 TV, 19
Angajat, 7
AngajatFactory, 7

calculeazaSalariu
 Receptioner, 13
 Supervizor, 16
 Tehnician, 17
CerereReparatie, 7

cloneaza
 Frigider, 10
 MasinaDeSpalat, 12
 TV, 19

CNPValidator, 8

Data, 8

Electrocasnic, 9
ElectrocasnicFactory, 9

Frigider, 10
 afisare, 10
 cloneaza, 10

getTipAngajat
 Receptioner, 13
 Supervizor, 16
 Tehnician, 17

MasinaDeSpalat, 11
 afisare, 12
 cloneaza, 12

Receptioner, 12
 afisare, 13
 calculeazaSalariu, 13
 getTipAngajat, 13

ServiceManager, 14
Supervizor, 15
 afisare, 16
 calculeazaSalariu, 16
 getTipAngajat, 16

Tehnician, 16