

РУ „Ангел Кънчев“

# КУРСОВА РАБОТА

## ПО

## ООП

Име: Алекс Георгиев Иванов

Група: 25р

Курс: Първи

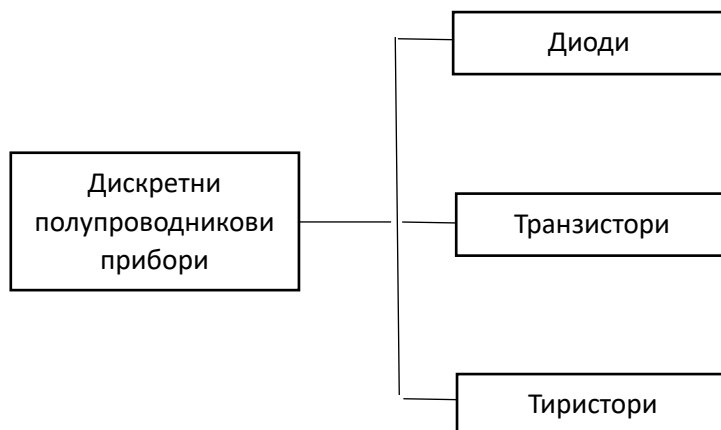
Факултетен номер: 233107

Дата:.....

Проверил:.....

## 1. Задание №6.

Дадена е следната класификация:



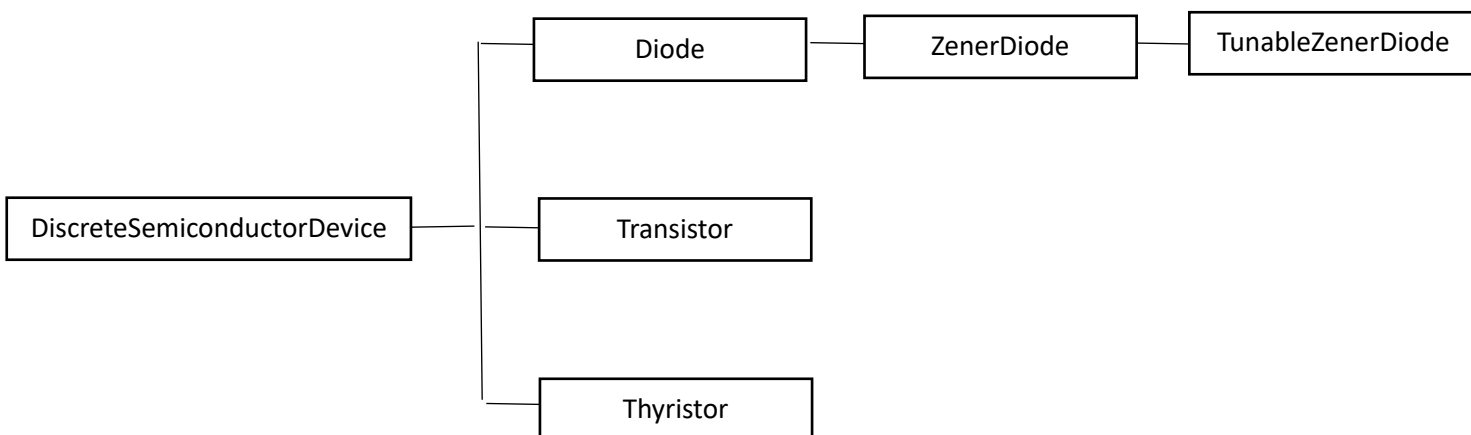
Класификацията да се продължи поне на още две нива. Да се състави йерархия от класове, отразяваща създадената класификация. Да се декларират съответните класове, като всеки клас, с изключение на базовия, да имат поне по 2 собствени атрибута.

Да се дефинира виртуална функция, която извежда характеристиките на обект от всеки клас на йерархията. Във функцията `main` да се изгради двусвързан списък от обекти от произволни класове в йерархията. Да се разработят функции, които обхождат двусвързания (в двете посоки) списък и извеждат информация за признаците на включените в него обекти.

Декларациите на всеки клас от йерархията да бъдат оформени в отделни заглавни (.h) файлове.

Дефинициите на всеки клас и функцията `"main"` да бъдат оформени в отделни модули (.cpp файлове). Във всички файлове, съдържащи дефинициите на класовете и функцията `"main"`, чрез директивата `#include` да се включат съответните заглавни файлове, съдържащи декларациите на класовете. Да се създаде проект, състоящ се от създадените модули.

## 2. Схема на наследяване на класовете.



3. Пълно описание на декларираните класове.

ИМЕ НА КЛАСА: DiscreteSemiconductorDevice

БАЗОВ ЗА: Diode, ZenerDiode, TunableZenerDiode, Transistor, Thyristor

ЧЛЕНОВЕ ДАННИ:

име: brand; тип: string

име: model; тип: string

ФУНКЦИИ-ЧЛЕНОВЕ (МЕТОДИ):

DiscreteSemiconductorDevice(const string &brand, const string &model) – инициализиращ конструктор

virtual string ToString() const; – не приема параметри и връща текс с информация за обекта:

Class DiscreteSemiconductorDevice:

- Brand: {brand}

- Model: {model}

ИМЕ НА КЛАСА: Diode

НАСЛЕДЯВА ОТ: DiscreteSemiconductorDevice

БАЗОВ ЗА: ZenerDiode

ЧЛЕНОВЕ ДАННИ:

име: brand; тип: string

име: model; тип: string

име: forwardVoltage; тип: double

име: maxCurrent; тип: double

ФУНКЦИИ-ЧЛЕНОВЕ (МЕТОДИ):

Diode(const string &brand, const string &model, double forwardVoltage, double maxCurrent);– инициализиращ конструктор

virtual string ToString() const override;– не приема параметри и връща текс с информация за обекта:

Class Diode, based on:

Class DiscreteSemiconductorDevice:

- Brand: {brand}

- Model: {model}

- ForwardVoltage: { forwardVoltage }

- MaxCurrent: { maxCurrent }

ИМЕ НА КЛАСА: ZenerDiode

НАСЛЕДЯВА ОТ: Diode

БАЗОВ ЗА: TunableZenerDiode

ЧЛЕНОВЕ ДАННИ:

име: brand; тип: string

име: model; тип: string

име: forwardVoltage; тип: double

име: maxCurrent; тип: double

име: reverseBreakdownVoltage; тип: double

име: temperatureCoefficient; тип: double

ФУНКЦИИ-ЧЛЕНОВЕ (МЕТОДИ):

ZenerDiode(const string &brand, const string &model, double forwardVoltage, double maxCurrent, double reverseBreakdownVoltage, double temperatureCoefficient);—

инициализиращ конструктор

virtual string ToString() const override;— не приема параметри и връща текст с информация за обекта:

Class ZenerDiode, based on:

Class Diode:

- Brand: {brand}

- Model: {model}

- ForwardVoltage: { forwardVoltage }

- MaxCurrent: { maxCurrent }

- ReverseBreakdownVoltage: { reverseBreakdownVoltage }

- TemperatureCoefficient: { temperatureCoefficient }

ИМЕ НА КЛАСА: TunableZenerDiode

НАСЛЕДЯВА ОТ: ZenerDiode

ЧЛЕНОВЕ ДАННИ:

име: brand; тип: string

име: model; тип: string

име: forwardVoltage; тип: double

име: maxCurrent; тип: double

име: reverseBreakdownVoltage; тип: double

име: temperatureCoefficient; тип: double

име: controlVoltage; тип: double

име: adjustmentRange; тип: double

ФУНКЦИИ-ЧЛЕНОВЕ (МЕТОДИ):

TunableZenerDiode(const string &brand, const string &model, double forwardVoltage, double maxCurrent, double reverseBreakdownVoltage, double temperatureCoefficient, double controlVoltage, double adjustmentRange); – инициализиращ конструктор  
virtual string ToString() const override; – не приема параметри и връща текс с информация за обекта:

Class TunableZenerDiode, based on:

Class ZenerDiode:

- Brand: {brand}

- Model: {model}

- ForwardVoltage: { forwardVoltage }

- MaxCurrent: { maxCurrent }

- ReverseBreakdownVoltage: { reverseBreakdownVoltage }

- TemperatureCoefficient: { temperatureCoefficient }

- ControlVoltage: { controlVoltage }

- AdjustmentRange: { adjustmentRange }

ИМЕ НА КЛАСА: Transistor

НАСЛЕДЯВА ОТ: DiscreteSemiconductorDevice

ЧЛЕНОВЕ ДАННИ:

име: brand; тип: string

име: model; тип: string

име: baseEmitterVoltage; тип: double

име: collectorCurrent; тип: double

ФУНКЦИИ-ЧЛЕНОВЕ (МЕТОДИ):

Transistor(const string &brand, const string &model, double baseEmitterVoltage, double collectorCurrent); – инициализиращ конструктор

virtual string ToString() const override; – не приема параметри и връща текс с информация за обекта:

Class Transistor based on DiscreteSemiconductorDevice:

- Brand: {brand}

- Model: {model}

- BaseEmitterVoltage: { baseEmitterVoltage }

- CollectorCurrent: { collectorCurrent }

ИМЕ НА КЛАСА: Thyristor

НАСЛЕДЯВА ОТ: DiscreteSemiconductorDevice

ЧЛЕНОВЕ ДАННИ:

име: brand; тип: string

име: model; тип: string

име: gateTriggerVoltage; тип: double

име: holdingCurrent; тип: double

ФУНКЦИИ-ЧЛЕНОВЕ (МЕТОДИ):

Thyristor (const string &brand, const string &model, double gateTriggerVoltage, double holdingCurrent); – инициализиращ конструктор

virtual string ToString() const override; – не приема параметри и връща текс с информация за обекта:

Class Thyristor based on DiscreteSemiconductorDevice:

- Brand: {brand}

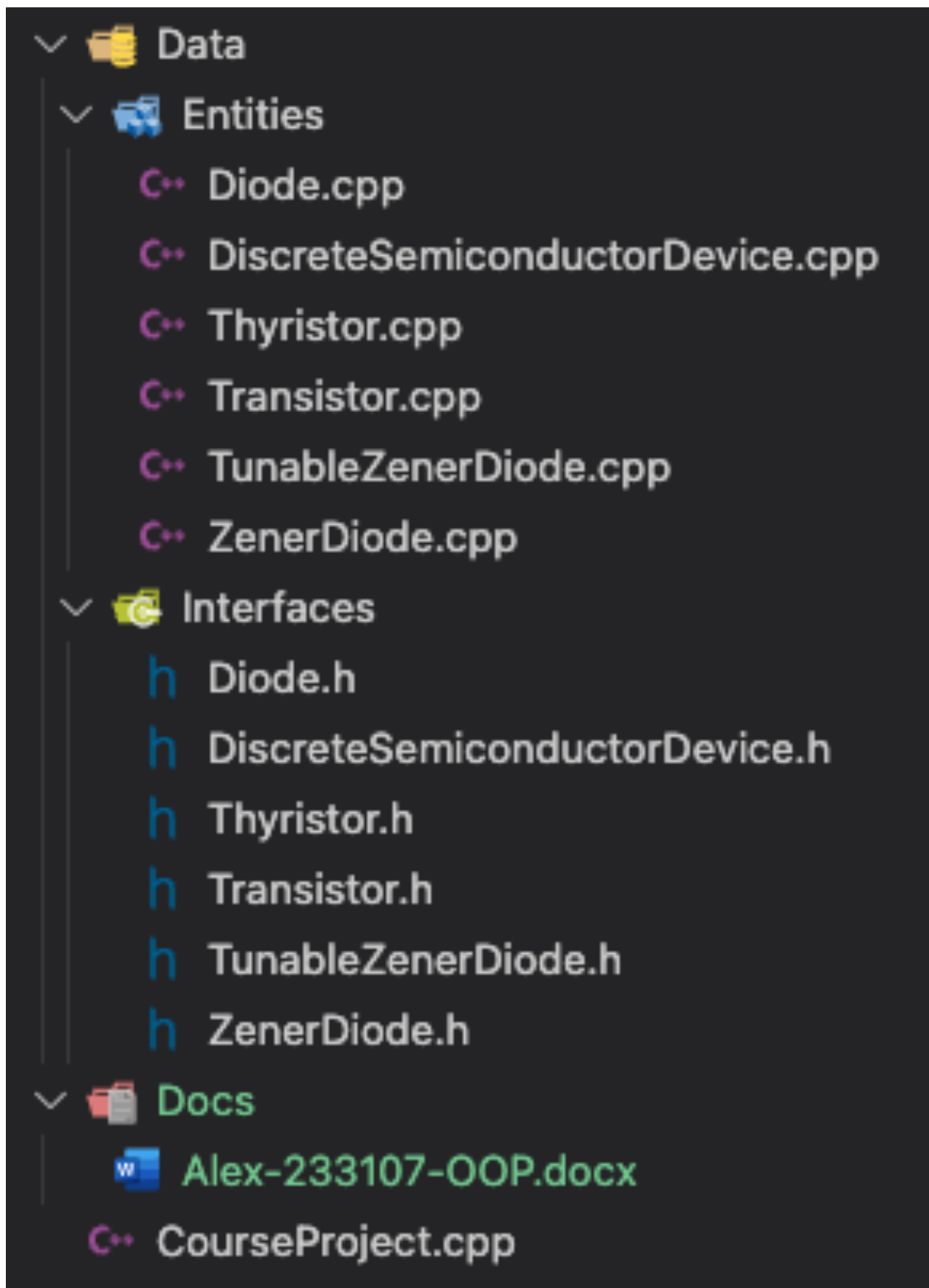
- Model: {model}

- GateTriggerVoltage: { gateTriggerVoltage }

- HoldingCurrent: { holdingCurrent }



4. Код.



```
DiscreteSemiconductorDevice.h - courseProject
Data > Interfaces > DiscreteSemiconductorDevice.h > ...
1 #ifndef DISCRETESEMICONDUCTORDEVICE_H
2 #define DISCRETESEMICONDUCTORDEVICE_H
3
4 #include <string>
5
6 using namespace std;
7
8 class DiscreteSemiconductorDevice
9 {
10 public:
11     // Constructor and Destructor
12     DiscreteSemiconductorDevice(const string &brand, const string &model);
13     virtual ~DiscreteSemiconductorDevice();
14
15     // Copy Constructor and Assignment Operator
16     DiscreteSemiconductorDevice(const DiscreteSemiconductorDevice &other);
17     DiscreteSemiconductorDevice &operator=(const DiscreteSemiconductorDevice &other);
18
19     // Accessor methods
20     string GetBrand() const;
21     string GetModel() const;
22
23     // Virtual method for displaying device details
24     virtual string ToString() const;
25
26 protected:
27     string brand;
28     string model;
29 };
30
31 #endif
32
```

```
DiscreteSemiconductorDevice.cpp - courseProject
Data > Entities > DiscreteSemiconductorDevice.cpp > Tostring() const
1 #include "../Interfaces/DiscreteSemiconductorDevice.h"
2
3 #include <iostream>
4 #include <sstream> // Include the header for demangling
5 #include <cstdlib>
6
7 // Constructor
8 DiscreteSemiconductorDevice::DiscreteSemiconductorDevice(const string &brand, const string &
model)
9 : brand(brand), model(model) {}
10
11 // Destructor
12 DiscreteSemiconductorDevice::~DiscreteSemiconductorDevice() {}
13
14 // Copy Constructor
15 DiscreteSemiconductorDevice::DiscreteSemiconductorDevice(const DiscreteSemiconductorDevice &
other)
16 : brand(other.brand), model(other.model) {}
17
18 // Assignment Operator
19 DiscreteSemiconductorDevice &DiscreteSemiconductorDevice::operator=(const
DiscreteSemiconductorDevice &other)
20 {
21     if (this != &other)
22     {
23         brand = other.brand;
24         model = other.model;
25     }
26     return *this;
27 }
28
29 // Accessor methods
30 string DiscreteSemiconductorDevice::GetBrand() const
31 {
32     return brand;
33 }
34
35 string DiscreteSemiconductorDevice::GetModel() const
36 {
37     return model;
38 }
39
40 // Virtual method for displaying device details
41 string DiscreteSemiconductorDevice::ToString() const
42 {
43     int status;
44     char *demangled_name = abi::__cxa_demangle(typeid(*this).name(), NULL, NULL, &status);
45
46     string result;
47     result += "\n";
48     result += "Device Details:";
49     result += "\n";
50     if (status == 0 && demangled_name != nullptr)
51     {
52         result += demangled_name; // Use the demangled name if demangling was successful
53         std::free(demangled_name); // Free the memory allocated by __cxa_demangle
54     }
55     else
56     {
57         result += typeid(*this).name(); // Fallback to mangled name if demangling fails
58     }
59     result += "\nBrand: " + brand;
60     result += "\nModel: " + model;
61     return result;
62 }
```

```
Diode.h - courseProject
Data > Interfaces > Diode.h > Tostring() const
1 #ifndef DIODE_H
2 #define DIODE_H
3
4 #include "DiscreteSemiconductorDevice.h"
5
6 #include <string>
7
8 using namespace std;
9
10 class Diode : public DiscreteSemiconductorDevice
11 {
12 public:
13     // Constructor and Destructor
14     Diode(const string &brand, const string &model, double forwardVoltage, double
maxCurrent);
15     virtual ~Diode();
16
17     // Copy Constructor and Assignment Operator
18     Diode(const Diode &other);
19     Diode &operator=(const Diode &other);
20
21     // Virtual method for displaying device details
22     virtual string ToString() const override;
23
24     // Accessor methods
25     double GetForwardVoltage() const;
26     double GetMaxCurrent() const;
27
28 protected:
29     double forwardVoltage;
30     double maxCurrent;
31 };
32
33 #endif
34
```

```
Diode.cpp - courseProject
Data > Entities > Diode.cpp > ...
1 #include "../Interfaces/Diode.h"
2
3 #include <iostream>
4 #include <typeinfo>
5
6 Diode::Diode(const string &brand, const string &model, double forwardVoltage, double
maxCurrent)
7 : DiscreteSemiconductorDevice(brand, model, forwardVoltage(forwardVoltage), maxCurrent
(maxCurrent)) {}
8
9 Diode::Diode(const Diode &other)
10 : DiscreteSemiconductorDevice(other, forwardVoltage(other.forwardVoltage), maxCurrent
(other.maxCurrent)) {}
11
12 Diode &Diode::operator=(const Diode &other)
13 {
14     if (this != &other)
15     {
16         DiscreteSemiconductorDevice::operator=(other);
17         forwardVoltage = other.forwardVoltage;
18         maxCurrent = other.maxCurrent;
19     }
20     return *this;
21 }
22
23 Diode::~Diode() {}
24
25 double Diode::GetForwardVoltage() const
26 {
27     return forwardVoltage;
28 }
29
30 double Diode::GetMaxCurrent() const
31 {
32     return maxCurrent;
33 }
34
35 string Diode::ToString() const
36 {
37     string result = DiscreteSemiconductorDevice::ToString();
38     result += "\nForward Voltage: " + to_string(forwardVoltage) + " V";
39     result += "\nMaximum Current: " + to_string(maxCurrent) + " A";
40     return result;
41 }
```

```

TunableZenerDiode.h
1 #ifndef TUNABLE_ZENER_DIODE_H
2 #define TUNABLE_ZENER_DIODE_H
3
4 #include "ZenerDiode.h"
5
6 class TunableZenerDiode : public ZenerDiode
7 {
8 public:
9     TunableZenerDiode(const string &brand, const string &model, double
        forwardVoltage, double maxCurrent,
        double reverseBreakdownVoltage, double
        temperatureCoefficient, double
        controlVoltage, double adjustmentRange);
10
11     virtual ~TunableZenerDiode();
12
13     // Copy Constructor and Assignment Operator
14     TunableZenerDiode(const TunableZenerDiode &other);
15     TunableZenerDiode &operator=(const TunableZenerDiode &other);
16
17     double GetControlVoltage() const;
18     double GetAdjustmentRange() const;
19
20     virtual string ToString() const override;
21
22 protected:
23     double controlVoltage;
24     double adjustmentRange;
25 };
26
27 #endif
28
29
TunableZenerDiode.cpp - courseProject
1 #include "../Interfaces/TunableZenerDiode.h"
2
3 #include <iostream>
4 #include <typeinfo>
5
6 TunableZenerDiode::TunableZenerDiode(const string &brand, const string &model,
        double forwardVoltage, double maxCurrent,
        double reverseBreakdownVoltage, double
        temperatureCoefficient, double
        controlVoltage, double adjustmentRange)
7 : ZenerDiode(brand, model, forwardVoltage, maxCurrent,
        reverseBreakdownVoltage, temperatureCoefficient),
        controlVoltage(controlVoltage), adjustmentRange(adjustmentRange) {}
8
9 TunableZenerDiode::~TunableZenerDiode() {}
10
11 // Copy Constructor
12 TunableZenerDiode::TunableZenerDiode(const TunableZenerDiode &other)
13 : ZenerDiode(other), controlVoltage(other.controlVoltage), adjustmentRange
        (other.adjustmentRange) {}
14
15 // Assignment Operator
16 TunableZenerDiode &TunableZenerDiode::operator=(const TunableZenerDiode &other)
17 {
18     if (this != &other)
19     {
20         ZenerDiode::operator=(other);
21         controlVoltage = other.controlVoltage;
22         adjustmentRange = other.adjustmentRange;
23     }
24     return *this;
25 }
26
27 double TunableZenerDiode::GetControlVoltage() const
28 {
29     return controlVoltage;
30 }
31
32 double TunableZenerDiode::GetAdjustmentRange() const
33 {
34     return adjustmentRange;
35 }
36
37 string TunableZenerDiode::ToString() const
38 {
39     string result = ZenerDiode::ToString();
40     result += "\n\t\tControl Voltage: " + to_string(controlVoltage) + " V";
41     result += "\n\t\tAdjustment Range: " + to_string(adjustmentRange) + " V";
42     return result;
43 }
44
45
46

```

```

ZenerDiode.h
1 #ifndef ZENER_DIODE_H
2 #define ZENER_DIODE_H
3
4 #include "Diode.h"
5
6 class ZenerDiode : public Diode
7 {
8 public:
9     ZenerDiode(const string &brand, const string &model, double forwardVoltage, double
        maxCurrent,
        double reverseBreakdownVoltage, double temperatureCoefficient);
10
11     virtual ~ZenerDiode();
12
13     ZenerDiode(const ZenerDiode &other);
14     ZenerDiode &operator=(const ZenerDiode &other);
15
16     double GetReverseBreakdownVoltage() const;
17     double GetTemperatureCoefficient() const;
18
19     virtual string ToString() const override;
20
21 protected:
22     double reverseBreakdownVoltage;
23     double temperatureCoefficient;
24 };
25
26 #endif
27
ZenerDiode.cpp - courseProject
1 #include "../Interfaces/ZenerDiode.h"
2
3 #include <iostream>
4 #include <typeinfo>
5
6 ZenerDiode::ZenerDiode(const string &brand, const string &model, double forwardVoltage,
        double maxCurrent,
        double reverseBreakdownVoltage, double temperatureCoefficient)
7 : Diode(brand, model, forwardVoltage, maxCurrent),
        reverseBreakdownVoltage(reverseBreakdownVoltage), temperatureCoefficient
        (temperatureCoefficient) {}
8
9 ZenerDiode::~ZenerDiode() {}
10
11 // Copy Constructor
12 ZenerDiode::ZenerDiode(const ZenerDiode &other)
13 : Diode(other), reverseBreakdownVoltage(other.reverseBreakdownVoltage),
        temperatureCoefficient(other.temperatureCoefficient) {}
14
15 // Assignment Operator
16 ZenerDiode &ZenerDiode::operator=(const ZenerDiode &other)
17 {
18     if (this != &other)
19     {
20         Diode::operator=(other);
21         reverseBreakdownVoltage = other.reverseBreakdownVoltage;
22         temperatureCoefficient = other.temperatureCoefficient;
23     }
24     return *this;
25 }
26
27 double ZenerDiode::GetReverseBreakdownVoltage() const
28 {
29     return reverseBreakdownVoltage;
30 }
31
32 double ZenerDiode::GetTemperatureCoefficient() const
33 {
34     return temperatureCoefficient;
35 }
36
37 string ZenerDiode::ToString() const
38 {
39     string result = Diode::ToString();
40     result += "\n\t\tReverse Breakdown Voltage: " + to_string(reverseBreakdownVoltage) + "
        V";
41     result += "\n\t\tTemperature Coefficient: " + to_string(temperatureCoefficient) + " %/
        C";
42     return result;
43 }
44
45
46

```

Transistor.h

```

1 #ifndef TRANSISTOR_H
2 #define TRANSISTOR_H
3
4 #include "DiscreteSemiconductorDevice.h"
5
6 #include <string>
7
8 using namespace std;
9
10 class Transistor : public DiscreteSemiconductorDevice
11 {
12 public:
13     // Constructor and Destructor
14     Transistor(const string &brand, const string &model, double baseEmitterVoltage, double
15         collectorCurrent);
16     virtual ~Transistor();
17
18     // Copy Constructor and Assignment Operator
19     Transistor(const Transistor &other);
20     Transistor &operator=(const Transistor &other);
21
22     // Virtual method for displaying device details
23     virtual string ToString() const override;
24
25     // Accessor methods
26     double GetBaseEmitterVoltage() const;
27     double GetCollectorCurrent() const;
28
29 protected:
30     double baseEmitterVoltage;
31     double collectorCurrent;
32 };
33
34 #endif
35

```

Transistor.cpp

```

1 #include "../Interfaces/Transistor.h"
2
3 #include <iostream>
4 #include <typeinfo>
5
6 Transistor::Transistor(const string &brand, const string &model, double baseEmitterVoltage,
7     double collectorCurrent)
8     : DiscreteSemiconductorDevice(brand, model), baseEmitterVoltage(baseEmitterVoltage),
9       collectorCurrent(collectorCurrent) {}
10
11 Transistor::Transistor(const Transistor &other)
12     : DiscreteSemiconductorDevice(other), baseEmitterVoltage(other.baseEmitterVoltage),
13       collectorCurrent(other.collectorCurrent) {}
14
15 Transistor &Transistor::operator=(const Transistor &other)
16 {
17     if (this != &other)
18     {
19         DiscreteSemiconductorDevice::operator=(other);
20         baseEmitterVoltage = other.baseEmitterVoltage;
21         collectorCurrent = other.collectorCurrent;
22     }
23     return *this;
24 }
25
26 Transistor::~Transistor() {}
27
28 double Transistor::GetBaseEmitterVoltage() const
29 {
30     return baseEmitterVoltage;
31 }
32
33 double Transistor::GetCollectorCurrent() const
34 {
35     return collectorCurrent;
36 }
37
38 string Transistor::ToString() const
39 {
40     string result = DiscreteSemiconductorDevice::ToString();
41     result += "\n\t\tBase-Emitter Voltage: " + to_string(baseEmitterVoltage) + " V";
42     result += "\n\t\tCollector Current: " + to_string(collectorCurrent) + " A";
43     return result;
44 }
45

```

Thyristor.h

```

1 #ifndef THYRISTOR_H
2 #define THYRISTOR_H
3
4 #include "DiscreteSemiconductorDevice.h"
5
6 #include <string>
7
8 using namespace std;
9
10 class Thyristor : public DiscreteSemiconductorDevice
11 {
12 public:
13     // Constructor and Destructor
14     Thyristor(const string &brand, const string &model, double gateTriggerVoltage, double
15         holdingCurrent);
16     virtual ~Thyristor();
17
18     // Copy Constructor and Assignment Operator
19     Thyristor(const Thyristor &other);
20     Thyristor &operator=(const Thyristor &other);
21
22     // Virtual method for displaying device details
23     virtual string ToString() const override;
24
25     // Accessor methods
26     double GetGateTriggerVoltage() const;
27     double GetHoldingCurrent() const;
28
29 protected:
30     double gateTriggerVoltage;
31     double holdingCurrent;
32 };
33
34 #endif
35

```

Thyristor.cpp

```

1 #include "../Interfaces/Thyristor.h"
2
3 #include <iostream>
4 #include <typeinfo>
5
6 Thyristor::Thyristor(const string &brand, const string &model, double gateTriggerVoltage,
7     double holdingCurrent)
8     : DiscreteSemiconductorDevice(brand, model), gateTriggerVoltage(gateTriggerVoltage),
9       holdingCurrent(holdingCurrent) {}
10
11 Thyristor::Thyristor(const Thyristor &other)
12     : DiscreteSemiconductorDevice(other), gateTriggerVoltage(other.gateTriggerVoltage),
13       holdingCurrent(other.holdingCurrent) {}
14
15 Thyristor &Thyristor::operator=(const Thyristor &other)
16 {
17     if (this != &other)
18     {
19         DiscreteSemiconductorDevice::operator=(other);
20         gateTriggerVoltage = other.gateTriggerVoltage;
21         holdingCurrent = other.holdingCurrent;
22     }
23     return *this;
24 }
25
26 Thyristor::~Thyristor() {}
27
28 double Thyristor::GetGateTriggerVoltage() const
29 {
30     return gateTriggerVoltage;
31 }
32
33 double Thyristor::GetHoldingCurrent() const
34 {
35     return holdingCurrent;
36 }
37
38 string Thyristor::ToString() const
39 {
40     string result = DiscreteSemiconductorDevice::ToString();
41     result += "\n\t\tGate Trigger Voltage: " + to_string(gateTriggerVoltage) + " V";
42     result += "\n\t\tHolding Current: " + to_string(holdingCurrent) + " A";
43     return result;
44 }
45

```

```

CourseProject.cpp
CourseProject.cpp > ...
1  #include <iostream>
2  #include <typeinfo>
3
4  #include "../Data/Interfaces/DiscreteSemiconductorDevice.h"
5  #include "../Data/Entities/DiscreteSemiconductorDevice.cpp"
6  #include "../Data/Interfaces/Diode.h"
7  #include "../Data/Entities/Diode.cpp"
8  #include "../Data/Interfaces/Transistor.h"
9  #include "../Data/Entities/Transistor.cpp"
10 #include "../Data/Interfaces/Thyristor.h"
11 #include "../Data/Entities/Thyristor.cpp"
12 #include "../Data/Interfaces/ZenerDiode.h"
13 #include "../Data/Entities/ZenerDiode.cpp"
14 #include "../Data/Interfaces/TunableZenerDiode.h"
15 #include "../Data/Entities/TunableZenerDiode.cpp"
16
17 using namespace std;
18
19 int main()
20 {
21     DiscreteSemiconductorDevice *devices[6];
22
23     devices[0] = new DiscreteSemiconductorDevice("NXP", "BC547");
24     devices[1] = new Diode("NXP", "BC557", 0.7, 0.1);
25     devices[2] = new ZenerDiode("Philips", "BZX55C5V1", 0.7, 0.05, 5.1, 0.03);
26     devices[3] = new TunableZenerDiode("Sony", "TZD100", 0.6, 0.04, 10.0, 0.02, 5, 1.0);
27     devices[4] = new Transistor("Motorola", "2N3904", 0.6, 0.2);
28     devices[5] = new Thyristor("STMicroelectronics", "T1234", 1.0, 1.5);
29
30     // Demonstrating copy constructor
31     Transistor transistorCopy = *(static_cast<Transistor *>(devices[2]));
32     Thyristor thyristorCopy = *(static_cast<Thyristor *>(devices[3]));
33
34     // Showing all device info
35     for (int i = 0; i < 6; i++)
36     {
37         cout << devices[i]->ToString() << endl;
38         if (auto di = dynamic_cast<DiscreteSemiconductorDevice *>(devices[i]))
39         {
40             cout << "\nAccessor Methods: " << endl;
41             cout << "Brand: " << di->GetBrand() << endl;
42             cout << "Model: " << di->GetModel() << endl;
43         }
44         else if (auto di = dynamic_cast<Diode *>(devices[i]))
45         {
46             cout << "\nAccessor Methods: " << endl;
47             cout << "Forward Voltage: " << di->GetForwardVoltage() << " V" << endl;
48             cout << "Max Current: " << di->GetMaxCurrent() << " A" << endl;
49             if (auto zd = dynamic_cast<ZenerDiode *>(devices[i]))
50             {
51                 cout << "Reverse Breakdown Voltage: " << zd->GetReverseBreakdownVoltage() << " V" << endl;
52                 cout << "Temperature Coefficient: " << zd->GetTemperatureCoefficient() << " %/C" << endl;
53                 if (auto tzd = dynamic_cast<TunableZenerDiode *>(devices[i]))
54                 {
55                     cout << "Control Voltage: " << tzd->GetControlVoltage() << " V" << endl;
56                     cout << "Adjustment Range: " << tzd->GetAdjustmentRange() << " V" << endl;
57                 }
58             }
59         }
60         else if (auto tr = dynamic_cast<Transistor *>(devices[i]))
61         {
62             cout << "\nAccessor Methods: " << endl;
63             cout << "Base-Emitter Voltage: " << tr->GetBaseEmitterVoltage() << " V" << endl;
64             cout << "Collector Current: " << tr->GetCollectorCurrent() << " A" << endl;
65         }
66         else if (auto th = dynamic_cast<Thyristor *>(devices[i]))
67         {
68             cout << "\nAccessor Methods: " << endl;
69             cout << "Gate Trigger Voltage: " << th->GetGateTriggerVoltage() << " V" << endl;
70             cout << "Holding Current: " << th->GetHoldingCurrent() << " A" << endl;
71         }
72     }
73
74     // Demonstrating the copy's independence
75     cout << "\n\nCopy constructor examples:" << endl;
76     cout << "\n\nOriginal Transistor:\n"
77         << devices[4]->ToString() << endl;
78     cout << "\n\nCopy of Transistor:\n"
79         << transistorCopy.ToString() << endl;
80
81     for (int i = 0; i < 6; i++)
82     {
83         if (devices[i])
84         {
85             delete devices[i];
86         }
87     }
88     return 0;
89 }

```

```
-----
Device Details:
  Class: DiscreteSemiconductorDevice
  Brand: NXP
  Model: BC547
```

```
Accessor Methods:
Brand: NXP
Model: BC547
```

```
-----
Device Details:
  Class: Diode
  Brand: NXP
  Model: BC557
    Forward Voltage: 0.700000 V
    Maximum Current: 0.100000 A
```

```
Accessor Methods:
Brand: NXP
Model: BC557
```

```
-----
Device Details:
  Class: ZenerDiode
  Brand: Philips
  Model: BZX55CSV1
    Forward Voltage: 0.700000 V
    Maximum Current: 0.050000 A
    Reverse Breakdown Voltage: 5.100000 V
    Temperature Coefficient: 0.030000 %/C
```

```
Accessor Methods:
Brand: Philips
Model: BZX55CSV1
```

```
-----
Device Details:
  Class: TunableZenerDiode
  Brand: Sony
  Model: TZD100
    Forward Voltage: 0.600000 V
    Maximum Current: 0.040000 A
    Reverse Breakdown Voltage: 10.000000 V
    Temperature Coefficient: 0.020000 %/C
    Control Voltage: 5.000000 V
    Adjustment Range: 1.000000 V
```

```
Accessor Methods:
Brand: Sony
Model: TZD100
```

```
-----
Device Details:
  Class: Transistor
  Brand: Motorola
  Model: 2N3904
    Base-Emitter Voltage: 0.600000 V
    Collector Current: 0.200000 A
```

```
Accessor Methods:
Brand: Motorola
Model: 2N3904
```

```
-----
Device Details:
  Class: Thyristor
  Brand: STMicroelectronics
  Model: T1234
    Gate Trigger Voltage: 1.000000 V
    Holding Current: 1.500000 A
```

```
Accessor Methods:
Brand: STMicroelectronics
Model: T1234
```

```
+ Copy constructor examples:
```

```
Original Transistor:
```

```
-----
Device Details:
  Class: Transistor
  Brand: Motorola
  Model: 2N3904
    Base-Emitter Voltage: 0.600000 V
    Collector Current: 0.200000 A
```

```
Copy of Transistor:
```

```
-----
Device Details:
  Class: Transistor
  Brand: Philips
  Model: BZX55CSV1
    Base-Emitter Voltage: 0.700000 V
    Collector Current: 0.050000 A
```

Всичко може да бъде видяно на тук:

[GitHub](#)