AAB University

Faculty of Computer Sciences

Object Oriented Programming

Java 4:

Hyrje në Klasa dhe Objekte

Asst. Prof. Dr. **Mentor Hamiti**

mentor.hamiti@universitetiaab.com

Orën e kaluar?!

- Struktura e programit
- Varablat
- Konceptet e memorjes
- Aritmetika
- Marrja e vendimeve (ang. Decision Making)
- .....

Sot

- Hyrje në Klasa dhe Objekte
- Definimi i Klasës me Anëtarin e Funksionit
- Definimi i Anëtarit të Funksionit me një Parametër
- Anëtarët e të dhënave, Funksionet set dhe get
- Inicializimi i Objekteve me Konstruktorë
- Vendosja e një Klase në një File të veçantë për përdorim

Hyrje

- Konceptet themelore të Programimit të Orientuar në Objekte janë **Klasat** dhe **Objektet**.

- Zakonisht, programet e shkruara në C ++ janë të përbërë nga një funksion kryesor **main** dhe një ose më shumë klasa.

- Çdo Klasë përmban:

  - **Anëtarë të të dhënave (ang. data members)** dhe

  - **Anëtarë të funksioneve (ang. member functions)**

- Hyrje në Klasa dhe Objekte
- Definimi i Klasës me Anëtarin e Funksionit
- Definimi i Anëtarit të Funksionit me një Parametër
- Anëtarët e të dhënave, Funksionet set dhe get
- Inicializimi i Objekteve me Konstruktorë
- Vendosja e një Klase në një File të veçantë për përdorim

Shembulli 1

```
//Example 1
#include <iostream>
using namespace std;


// GradeBook class definition
class GradeBook
{
    public:
        // function that displays a welcome message to the GradeBook user
        void displayMessage()
            {
                cout << "Welcome to the Grade Book!" << endl;
            }
}; // end class GradeBook


int main()
{
    GradeBook myGradeBook;           // create a GradeBook object named myGradeBook
    myGradeBook.displayMessage();    // call object's displayMessage function

    return 0;
}
```

```
Welcome to the Grade Book!
```

Definimi i Klasës me Anëtarin e Funksionit

- Përkufizimi i Klasës fillon me fjalën **class** i ndjekur nga emri i Klasës **GradeBook**
  - Emri i një klase i përcaktuar nga shfrytëzuesi fillon me shkronjë të madhe, dhe për lexueshmëri më të mirë çdo fjalë e mëvonshme në emrin e klasës fillon me një shkronjë të madhe.
- Çdo klasë definohet me kllapa **{** dhe **}** .
- Fundi i Klasës përfundon me pikëpresje **;**

Common Programming Error 3.1

Forgetting the semicolon at the end of a class definition is a syntax error.

Definimi i Klasës me Anëtarin e Funksionit

- Funksioni **main** gjithmonë thirret automatikisht kur ekzekutohet programi.
- Shumica e funksioneve nuk thirren automatikisht.
- Ju duhet të thirrni anëtarin e funksionit **displayMessage** në mënyrë eksplicite apo të qartë për ti treguar programit se duhet të kryejë detyrën e saj.
- Fjala **Public:** është një specifikues i qasjes (ang. access specifier) e cila:
  - Tregon se funksioni është "në dispozicion të publikut" që do të thotë se ajo mund të thirret nga funksionet e tjera në program (**main**), dhe nga anëtarët e funksioneve të klasave tjera.
  - Specifiesi i qasjes (ang. access specifier) gjithmonë ndiqet nga dy pika (:).

Definimi i Klasës me Anëtarin e Funksionit

- Çdo funksion në program kryen një detyrë dhe mund të kthejë vlerë kur të përfundojë me detyrën e saj.

```
// function that displays a welcome message to the GradeBook user
void displayMessage()
    {
        cout << "Welcome to the Grade Book!" << endl;
    }
```

- Fjala **void** në të majtë të emrit të funksionit **displayMessage** është tipi kthyes i funksionit (ang. return type)
  - Tregon se **displayMessage** nuk do të kthejë ndonjë të dhënë për funksionin e thirrur kur të përfundojë detyrën e saj.
- Shkronja e parë e funksionit shkruhet me shkronjë të vogël.

Definimi i Klasës me Anëtarin e Funksionit

```
// function that displays a welcome message to the GradeBook user
void displayMessage()
    {
        cout << "Welcome to the Grade Book!" << endl;
    }
```

- Kllapat pas emrit të anëtarit të funksionit tregojnë se ai është **funksion**
  - Kllapat bosh tregojnë se anëtari i funksionit nuk kërkon të dhëna shtesë për të kryer detyrën e saj.

- Rreshti i parë i përkufizimit të funksionit zakonisht quhet funksioni header.
- Çdo funksion është i përkufizuar me kllapa **{** dhe **}** .
- Funksioni përmban ngjarje që kryejnë detyrën e funksionit.

Testimi i Klasës GradeBook

- Nuk mund të thirret një anëtar i funksionit të një klase deri sa të krijohet një **objekt** i asaj **klase**.

- Së pari, krijojmë një objekt të klasës GradeBook të quajtur myGradeBook

  - Tipi i variablës është GradeBook

  - Kompajleri automatikisht nuk e di se çfarë tipi është GradeBook

  - I tregojmë kompajlerit çfarë është GradeBook duke përfshirë edhe definimin e klasës.

  - Çdo klasë që krijojmë bëhet tip ri që mund të përdoret për të krijuar objekte.

Testimi i Klasës GradeBook

- Krijohet **Objekt** i klasës GradeBook i quajtur **myGradeBook**

```
GradeBook myGradeBook;
myGradeBook.displayMessage();
```

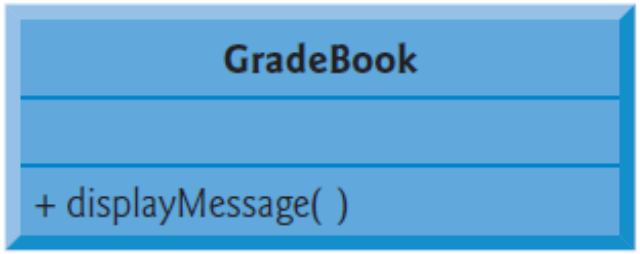
- Thirret anëtari i funksionit displayMessage - duke përdorur variablën **myGradeBook** i ndjekur nga operatori **.** (pikë), emrin e funksionit **displayMessage** dhe kllapat **()**

- Shfaqet funksioni displayMessage për të kryer detyrën e saj.

```
Welcome to the Grade Book!
```

UML Klasë Diagrami për Klasën GradeBook

- Në UML, çdo klasë është modeluar në diagram të klasës UML si një drejtkëndësh me tri ndarje:



  - **Emri i klasës** i përqendruar horizontalisht dhe me font Bold.
  - **Atributet e klasës**, të cilat korrespondojnë me anëtarët e të dhënave në C ++
    - Aktualisht është bosh, sepse klasa GradeBook ende nuk ka atribute
  - **Operacionet e klasës**, të cilat korrespondojnë me anëtarët e funksioneve në C ++

- Shenja plus + para emrit të operacionit tregon se displayMessage është një operacion publik në UML

- Hyrje në Klasa dhe Objekte
- Definimi i Klasës me Anëtarin e Funksionit
- Definimi i Anëtarit të Funksionit me një Parametër
- Anëtarët e të dhënave, Funksionet set dhe get
- Inicializimi i Objekteve me Konstruktorë
- Vendosja e një Klase në një File të veçantë për përdorim

Shembulli 2

```
Please enter the course name:
CS101 Introduction to C++ Programming

Welcome to the grade book for
CS101 Introduction to C++ Programming!
```

```cpp
//Example 2
#include <iostream>
#include <string>       // program uses C++
using namespace std;

// GradeBook class definition
class GradeBook
{
public:
    // function that displays a welcome message to the GradeBook user
    void displayMessage( string courseName )
    {
        cout << "Welcome to the grade book for\n" << courseName << "!"<< endl;
    }
};

int main()
{
    string nameOfCourse;    // string of characters to store the course name
    GradeBook myGradeBook;  // create a GradeBook object named myGradeBook

    cout << "Please enter the course name:" << endl;
    getline(cin, nameOfCourse ); // read a course name with blanks
    myGradeBook.displayMessage(nameOfCourse);

    return 0;
}
```

Definimi i Anëtarit të Funksionit me një Parametër

```
#include <string>
```

- Variabla e tipit **string** paraqet një varg të karaktereve.

- String është në fakt një objekt i vargut të klasës së bibliotekës standarde në C ++

  - Përcaktohet në header file-in **<string>** dhe është pjesë e **namespace std**
  - Tani për tani, mund të mendoni për variablën string si variabël e tipeve tjera të tilla si **int**

Definimi i Anëtarit të Funksionit me një Parametër

```
getline(cin, nameOfCourse );
```

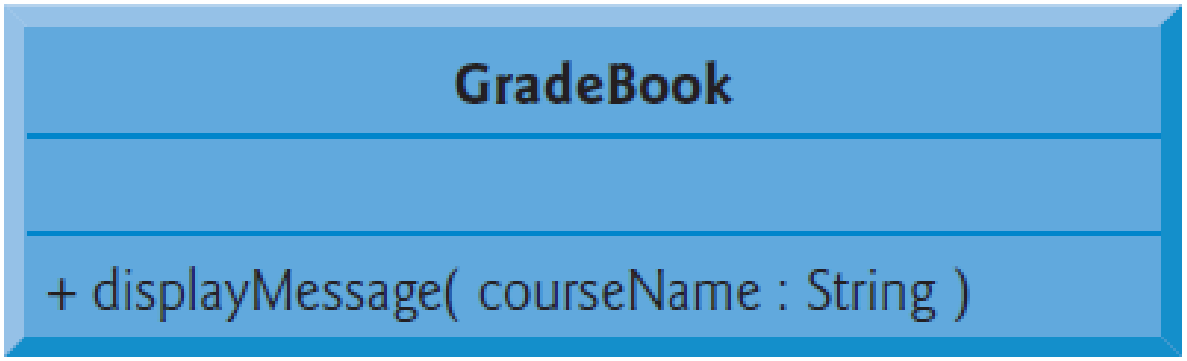
- Getline funksioni lexon një rresht të tekstit në një varg (string)
  - Thirrja e funksionit **getline (cin, nameOfCourse)** lexon karakteret (duke përfshirë hapësirat me karaktere që ndajnë fjalët në input) nga **cin** (keyboard) .
  - Kur shtypet Enter përderisa insertohen të dhëna , rresht i ri është insertuar në input stream.
- Header file-i <**String**> duhet të përfshihet në program për të përdorur funksionin **getline**

Definimi i Anëtarit të Funksionit me një Parametër

- Funksionet me parametra kërkojnë të dhëna për të kryer detyrën e tyre
  - Lista me parametra mund të përmbajë më shumë parametra, por mundet mos përmbajë asnjë parametër, me këtë tregon se një funksion nuk kërkon asnjë parametër hyrës.
  - Çdo parametër duhet të specifikojë tipin dhe identifikatorin e tij.
  - Një funksion mund të specifikojë parametra të shumta duke ndarë çdo parametër me një presje.
  - Numri dhe rradhitja e argumenteve në një thirrje të funksionit duhet të përputhet me numrin dhe rradhitjen e parametrave të listës së parametrave në header-in e thirrur të anëtarit të funksionit.

UML Klasë Diagrami për Klasën GradeBook (Sh.2)

- UML ka tipe të veta të të dhënave të ngjashme me ato të C ++
  - UML modelon një parametër duke listuar apo rradhitur emrin e parametrit, e ndjekur nga dy pika dhe tipi i parametrit në kllapa pas emrit të operacionit.



- UML është gjuhë e pavarur
  - Përdoret në shumë gjuhë të ndryshme programuese, kështu që terminologjia e saj nuk përputhet saktësisht me atë të C ++

- Hyrje në Klasa dhe Objekte
- Definimi i Klasës me Anëtarin e Funksionit
- Definimi i Anëtarit të Funksionit me një Parametër
- Anëtarët e të dhënave, Funksionet set dhe get
- Inicializimi i Objekteve me Konstruktorë
- Vendosja e një Klase në një File të veçantë për përdorim

Anëtarët e të dhënave, Funksionet set dhe get

- Variablat e deklaruara në funksion janë të njohur si variabla lokale dhe mund të përdoren vetëm nga rreshti i deklaruar në funksion deri në mbyllje të saj, të kllapës mbyllëse } i bllokut në të cilën ata janë deklaruar

  - Një variabël lokale duhet të deklarohet para se ajo të përdoret në një funksion

  - Një variable lokale nuk mund ti qaset nga jashtë funksionit në të cilin ajo është deklaruar

  - Kur një funksion përfundon, vlerat e variablës së saj lokale janë humbur

Anëtarët e të dhënave, Funksionet set dhe get

- Një objekt ka atribute që janë realizuar me të si ajo e përdorur në program
  - Atributet e tilla ekzistojnë në të gjithë jetën e objektit
  - Një klasë normalisht përbëhet nga një ose më shumë anëtarë të funksioneve që manipulon me atribute që i përkasin një objekti të veçantë të klasës
- Atributet janë të përfaqësuara si variabla në një përkufizim të klasës
  - Variabla të tilla janë quajtur anëtarë të të dhënave (ang. data members) dhe janë deklaruar brenda një definimi të klasës por jashtë përkufizimit të klasës së anëtarit të funksionit
- Çdo objekt i një klase mban atributet e veta në memorje.

Anëtarët e të dhënave, Funksionet set dhe get

- Një variabël që është deklaruar në definicionin e klasës por jashtë përkufizimit të klasës së anëtarit të funksionit është **anëtar i të dhënave (ang. data member).**

- Çdo instancë (dmth, objekt) i një klase përmban secilin anëtarë të të dhënave të klasës.

- Përfitimi i ndryshimit nga variabël në anëtar të të dhënave është se të gjithë anëtarët e funksionit të klasës mund të manipulojnë me anëtarë të të dhënave që shfaqen në definimin e klasës.

Shembulli 3

```
//Example 3
#include <iostream>
#include <string> // program uses C++ standard string class
using namespace std;

// GradeBook class definition
class GradeBook
{
    public:
        // function that sets the course name
        void setCourseName( string name )
        {
            courseName = name; // store the course name in the object
        } // end function setCourseName

        // function that gets the course name
        string getCourseName()
        {
            return courseName; // return the object's courseName
        } // end function getCourseName

    // function that displays a welcome message
    void displayMessage()
    {
        // this statement calls getCourseName to get the
        // name of the course this GradeBook represents
        cout << "Welcome to the grade book for\n" <<getCourseName() << "!"
        << endl;
    } // end function displayMessage

    private:
        string courseName; // course name for this GradeBook
}; // end class GradeBook
```

Shembulli 3 (vazhd.)

```
// function main begins program execution
int main()
{
    string nameOfCourse; // string of characters to store the course name
    GradeBook myGradeBook; // create a GradeBook object named myGradeBook

    // display initial value of courseName
    cout << "Initial course name is: " <<myGradeBook.getCourseName()
        << endl;

    // prompt for, input and set course name
    cout << "\nPlease enter the course name:" << endl;
    getline( cin, nameOfCourse ); // read a course name with blanks

    myGradeBook.setCourseName( nameOfCourse ); // set the course name
    cout << endl; // outputs a blank line

    myGradeBook.displayMessage(); // display message with new course name
} // end main
```

```
Initial course name is:

Please enter the course name:
CS101 Introduction to C++ Programming

Welcome to the grade book for
CS101 Introduction to C++ Programming!
```

Anëtarët e të dhënave, Funksionet set dhe get

- Shumica e deklarimeve të anëtarit me të dhëna shfaqet pas specifikuesit të qasjes **private:**
- Ashtu si **public**, fjala **private** është një specifikues i qasjes.
- Variablat ose funksionet e deklaruar pas specifikuesit të qasjes private (dhe para specifikuesit të qasjes së ardhëshme) janë të arritshme vetëm me anëtarin e funksionit të klasës për të cilin janë deklaruar.
- Të gjithë anëtarët e klasës janë të definuara si private nga sistemi.
- Specifikuesit e qasjes publike dhe private mund të përsëriten, por kjo është e panevojshme dhe mund të jetë konfuze.

Anëtarët e të dhënave, Funksionet set dhe get

- Deklarimi i anëtarit me të dhëna me specifikuesin e qasjes **private** është i njohur si të **dhëna të fshehur (ang. data hiding).**

- Kur një program krijon një objekt, anëtarët e saj me të dhëna janë **përmbledhur** (**të fshehura)** në objekt dhe mund të arrihen vetëm nga funksionet anëtare të klasës së objektit.

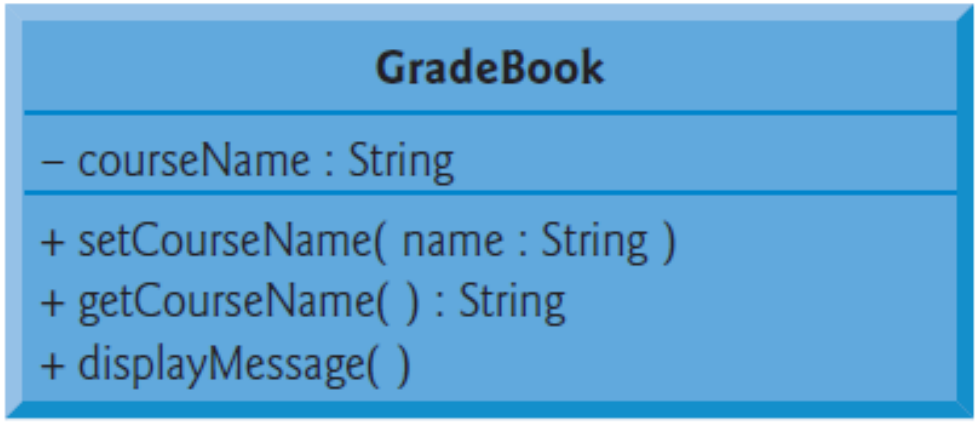
- Klasat shpesh japin funksione me anëtarë publikë për të lejuar klientët e klasës të vendosin - **set** (dmth, të caktojë vlerat në) ose të marrin - **get** (p.sh., të marrë vlerat e) anëtarë me të dhëna private.

  - Këto emra të anëtarit të funksionit nuk duhet të fillojnë me **set** ose **get**, por kjo traditë e emërtimit është i zakonshëm.

  - Funksionet set ndonjëherë quhen **mutators** (sepse ata shndrrohen, ose ndryshohen), dhe get funksionet ndonjëherë quhen **accessors** (sepse ata ju qasen vlerave).

UML Klasë Diagrami për Klasën GradeBook (Sh.3)

- UML klasë diagrami për klasën GradeBook me një atribut privat courseName dhe operacionet publike setCourseName, getCourseName dhe displayMessage



- *UML (Unified Modeling Language) përfaqëson anëtarët e të dhënave si atribute duke rradhitur emrin atribut, e ndjekur nga dy pika dhe tipin e atributit.*

- Hyrje në Klasa dhe Objekte
- Definimi i Klasës me Anëtarin e Funksionit
- Definimi i Anëtarit të Funksionit me një Parametër
- Anëtarët e të dhënave, Funksionet set dhe get
- Inicializimi i Objekteve me Konstruktorë
- Vendosja e një Klase në një File të veçantë për përdorim

Inicializimi i Objekteve me Konstruktorë

- Çdo klasë mund të sigurojë një ose më shumë **konstruktorë** që mund të përdoren për inicializim të një objekti të klasës kur objekti është i krijuar.

- Një konstruktor është një anëtar i funksionit të veçantë që duhet të përcaktohet me të **njëjtin emër** si **klasa**, kështu që kompajleri mund ta dallojë atë nga anëtarët e funksioneve tjera të klasës.

- Një dallim i rëndësishëm mes konstruktorëve dhe funksioneve të tjera është se konstruktorët *nuk mund të kthejnë vlera*, kështu që ata nuk mund të specifikojnë tipe kthyese (as edhe **void**)

- Konstruktorët janë deklaruar **publik**

Inicializimi i Objekteve me Konstruktorë

- C ++ automatikisht thërret një konstruktor për çdo objekt që është krijuar, e cila ndihmon për tu siguruar që objektet janë inicializuar siç duhet para se ata të përdoren në program.

- Thirrja e konstruktorit ndodh kur objekti është krijuar.

- Nëse një klasë nuk përfshin konstruktorët, kompajleri ofron një konstruktor të parazgjedhur (default constructor) pa parametra.

Shembulli 4

```
class GradeBook
{
public:
    // constructor initializes courseName with string supplied as argument
    GradeBook ( string name )
    {
        setCourseName( name );  // call set function to initialize courseName
    } // end GradeBook constructor

    // function to set the course name
    void setCourseName( string name )
    {
        courseName = name; // store the course name in the object
    } // end function setCourseName

    // function to get the course name
    string getCourseName()
    {
    return courseName; // return object's courseName
    } // end function getCourseName

    // display a welcome message to the GradeBook user
    void displayMessage()
    {
    // call getCourseName to get the courseName
        cout << "Welcome to the grade book for\n" <<getCourseName()<< "!" << endl;
    } // end function displayMessage

private:
    string courseName; // course name for this GradeBook
}; // end class GradeBook
```

Shembulli 4 (vazhd.)

```
//Example_4
# include <iostream>
# include <string>
using namespace std;

// function main begins program execution
int main()
{
    // create two GradeBook objects
    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
    GradeBook gradeBook2( "CS102 Data Structures in C++" );
    // display initial value of courseName for each GradeBook
    cout << "gradeBook1 created for course: " <<gradeBook1.getCourseName()
        << "\ngradeBook2 created for course: " <<gradeBook2.getCourseName()
        << endl;
} // end main
```

```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

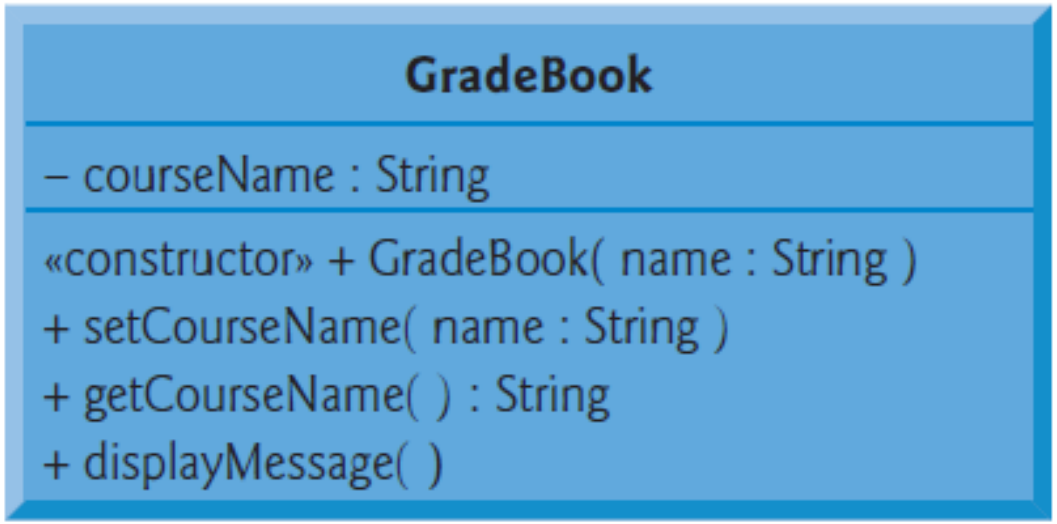
Inicializimi i Objekteve me Konstruktorë

- Një konstruktor specifikon në listën e tij me parametra të dhënat që nevoiten për të kryer detyrën e tij.
- Kur krijohet një objekt i ri, të dhënat vendosen në kllapa që ndjekin emrin e objektit.
- Çdo konstruktor që nuk merr argumente quhet konstruktor i paracaktuar (default).
- Një klasë merr një konstruktor të parazgjedhur në disa mënyra:
  - Kompajleri në mënyrë implicite krijon një konstruktor të përzgjedhur në çdo klasë që nuk ka konstruktor të definuar nga user-i apo shfrytëzuesi.
  - Në mënyrë eksplicite apo të qartë mund të definoni konstruktor që nuk kanë argumente.
  - Nëse përcaktohet ndonjë konstruktor me argumente, C ++ nuk do të krijojë një konstruktor të paracaktuar për atë klasë.

UML Klasë Diagrami për Klasën GradeBook (Sh.4)

- Ashtu si operacionet, UML modelon konstruktorët në pjesën e tretë të klasës në diagramin e klasës.

- Për të dalluar një konstruktor nga operacionet e një klase, UML vendos fjalën " constructor " mes « dhe » para emrit të konstruktorit.



- Hyrje në Klasa dhe Objekte
- Definimi i Klasës me Anëtarin e Funksionit
- Definimi i Anëtarit të Funksionit me një Parametër
- Anëtarët e të dhënave, Funksionet set dhe get
- Inicializimi i Objekteve me Konstruktorë
- Vendosja e një Klase në një File të veçantë për përdorim

Vendosja e një Klase në një File të veçantë për përdorim

- Një nga përfitimet e krijimit të definimit të klasës është se, kur të paketohen siç duhet, klasat tona mund të ripërdoren nga programuesit - potencialisht në gjithë botën.

- Programuesit të cilët dëshirojnë të përdorin klasën tonë GradeBook nuk mund thjesht të përfshijë një file nga një tjetër program

  - Funksioni kryesor apo main fillon ekzekutimin e çdo programi, dhe çdo program duhet të ketë pikërisht një funksion kryesor.

- Kur ndërtohet një program në object-oriented C ++, është e zakonshme që të përcaktohet burimi i kodit për ripërdorim në një file që ka ekstenzionin **.h filename** - i njohur si **header**

- Programet përdorin direktivat për procesim # include për të përfshirë header file dhe të përfitojnë nga komponentët për ripërdorim softuerik.

Shembulli 5 (vazhd.)

```
//GradeBook.h File
# include <iostream>
# include <string>
using namespace std;
// GradeBook class definition
class GradeBook
{
public:
    // constructor initializes courseName with string supplied as argument
    GradeBook ( string name )
    {
        setCourseName( name );  // call set function to initialize courseName
    } // end GradeBook constructor

    // function to set the course name
    void setCourseName( string name )
    {
        courseName = name; // store the course name in the object
    } // end function setCourseName

    // function to get the course name
    string getCourseName()
    {
    return courseName; // return object's courseName
    } // end function getCourseName

    // display a welcome message to the GradeBook user
    void displayMessage()
    {
    // call getCourseName to get the courseName
        cout << "Welcome to the grade book for\n" <<getCourseName()<< "!" << endl;
    } // end function displayMessage
private:
    string courseName; // course name for this GradeBook
}; // end class GradeBook
```

Shembulli 5

```
//Example_5
# include <iostream>
# include "GradeBook.h"
using namespace std;

// function main begins program execution
int main()
{
    // create two GradeBook objects
    GradeBook gradeBook1( "CS101 Introduction to C++ Programming" );
    GradeBook gradeBook2( "CS102 Data Structures in C++" );
    // display initial value of courseName for each GradeBook
    cout << "gradeBook1 created for course: " <<gradeBook1.getCourseName()
        << "\ngradeBook2 created for course: " <<gradeBook2.getCourseName()
        << endl;
} // end main
```

```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

- **#include** direktiva udhëzon paraprocesorin C ++ të zëvendësojë direktivën me një kopje të përmbajtjes së **GradeBook.h** para se programi të kompajlohet.

- Kur file-i i burimit të kodit është kompajluar, ajo përmban definimin e klasës GradeBook *(për shkak të #include*), dhe kompajleri është në gjendje që të përcaktojë se si ti krijojë objektet GradeBook dhe të shohë se a janë thirrur saktë anëtarët e funksioneve të tyre.

- Tani që përkufizimi i klasës është në header file-in (pa funksion kryesor apo main), ne mund të përfshijmë atë header në çdo program që do të kemi nevojë për ripërdorim të klasës tonë GradeBook.

Vendosja e një Klase në një File të veçantë për përdorim

- Vini re se emri i file-it të **GradeBook.h** header-it është mbyllur në thonjëza “ ” në vend të kllapave <>
  - File-et e kodit të programit dhe file-et header të definuar nga user-i apo shfrytëzuesi janë të vendosur në të njëjtin dierktorium.
  - Kur paraprocesori takon emrin e header file-it në thonjëza, ajo tenton të gjejë header file-in në të njëjtin direktorium të file-it në të cilin shfaqet direktiva #include
  - Nëse paraprocesori nuk mund të gjejë header file-in në atë direktorium, ajo kërkon vendin e header file-it të C ++ Library Standard.
  - Kur paraprocesori takon emrin e një header file-i në kllapa këndore (p.sh., <iostream>), ajo supozon se header-i është pjesë e *C++ Standard Library* dhe nuk kërkon në direktoriumin e programit që është duke u paraprocesuar.

- Pyetje?!

mentor.hamiti@universitetiaab.com