Algoritmalar ve Programlama II – BLM 102

Hafta 2: Sınıflar ve Objelere Giriş

Fenerbahçe Üniversitesi

Öğretim Elemanları

Öğretim Üyesi: Dr. Vecdi Emre Levent

Ofis: 311

Email: emre.levent@fbu.edu.tr

Asistan: Arş. Gör. Uğur Özbalkan

Ofis: 311

Email: ugur.ozbalkan@fbu.edu.tr

Ders Planı

- Sınıflar ve Objelere Giriş
  - Sınıf tanımlaması
  - Üyelik fonksiyonları tanımları
  - Veri üyeleri, ayarlama (set) ve alma (get) fonksiyonları
  - Objeleri yapıcı (constructor) ile ilk değerini atama (initialize)
  - Tekrar kullanılabilirlik için (reusability) sınıfı farklı bir dosyaya taşıma
  - Arayüzü tasarımdan ayırma

Sınıf Tanımlaması

- Özellikle büyük bir yazılımın geliştirme sürecinde çok faydalı olan nesneye yönelimli programlama yaklaşımı kullanılmaktadır.
- Doğru tasarlandığında, kodun tekrar kullanılabilir olmasını sağlamaktadır.
- C dilinden farklı olarak sınıf kavramı kullanılmaktadır.
- Sınıf kavramı ile özel objeler yaratılacaktır. (C dilindeki struct'lar gibi)
- Ders kapsamında sınıflar anlatılırken GradeBook isimli bir yazılımın tasarımı üzerinden anlatılacaktır. Bu yazılım temel olarak, bir öğretmenin öğrencilerinin test sonuçlarını görüntüleyip, düzenleyebildiği bir yazılımdır. Bu yazılım adım adım gerçeklenecektir.

Sınıf Tanımlaması

```
#include <iostream>
using std::cout;
using std::endl;

class GradeBook
{
public:
    void displayMessage()
    {
        cout << "Welcome to the Grade Book!" << endl;
    }
};

int main()
{
    GradeBook myGradeBook;
    myGradeBook.displayMessage();
    return 0;
}
```

Sınıf Tanımlaması

- Sınıf (class) tanımlaması, sınıf isminin başına class ifadesi yazılarak yapılmaktadır.
- Bir sınıf'ın içerisinde değişkenler ve fonksiyonlar olabilir.
- Fonksiyonlara, üye fonksiyonlar (member functions)
- Değişkenlere, veri üyeleri (data members) denmektedir.

Sınıf Tanımlaması

```
#include <iostream>
using std::cout;
using std::endl;

class GradeBook
{
public:
    void displayMessage()
    {
        cout << "Welcome to the Grade Book!" << endl;
    }
};

int main()
{
    GradeBook myGradeBook;
    myGradeBook.displayMessage();
    return 0;
}
```

Erişim iznini belirtir
Public olduğunda, public'in altındaki değişken ve fonksiyonlar sınıfın dışından (diğer tüm sınıflar ve fonksiyonlardan) da kullanılabilir demektir.

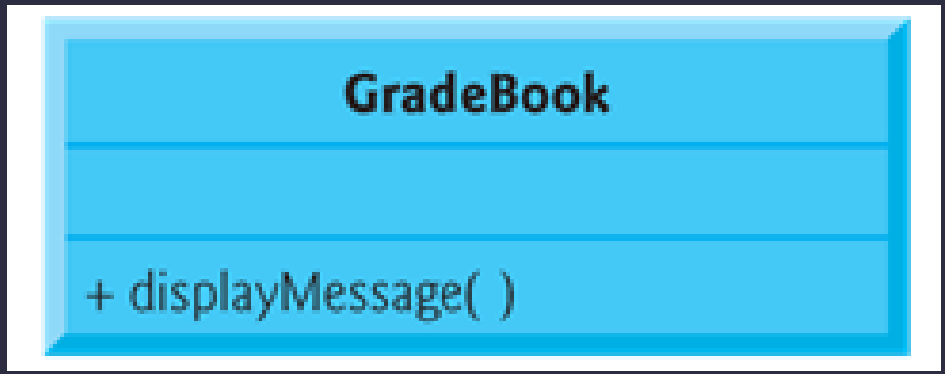
Sınıf Tanımlaması

```
#include <iostream>
using std::cout;
using std::endl;

class GradeBook
{
public:
    void displayMessage()
    {
        cout << "Welcome to the Grade Book!" << endl;
    }
};

int main()
{
    GradeBook myGradeBook;
    myGradeBook.displayMessage();
    return 0;
}
```

Okunabilirliğin daha iyi olması için, fonksiyon ve değişken tanımlarında ilk harfin küçük, eğer kullanılacak isim birden fazla kelime içeriyorsa her bir kelimenin ilk harfi büyük kullanılmalıdır.

Sınıf Tanımlaması

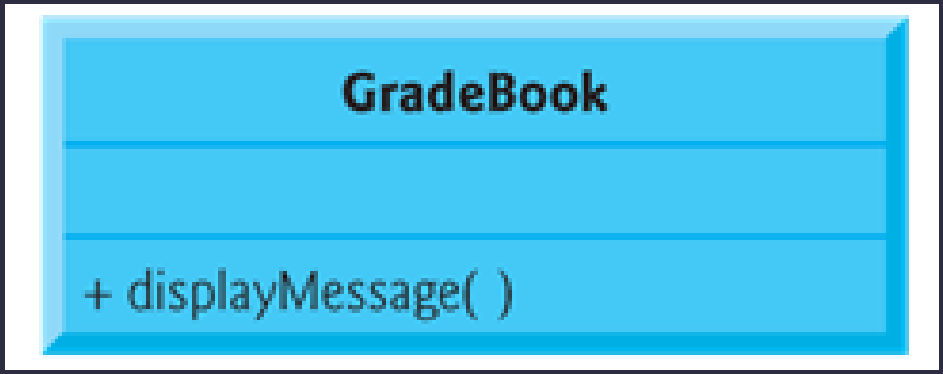
- Bir sınıftaki fonksiyonların ve değişkenlerin neler olduğunu görsel olarak ifade eden UML (Unified Modeling Language) isminde bir dil bulunmaktadır.



GradeBook UML Gösterimi

- En üstte kalın harfler ile yazılan kısım, sınıf ismini belirtmektedir.
- Ortadaki bölüm sınıfın içerisindeki değişkenlerini (veri üyelikleri – data members) ifade etmektedir.
- En alttaki bölüm ise üyelik fonksiyonlarını ifade etmektedir.

Sınıf Tanımlaması



GradeBook UML Gösterimi

- Üyelik fonksiyonunun yanındaki + işareti, o fonksiyonunun public olduğunu gösterir.
- – işareti olduğunda ise o fonksiyonunun, private olduğunu ifade etmektedir.
- (Private'nin ne olduğu daha sonra açıklanacaktır)

Sınıf Tanımlaması

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

#include <string>
using std::string;
using std::getline;

class GradeBook
{
public:
void displayMessage(string courseName)
{
cout << "Welcome to the grade book for\n" <<
courseName << "!"
<< endl;
}
};
```

```
int main()
{
string nameOfCourse;
GradeBook myGradeBook;

cout << "Please enter the course name:" << endl;
getline(cin, nameOfCourse);
cout << endl;

myGradeBook.displayMessage(nameOfCourse);
return 0;
}
```

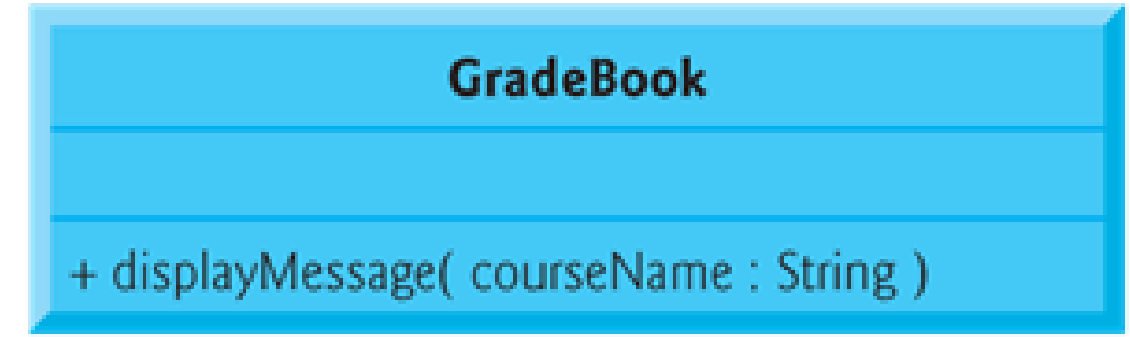
Sınıf Tanımlaması

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

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

Sınıf Tanımlaması

- Bir metinin tutulması için önceden char dizisi kullanılıyordu (C dersinde)
- C++ dilinde metinler ile işlem yapmak için string isminde bir kütüphane bulunmaktadır.
- <string> kütüphanesinin eklenmesi ile kullanılabilir.
- getline fonksiyonu tanımlanmış olan bir string'in içine giriş alınmasını sağlar (\n karakterini sonuna eklemez). <string> kütüphanesinin bir üyesidir.

Sınıf Tanımlaması



GradeBook UML Gösterimi

Sınıf Tanımlaması

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;

#include <string>
using std::string;
using std::getline;

class GradeBook
{
public:
void setCourseName(string name)
{
courseName = name;
}

string getCourseName()
{
return courseName;
}

void displayMessage()
{

cout << "Welcome to the grade book for\n" <<
getCourseName() << "!" << endl;
}
private:
string courseName;
};
```

Sınıf Tanımlaması

```
int main()
{
string nameOfCourse;
GradeBook myGradeBook;

cout << "Initial course name is: " <<
myGradeBook.getCourseName() << endl;

cout << "\nPlease enter the course name:" <<
endl;
getline(cin, nameOfCourse);
myGradeBook.setCourseName(nameOfCourse);

cout << endl;
myGradeBook.displayMessage();
return 0;
}
```

Sınıf Tanımlaması

```
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!
```

Sınıf Tanımlaması

- public gibi private'de bir sınıfıdaki elemanın erişim izninin ne olduğunu gösteren bir ifadedir.
- private tanımlanmış olan tüm değişken ve fonksiyonlar sadece o sınıfın içerisinde tanımlanmış olan fonksiyonlar tarafından erişilebilirler. Varsayılan tanım private'dir. Bir tanım yapılmadığında private olarak tanımlanmış olarak çalışacaktır.
- Genellikle, değişkenler (veri üyeleri) private, fonksiyonlar (üyelik fonksiyonları) public tanımlanır.
- Bu yaklaşım, debugging'i kolaylaştırmaktadır. Değişkenleri bir fonksiyon üzerinden değiştirmek veya değişkenin değerini fonksiyon aracılığı ile alma yaklaşımı;
  - Erişim yönteminde bir hata varsa, sadece fonksiyonda değişiklik yapmak yeterli olacaktır.
  - Örn. İnsan isminde bir sınıftaki isim string değişkenine, 123 değeri yazılsın. İsim olduğu için, 123 değerinin aslında yazılmaması gerekmektedir. İsimYaz isminde bir fonksiyon tanımlanıp, adlığı parametrenin içinde sayı var mı yok mu kontrolü yaparak yazarsa, bu sorunu aşmış olur.
- Private tanımlanmış olan bir değişkene, sınıf dışından erişim yapılması halinde derleme hatası alınır.

Sınıf Tanımlaması

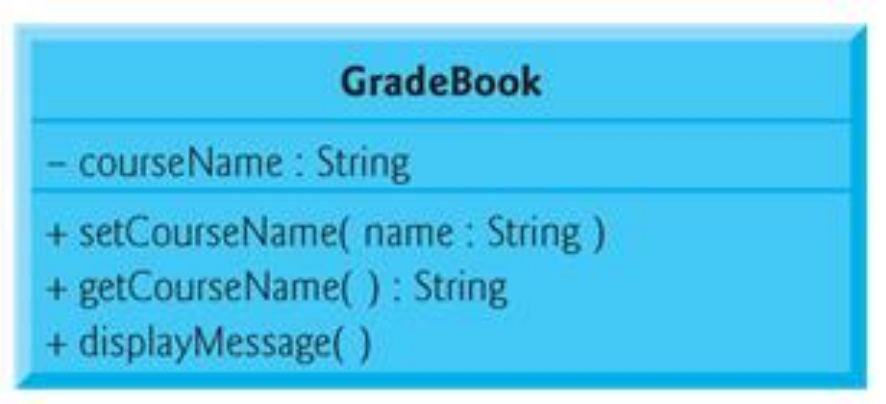
- Bir değişken veya fonksiyonun private olarak tanımlanması veri gizleme olarak tanımlanır.

Özetle

- set fonksiyonları;
  - Private tanımlanmış değişkenlerin değerlerini değiştirmek için
- get fonksiyonları;
  - Private tanımlanmış değişkenlerin değerlerini almak için

kullanılmaktadır.

Sınıf Tanımlaması



GradeBook UML Gösterimi

Objeleri Yapıcı (Constructor) ile İlk Değerini Atama (İnitialize)

- Her bir sınıf, içerisinde barındırdığı değişkenlerin başlangıç değerlerini, constructor denen mekanizma ile gerçekleştirebilir.

- Constructor özel bir fonksiyondur. Tanımlanmış olan bir sınıftan yeni bir obje yaratıldığında bir kere çalıştırılır.

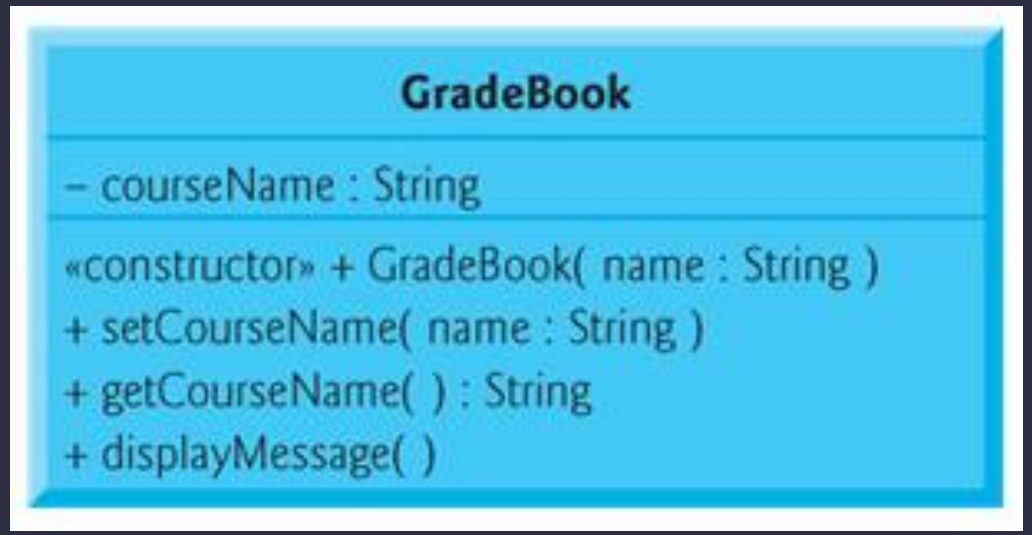
Objeleri Yapıcı (Constructor) ile İlk Değerini Atama (İnitialize)

```
#include <iostream>
using std::cout;
using std::endl;

#include <string>
using std::string;

class GradeBook
{
public:
GradeBook(string name)
{
setCourseName(name);
}
void setCourseName(string name)
{
courseName = name;
}

string getCourseName()
{
return courseName;
}

void displayMessage()
{
cout << "Welcome to the grade book for\n" <<
getCourseName() << "!" << endl;
}
private:
string courseName;
};
```

Objeleri Yapıcı (Constructor) ile İlk Değerini Atama (İnitialize)

```
int main()
{
GradeBook gradeBook1("CS101 Introduction to C++
Programming");
GradeBook gradeBook2("CS102 Data Structures in
C++");

cout << "gradeBook1 created for course: " <<
gradeBook1.getCourseName()
<< "\ngradeBook2 created for course: " <<
gradeBook2.getCourseName() << endl;
return 0;
}
```

Sınıf Tanımlaması



GradeBook UML Gösterimi

Tekrar Kullanılabilirlik İçin (Reusability) Sınıfı Farklı Bir Dosyaya Taşıma

- Daha önceki tüm örnekler tek bir .c/.cpp uzantılı dosyalar ile yapıldı.
- Bir sınıfın, diğer uygulamalar tarafından kullanılabilmesi için ayrı bir dosyada olması gerekmektedir.
- Nesneye yönelimli bir C++ uygulaması geliştirirken, .cpp dosyasının yanında .h uzantılı bir kütüphane dosyası da geliştirilmesi gerekmektedir.
- Bu .h uzantılı kütüphane dosyası, diğer programlar tarafından include kullanılarak kendi kodlarına dahil edilebilirler.

Tekrar Kullanılabilirlik İçin (Reusability) Sınıfı Farklı Bir Dosyaya Taşıma

- GradeBook yazılımı şu anki hali ile doğrudan diğer programcılar tarafından kullanılması zordur.
- Büyük yazılım geliştirme süreçlerinde programı sınıflara ayırırken, geliştirilen sınıflar farklı dosyalarda tutulacaktır.
- Bunun için GradeBook yazılımı, gradebook.h ve main.cpp isimli iki dosyaya ayrılmıştır.

Tekrar Kullanılabilirlik İçin (Reusability) Sınıfı Farklı Bir Dosyaya Taşıma

```
#include <iostream>
using std::cout;
using std::endl;

#include <string>
using std::string;

class GradeBook
{
public:
GradeBook(string name)
{
setCourseName(name);
}

void setCourseName(string name)
{
courseName = name;
}

string getCourseName()
{
return courseName;
}

void displayMessage()
{
cout << "Welcome to the grade book for\n" <<
getCourseName()
<< "!" << endl;
}
private:
string courseName;
};
```

Tekrar Kullanılabilirlik İçin (Reusability) Sınıfı Farklı Bir Dosyaya Taşıma

```
#include <iostream>
using std::cout;
using std::endl;

#include "GradeBook.h"

int main()
{
GradeBook gradeBook1("CS101 Introduction to C++
Programming");
GradeBook gradeBook2("CS102 Data Structures in
C++");

cout << "gradeBook1 created for course: " <<
gradeBook1.getCourseName()
<< "\ngradeBook2 created for course: " <<
gradeBook2.getCourseName()
<< endl;
return 0;
}
```

Tekrar Kullanılabilirlik İçin (Reusability) Sınıfı Farklı Bir Dosyaya Taşıma

- GradeBook.h dosyasının içerisinde main fonksiyonu yoktur. Dolayısıyla kendi başına çalışamaz.
- Bu kütüphanenin kullanılması için, içerisinde main fonksiyonu barındıran bir yazılım kullanılmalıdır.
- Bu kodun içerisinde include ifadesi ile gradebook.h dosyası include edilmelidir.
- Bu yaklaşımın avantajı, kütüphaneki fonksiyonları içeriği değişse bile, tasarımcının mainde kullandığı fonksiyonu çağırma biçimi değişmediğinde, kütüphaneyi kullanan kişinin kodunu değiştirmesine ihtiyaç yoktur.

Arayüzü Tasarımdan Ayırma

- Arayüzler genel olarak, bir sistemden diğer bir sisteme bilgi aktarırken kullanılan tanım ve standartların tamamına denmektedir.

- Bir sınıf'ın arayüzü ise, o sınıfı kullanacak programcıya sınıfın içerisindeki değişken ve fonksiyonların neler olduğu, fonksiyonların hangi argümanları alıp ne geriye döndürdüğünü söyleyen bilgidir.

- Arayüz dosyalarında fonksiyonların içerisinde yapılan işlemler gösterilmez.

- Bunun nedeni, tasarımın gizliliğinin korunmasıdır. Sınıf derlenerek kütüphane ve .h uzantılı dosya ile diğer programcılara servis edildiğinde, tasarımın iç detaylı programcılara açılmamış olacaktır.

Arayüzü Tasarımdan Ayırma

- Tasarlanan sınıfın ismi, dosya adı ile aynı verilmelidir.

- Örn GradeBook sınıfı tasarımında, dosya isimleri

  - GradeBook.cpp
  - GradeBook.h

konulmalıdır.

Arayüzü Tasarımdan Ayırma

```
#include <string>
using std::string;

class GradeBook
{
public:
GradeBook(string);
void setCourseName(string);
string getCourseName();
void displayMessage();
private:
string courseName;
};
```

GradeBook.h

Arayüzü Tasarımdan Ayırma

- .h uzantılı dosyada tanımlanmış olan fonksiyon parametrelerinde, argümanların türünün yazılması yeterlidir. Ancak .h uzantılı dosyaya bakan bir yazılımcı, o kütüphaneyi daha iyi anlayabilmesi için, istenirse argümanın türünün yanına bir değişken ismi de yazılabilir.

- Yazılan değişken isminin bir önemi yoktur, derleyici tarafından gözardı edilir.

Arayüzü Tasarımdan Ayırma

```cpp
#include <iostream>
using std::cout;
using std::endl;

#include "GradeBook.h"

GradeBook::GradeBook(string name)
{
setCourseName(name);
}

void GradeBook::setCourseName(string name)
{
courseName = name;
}

string GradeBook::getCourseName()
{
return courseName;
}
```

```cpp
void GradeBook::displayMessage()
{
cout << "Welcome to the grade book for\n" <<
getCourseName() << "!" << endl;
}
```

GradeBook.cpp

Arayüzü Tasarımdan Ayırma

- GradeBook.cpp dosyasında kullanılan :: işareti, .h uzantılı dosyada tanımlanmış olan fonksiyon prototiplerinin ile .cpp uzantılı dosyada yazılmış olan fonksiyon içeriklerini birbirine bağlar.
- .cpp uzantılı dosyanın başına include ile .h uzantılı dosyanın eklenmesi gerekmektedir.
- Böylelikle, derleyici, o sınıfa ait bir fonksiyon mu değil mi kontrolü yapabilir ve .h uzantılı dosyada tanımlanmış olan değişkenler, .cpp uzantılı dosyadaki fonksiyon içeriklerinde kullanılabilir hale gelmektedir.

Arayüzü Tasarımdan Ayırma

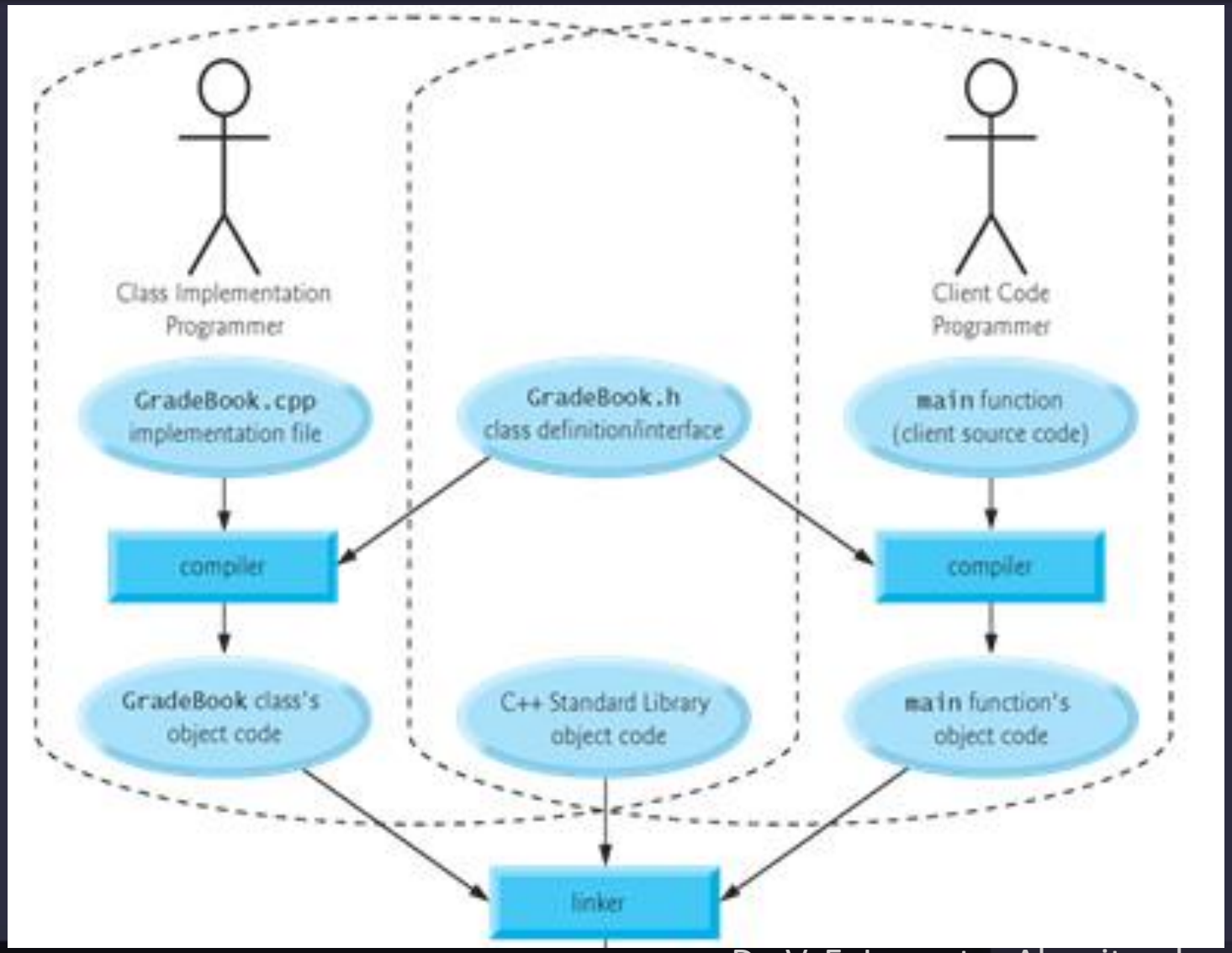
```
#include <iostream>
using std::cout;
using std::endl;

#include "GradeBook.h"

int main()
{
GradeBook gradeBook1("CS101 Introduction to C++
Programming");
GradeBook gradeBook2("CS102 Data Structures in C++");

cout << "gradeBook1 created for course: " <<
gradeBook1.getCourseName()
<< "\ngradeBook2 created for course: " <<
gradeBook2.getCourseName()
<< endl;
return 0;
}
```

main.cpp

Arayüzü Tasarımdan Ayırma

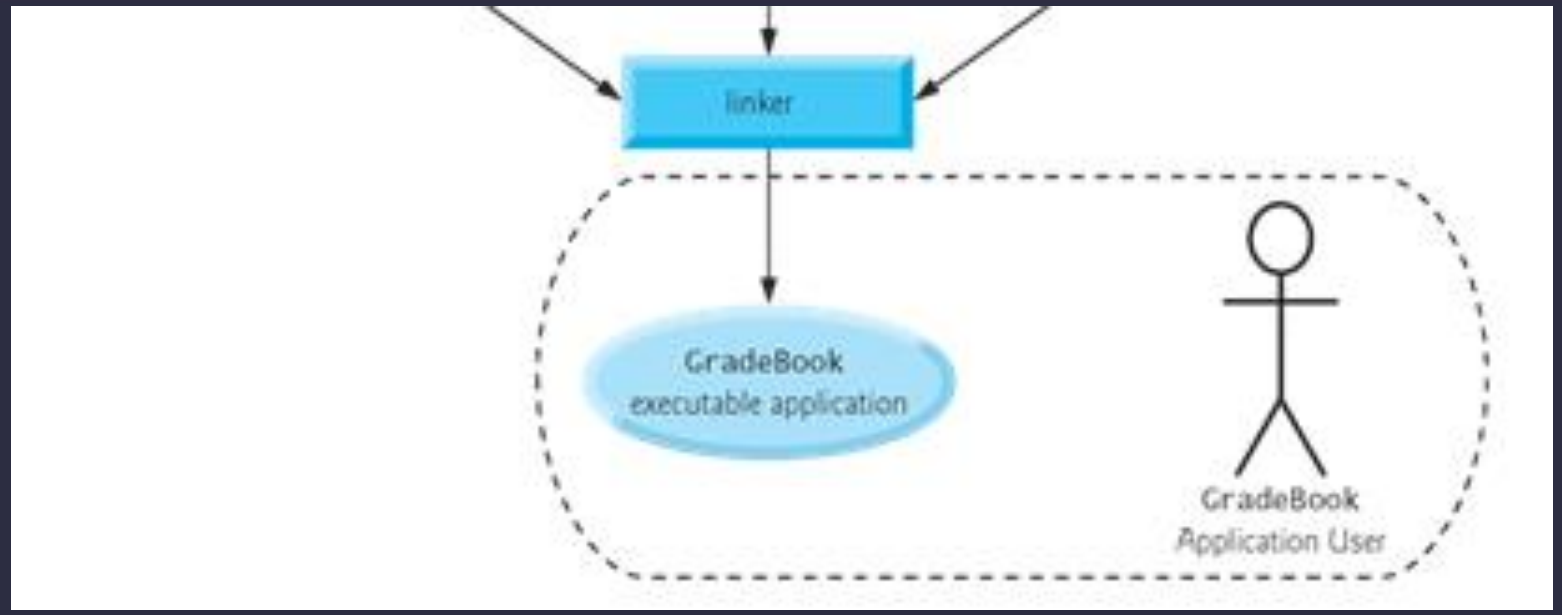
```
gradeBook1 created for course: CS101 Introduction to C++ Programming
gradeBook2 created for course: CS102 Data Structures in C++
```

Arayüzü Tasarımdan Ayırma



- Kütüphane kullanılmadan önce kaynak kodların derlenmesi ve birbirlerine bağlanması (linking) gerekmektedir.

Arayüzü Tasarımdan Ayırma



- Bağlama işleminden sonra çalıştırılabilir uygulama oluşturulacaktır.

Arayüzü Tasarımdan Ayırma

```
void GradeBook::setCourseName(string name)
{
courseName = name;
}
```

```
void GradeBook::setCourseName(string name)
{
if (name.length() <= 25) // if name has 25 or fewer
characters
courseName = name; // store the course name in the
object

if (name.length() > 25) // if name has more than 25
characters
{
// set courseName to first 25 characters of parameter
name
courseName = name.substr(0, 25); // start at 0,
length of 25

cout << "Name \"" << name << "\" exceeds maximum
length (25).\n"
<< "Limiting courseName to first 25 characters.\n" <<
endl;
} // end if
}
```

- Değişiklik yapıldığında, fonksiyonların giriş çıkışlarında değişiklik yoksa, kütüphanenin kullanıldığı main.cpp dosyasında bir değişiklik yapılmasına gerek yoktur.