Algoritmalar ve Programlama II – BLM 102

Hafta 9: Çok Biçimlilik (Polymorphism)

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

- Çok Biçimlilik
- Sanal Fonksiyonlar
- Saf Sanal Fonksiyonlar
- Soyut Sınıflar

Çok Biçimlilik (Polymorphism)

- Çok biçimlilik, genel olarak bir nesnenin farklı durumlar altında farklı davranışlar sergilemesidir.

- Örneğin, bir insan hem baba, hem eş hem de çalışan olabilir. Bu insan'ın konuşma hareketi, farklı durumlarda farklı olacaktır.

Çok Biçimlilik (Polymorphism)

- Objeye yönelik programlama dillerinde polymorphism, aynı isimli fonksiyonların birden çok gerçeklemesinin olabileceğidir.

- Daha önceki anlatılan konularda, bir fonksiyon'un (örneğin constructor) birden fazla tanımı olabileceği anlatılmıştı.

Çok Biçimlilik (Polymorphism)

```
#include <iostream>
using namespace std;

class deneme
{

public:

deneme(string abc)
{
cout << abc << "\n";
}

deneme(int a, int b)
{
cout << a << " " << b << "\n";
}
};
```

```
int main()
{

deneme obj1("test");
deneme obj2(3, 5);

return 0;
}
```

Çıktı:

```
Test
3 5
```

Çok Biçimlilik (Polymorphism)

```
#include <iostream>
using namespace std;

class deneme
{

public:

void func(string abc)
{
cout << abc << "\n";
}

void func(int a, int b)
{
cout << a << " " << b << "\n";
}
void func(string a, int b)
{
cout << a << " " << b << "\n";
}
};

int main()
{
deneme obj1;

obj1.func("test");
obj1.func(3, 5);
obj1.func("ABC", 10);

return 0;
}
```

Çıktı:

```
test
3 5
ABC 10
```

Çok Biçimlilik (Polymorphism)

- Operatör aşıyı yüklemesi de bir çeşit polymorphism'dir.

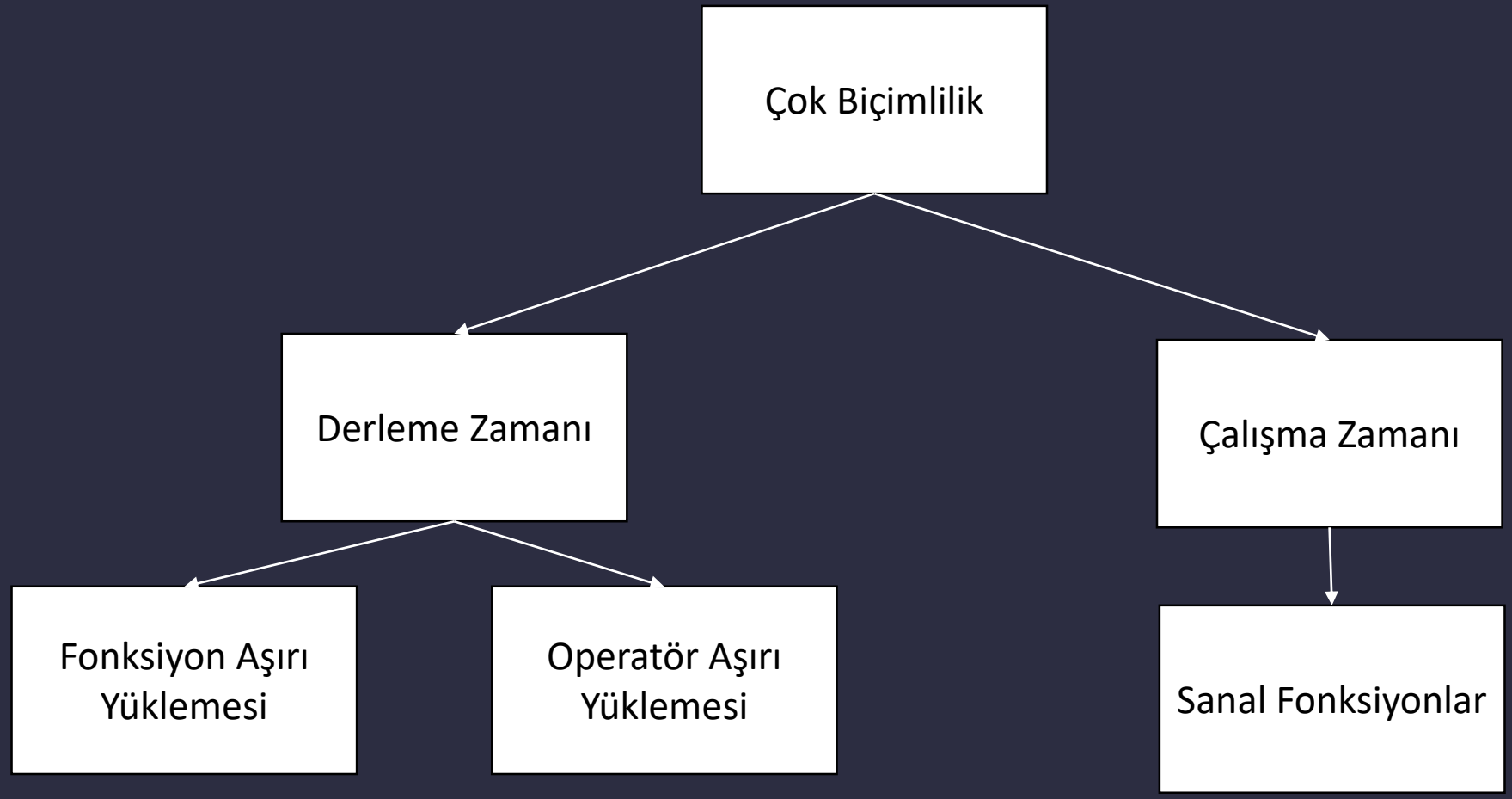
```
#include<iostream>
using namespace std;

class deneme {
private:
int x;
public:
deneme(int arg = 0) {
x = arg;
}

int operator + (int tmp) {
return x + tmp;
}
};

int main()
{
deneme obj1(3);
cout << obj1 + 5 << "\n";
return 0;
}
```

Çok Biçimlilik (Polymorphism)

- Bu tür polymorphism operasyonları derleme zamanında yapılmaktadır.

- Derleyici uygulama derlenirken, hangi fonksiyonun nereden çağıracağını ayarlamaktadır.

Çok Biçimlilik (Polymorphism)



Çok Biçimlilik (Polymorphism)

- Çalışma zamanı polymorphism özellikleri, bir base sınıfından başka bir sınıf kalıtım yapıldığında kullanılmaktadır.

Çok Biçimlilik (Polymorphism)

```
#include <iostream>
using namespace std;

class sekiller {
protected:
int en, boy;

public:
sekiller(int a = 0, int b = 0) {
en = a;
boy = b;
}
};

class dikdortgen : public sekiller {
public:
dikdortgen(int a = 0, int b = 0) :sekiller(a, b) { }

int alan() {
cout << "Dikdortgen alani donduruluyor" << endl;
return (en * boy);
}
};
```

```
class ucken : public sekiller {
public:
ucken(int a = 0, int b = 0) :sekiller(a, b) { }

int alan() {
cout << "Ucken alani donduruluyor." << endl;
return (en * boy / 2);
}
};

int main() {

sekiller* obj1;
dikdortgen obj2(10, 7);
ucken  obj3(10, 5);

obj1 = &obj2;
obj1->alan();

obj1 = &obj3;
obj1->alan();

return 0;
}
```

Derleme Hatası alınır

Çok Biçimlilik (Polymorphism)

- Base sınıfına alan fonksiyonu eklendiğinde ise;

Çok Biçimlilik (Polymorphism)

```
#include <iostream>
using namespace std;

class sekiller {
protected:
int en, boy;

public:
sekiller(int a = 0, int b = 0) {
en = a;
boy = b;
}
int alan() {
cout << "Base sinifi alan hesaplama fonksiyonu calisti!" <<
endl;
return 0;
}
};

class dikdortgen : public sekiller {
public:
dikdortgen(int a = 0, int b = 0) :sekiller(a, b) { }
int alan() {
cout << "Dikdortgen alani donduruluyor" << endl;
return (en * boy);
}
};

class ucken : public sekiller {
public:
ucken(int a = 0, int b = 0) :sekiller(a, b) { }
int alan() {
cout << "Ucken alani donduruluyor." << endl;
return (en * boy / 2);
}
};

int main() {
sekiller* obj1;
dikdortgen obj2(10, 7);
ucken  obj3(10, 5);
obj1 = &obj2;
obj1->alan();
obj1 = &obj3;
obj1->alan();
return 0;
}
```

Çıktı:

```
Base sinifi alan hesaplama fonksiyonu calisti!
Base sinifi alan hesaplama fonksiyonu calisti!
```

Çok Biçimlilik (Polymorphism)

- sekiller sınıfının pointer'ından çağrıldığında base sınıfının içerisindeki alan fonksiyonu çağrılmaktadır.

- Bunun sebebi, bu sınıftaki alan fonksiyonunun nereyi çağıracağı derleme zamanında sabitlenmesidir.

Çok Biçimlilik (Polymorphism)

```
#include <iostream>
using namespace std;

class sekiller {
protected:
int en, boy;

public:
sekiller(int a = 0, int b = 0) {
en = a;
boy = b;
}
virtual int alan() {
cout << "Base sinifi alan hesaplama fonksiyonu calisti!" <<
endl;
return 0;
}
};

class dikdortgen : public sekiller {
public:
dikdortgen(int a = 0, int b = 0) :sekiller(a, b) { }
int alan() {
cout << "Dikdortgen alani donduruluyor" << endl;
return (en * boy);
}
};

class ucken : public sekiller {
public:
ucken(int a = 0, int b = 0) :sekiller(a, b) { }
int alan() {
cout << "Ucken alani donduruluyor." << endl;
return (en * boy / 2);
}
};

int main() {
sekiller* obj1;
dikdortgen obj2(10, 7);
ucken  obj3(10, 5);
obj1 = &obj2;
obj1->alan();
obj1 = &obj3;
obj1->alan();
return 0;
}
```

Çıktı:

```
Dikdortgen alani
donduruluyor
Ucken alani
donduruluyor.
```

Çok Biçimlilik (Polymorphism)

- Base sınıf ve kalıtımlı sınıfta aynı isimli fonksiyon bulunduğunda ve kalıtımlı sınıftaki fonksiyon kullanılması istendiğinde, ana sınıftaki fonksiyonun başına virtual ifadesi yazılır.

- Derleyici pointer'in içeriğine bakarak, alan fonksiyonunu kalıtımlı sınıflardan çağırdı.

Çok Biçimlilik (Polymorphism)

- Virtual fonksiyonlar ile derleyiciye, sabit bir bağlama işlemi yapılmaması istendiği söylenmektedir.
- Yani, virtual tanımlanmış olan fonksiyondan türetilmiş olan sınıflardan istendiğinde çağrılacağı, dinamik olarak bağlanacağı söylenmektedir.,
- Artık fonksiyonun tipi olan sekiller sınıfına değil, içeriğine bakılacaktır.

Çok Biçimlilik (Polymorphism)

```
#include <iostream>
using namespace std;

class anaSinif
{
public:
virtual void func1()
{
cout << "Test 1" << endl;
}

void func2()
{
cout << "Test 2" << endl;
}
};

class turetilmisSinif :public anaSinif
{
public:
void func1()
{
cout << "Test 3" << endl;
}

void func2()
{
cout << "Test 4" << endl;
}
};

//main function
int main()
{
anaSinif* obj1;
turetilmisSinif obj2;
obj1 = &obj2;

obj1->func1();
obj1->func2();

return 0;
}
```

Çıktı:

```
Test 3
Test 2
```

Çok Biçimlilik (Polymorphism)

- Dinamik olarak, çalışma zamanında, bir fonksiyon nereyi çağıracağının belirtilmesine "late binding" denmektedir.

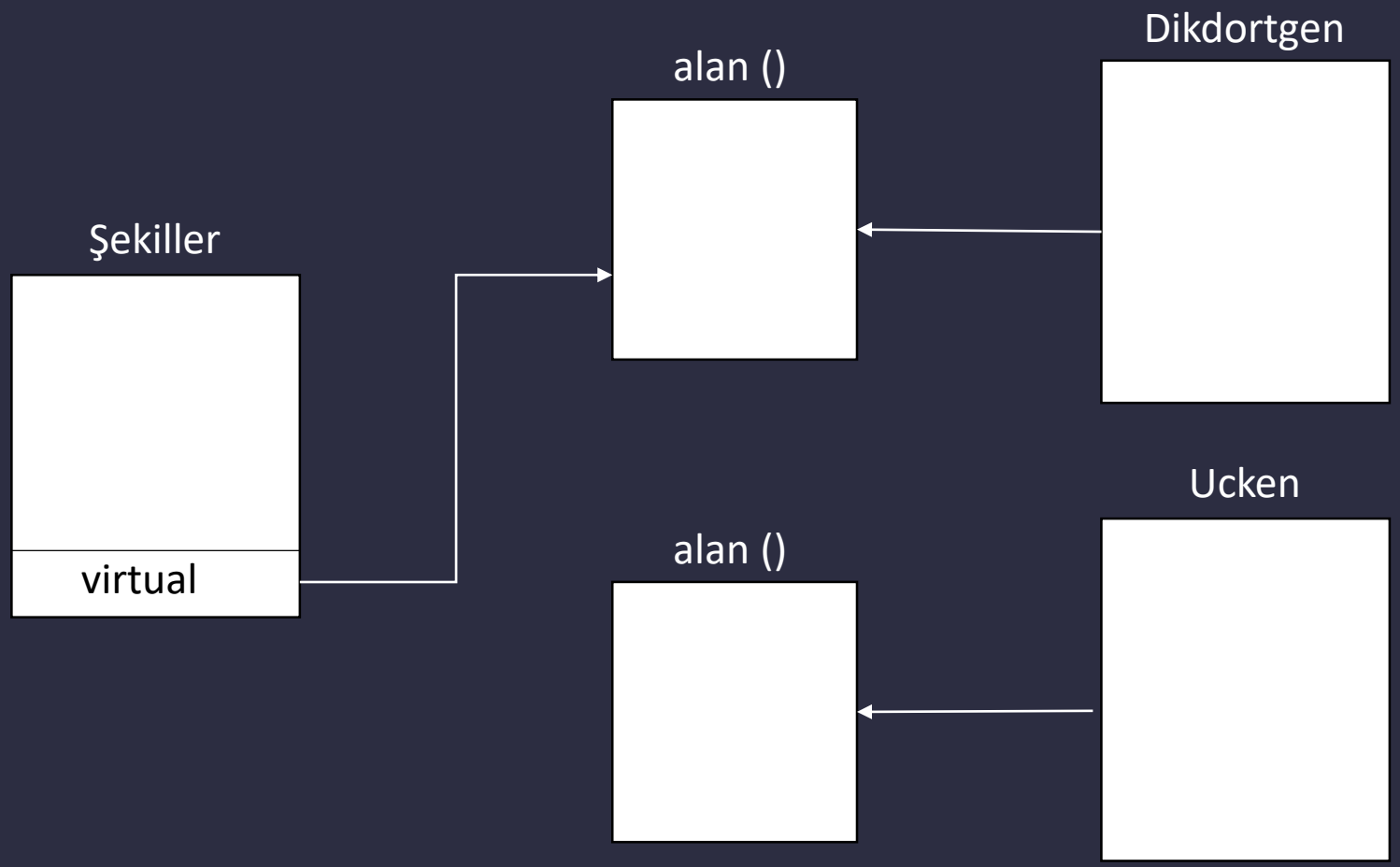
Çok Biçimlilik (Polymorphism)

- Derleyiciler, sınıfta virtual bir fonksiyon tanımı gördüklerinde arka planda virtual method table (VMT) isminde bir dizi eklerler.
- Bu tabloda fonksiyonların nereyi gösterecekleri adresler, yani pointer'lar tutulmaktadır.
- Program derlenirken bu tablo boştur.

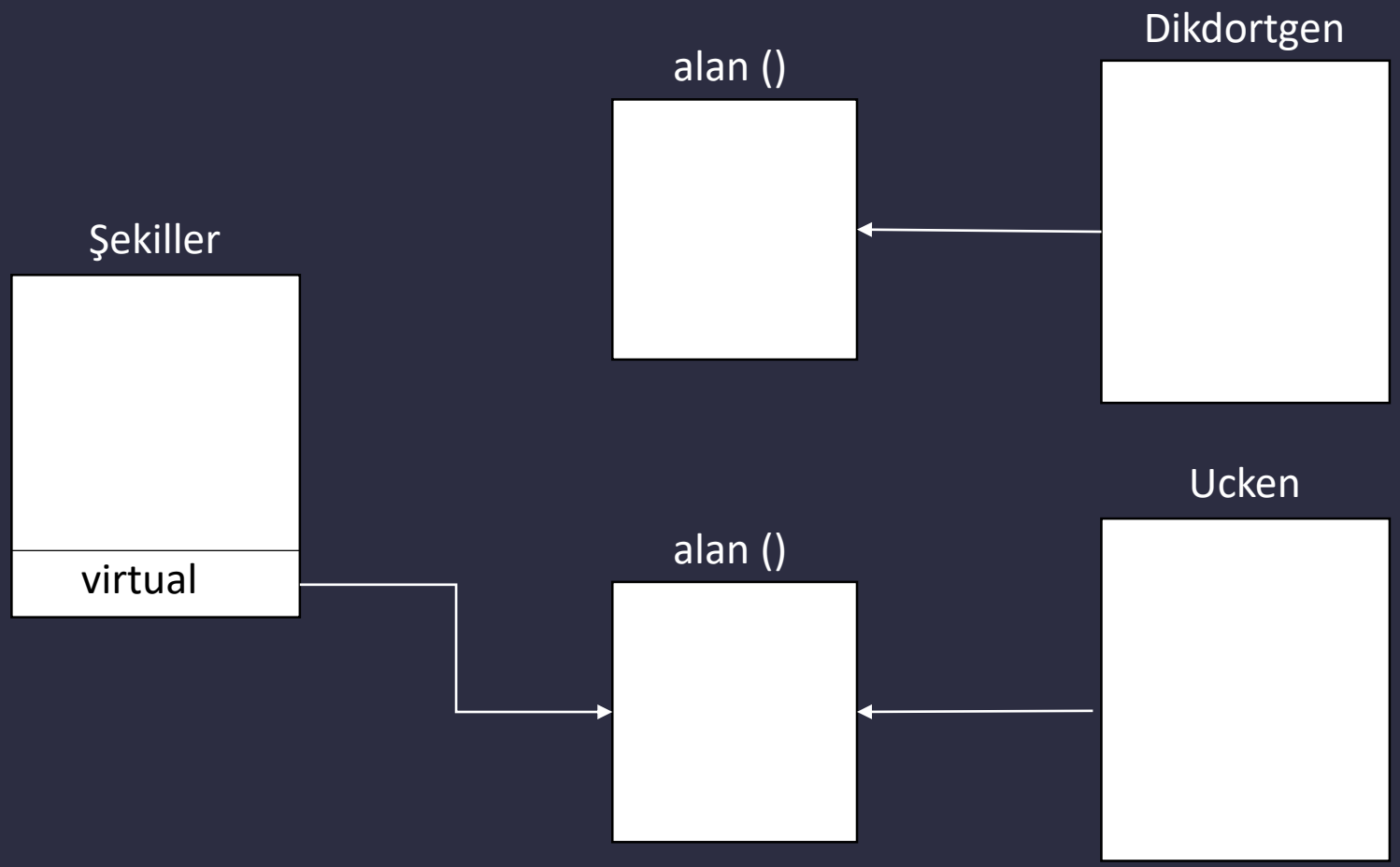
Çok Biçimlilik (Polymorphism)

- Bu tablo çalışma zamanında güncellenmektedir.

Çok Biçimlilik (Polymorphism)



Çok Biçimlilik (Polymorphism)



Çok Biçimlilik (Polymorphism)

Soyut (Abstract) Sınıflar

- İçersinde virtual olarak tanımlanmış fonksiyonların tanımının olmadığı sınıflardır.
- Kendi başlarına bu sınıflardan obje yaratılıp kullanılamaz.
- Tanımlamak için fonksiyonun yanına = 0 yazılmalıdır.
- Bu yaklaşıma pure virtual denir.

Çok Biçimlilik (Polymorphism)

```
#include <iostream>
using namespace std;

class sekiller {
protected:
int en, boy;
public:
sekiller(int a = 0, int b = 0) {
en = a;
boy = b;
}
virtual int alan() = 0;
};

class dikdortgen : public sekiller {
public:
dikdortgen(int a = 0, int b = 0) :sekiller(a, b) { }

int alan() {
cout << "Dikdortgen alani donduruluyor" << endl;
return (en * boy);
}
};
```

```
class ucken : public sekiller {
public:
ucken(int a = 0, int b = 0) :sekiller(a, b) { }

int alan() {
cout << "Ucken alani donduruluyor." << endl;
return (en * boy / 2);
}
};

int main() {

sekiller* obj1;
dikdortgen obj2(10, 7);
ucken  obj3(10, 5);

obj1 = &obj2;
obj1->alan();

obj1 = &obj3;
obj1->alan();

return 0;
}
```

Çıktı:

```
Dikdortgen alani donduruluyor
Ucken alani donduruluyor.
```