190600213201 5
Programlama Dilleri Temelleri

BAİBÜ Bilgisayar Müh.

Ders 8

Dr. Araş. Gör. İsmail Hakkı Parlak

Aşırı Yükleme (Overloading) ve Geçersiz Kılma (Overriding)

- Aşırı Yükleme (Overloading): Bir metodun (method) veya bir operatörün, metoda iletilen parametrelere veya operatörün işlem yaptığı işlenenlere bağlı olarak farklı şekillerde davranma yeteneğini ifade eder.
- Bir metodun aşırı yüklenmesi yeniden kullanılabilirliği artırır. Örneğin, çok az farklılık gösteren birden çok yöntem yazmak yerine, bir yöntem yazıp onu aşırı yükleyebiliriz.

Method Overloading: Java

```java
public class Sum {

 public int sum(int x, int y) {
  return (x + y);
 }

 public int sum(int x, int y, int z) {
  return (x + y + z);
 }

 public double sum(double x, double y) {
  return (x + y);
 }

 public static void main(String args[]) {
  Sum s = new Sum();
  System.out.println(s.sum(10, 20));
  System.out.println(s.sum(10, 20, 30));
  System.out.println(s.sum(10.5, 20.5));
 }
}
```

Metot aşırı yükleme, farklı metotların aynı ada ve farklı imzalara (signature) sahip olmasına izin verir.

Metot aşırı yükleme, Java'da Derleme Zamanı Polimorfizmi (Compile-time Polymorphism), Statik Polimorfizm veya erken bağlama (Early binding) olarak da bilinir.

Method Overloading: Python

```
def carp(a, b):
    return a * b

def carp(a, b, c):
    return a * b * c

print(carp(4, 5, 6))
print(carp(3, 4))
```

Python, varsayılan (default) olarak metot aşırı yüklemesini desteklemez.

Python'da yöntem aşırı yüklemesini gerçekleştirmenin farklı yolları vardır.

Method Overloading: Python

```
def carp(a, b, *args):
  carpim = a * b
  for arg in args:
    carpim *= arg
  return carpim


print(carp(4, 5, 6))
print(carp(3, 2))
```

Python, varsayılan (default) olarak metot aşırı yüklemesini desteklemez.

Python'da yöntem aşırı yüklemesini gerçekleştirmenin farklı yolları vardır.

Operator Overloading: Python

Operator	Method
+	__add__(self, other)
-	__sub__(self, other)
*	__mul__(self, other)
/	__truediv__(self, other)
%	__mod__(self, other)
<	__lt__(self, other)
<=	__le__(self, other)
==	__eq__(self, other)
!=	__ne__(self, other)
>	__gt__(self, other)
>=	__ge__(self, other)

(Ref: https://stackabuse.com/overloading-functions-and-operators-in-python/)

Operator Overloading: Python

```
a = 3
b = 4
c = a.__add__(b)
print(a, c)
```

```
class Sayi:
    def __init__(self, n):
      self.n = n

    def __add__(self, other):
      return self.n + other.n


a, b = Sayi(4), Sayi(5)
print(a + b)
```

Geçersiz Kılma (Overriding)

- Nesne yönelimli programlamada "metot geçersiz kılma" (ezme / method overriding), bir alt sınıfın üst sınıflarından biri tarafından sağlanmış olan bir metodun belirli bir implementasyonunu sağlayabilmesidir.
- Alt sınıftaki override edilen metot, üst sınıfındakiyle aynı ada ve parametrelere sahip olmalıdır.
- Method overriding, çalışma zamanı polimorfizmi (runtime polymorphism) için kullanılır.

Overriding: Java

```java
class Parent {
    void show() {
        System.out.println("Parent");
    }
}

class Child extends Parent {
    @Override
    void show() {
        System.out.println("Child");
    }
}

class Main {
    public static void main(String[] args) {
        Parent obj1 = new Parent();
        obj1.show();

        // Runtime polymorphism
        Parent obj2 = new Child();
        obj2.show();
    }
}
```

Method Overriding: Python

```
class Parent1:
    def show(self):
        print("Parent 1")


class Parent2:
    def display(self):
        print("Parent 2")


class Child(Parent1, Parent2):
    def show(self):
        print("Child")


obj = Child()
obj.show()
obj.display()
```

Geçersiz Kılma vs. Aşırı Yükleme

Overriding (Geçersiz Kılma)	**Overloading (Aşırı Yükleme)**
Çalışma zamanı (runtime) polimofizmi.	Derleme zamanı (compile time) polimorfizmi.
Metot çağrısı, nesne türüne göre çalışma zamanında belirlenir.	Metot çağrısı derleme zamanında belirlenir.
Üst sınıf ve alt sınıf arasında gerçekleşir.	Aynı sınıfa ait metotlar arasında gerçeklerşir.
Aynı imzaya sahip (isim ve parametreler)	Aynı isme (identifier) sahip ancak parametreler farklıdır.
Hatalar çalışma zamanında gözlemlenebilir.	Hatalar derleme zamanında yakalanabilir.

https://www.digitalocean.com/community/tutorials/overriding-vs-overloading-in-java