Overriding

Ayni isimde farkli iki method olusturmanin iki yolu vardir

1- Method Signature'ini degistirerek ayni isimde farkli iki method yapmak

Overloading

2- Method Signature'ini degistirmeden iki method'dan sadece birinin calismasini saglamak

Overriding

NOT 1: Method Signature'ini degistirmezsek Java her iki method'u ayni method olarak gorur ve bir class icerisinde ayni method'u iki kez olusturmaya IZIN VERMEZ.
Biz parent ve child class'da signature'I ayni olan iki method olusturursak Java ikisinden sadece birini calistirir

NOT : Her iki yonteme dikkat edilirse Method Body'nin degismesi sart degildir.

Ancak 2.yontemde signature zaten degismedigi icin, Body degismezse 2.method farkli bir method olmaz.

Overriding

Method Signature

Method signature, method ismi ve parametrelerden olusur.

Signature'i degistirmek icin bilesenlerinden isim veya parametrelerle ilgili degisiklikler yapilmalidir.

1) Isim ayni kalsa da parametre sayisi degistirildiginde signature degisir

```
public void toplama(int a, int b) {
    System.out.println(a+b);
}
```

```
public void toplama(int a, int b,int c) {
    System.out.println(a+b+c);
}
```

```
public void toplama(int a, int b,int c,int d) {
    System.out.println(a+b+c+d);
}
```

2) Farkli data turlerine sahip parametrelerin yerleri degistirildiginde method signature degisir.

```
public void toplama(int a, String b, boolean c) {
    System.out.println("Merhaba");
}
```

```
public void toplama(int a, boolean c,String b) {
    System.out.println("Hosgeldin");
}
```

```
public void toplama(String b, int a, boolean c) {
    System.out.println("Hoscakal");
}
```

Overriding

Method Overriding nedir ?

Parent class'da varolan bir methodu method signature'ini degistirmeden, method body'sini degistirerek kullanmaya Method Overriding denir.

Method Overriding nicin kullanilir ?

Overriding kullanarak, child class'in parent class'daki methodu kendine uyarlayarak (implement) kullanmasini saglamis oluruz.

Overriding yapildiginda parent class'daki methoda Overridden Method,
child class'daki methoda Overriding Method denir.

Eclipse menu'den Source sekmesinde bulunan Override/Implement methods secenegiyle otomatik olarak overriding method'u olusturabiliriz. Bu sekilde yapilan islemde Java @Override annotation'i kullanir.

@Override kullanmak zorunda degiliz, istersek silebiliriz. Ancak kodun anlasilabilir ve okunabilir olmasi icin degil, overridden method'da degisiklik yapildiginda Java'nin rapor etmesi icin kullanilmasi tercih edilir.

Overriding

```java
public class Isci extends Personel{

    public static void main(String[] args) {

        Isci isci=new Isci();
        isci.isim="Mehmet";
        isci.soyisim="Bulut";
        isci.statu="isci";
        System.out.println(isci.isim + " "+isci.soyisim+" "+
        isci.statu+ " "+ isci.maasHesapla());

    }
    public int maasHesapla() {

        return (30*8*15) ;
    }
    public void calismaSaati() {
        System.out.println("Isciler gunluk 8 saat calisir");
    }
```

Overridden Methods

```java
public class UstaBasi extends Isci {

    public static void main(String[] args) {

        UstaBasi ub1 = new UstaBasi();
        ub1.isim = "Seher";
        ub1.soyisim = "Boss";
        ub1.statu = "Usta Basi";
        System.out.println(ub1.isim + " "+ub1.soyisim+" "+
        ub1.statu+ " "+ ub1.maasHesapla());
        ub1.calismaSaati();
    }
    public int maasHesapla() {

        return (30 * 8 * 20);
    }
    public void calismaSaati() {
        System.out.println("Ustabasi is bitene kadar calisir");
    }
```

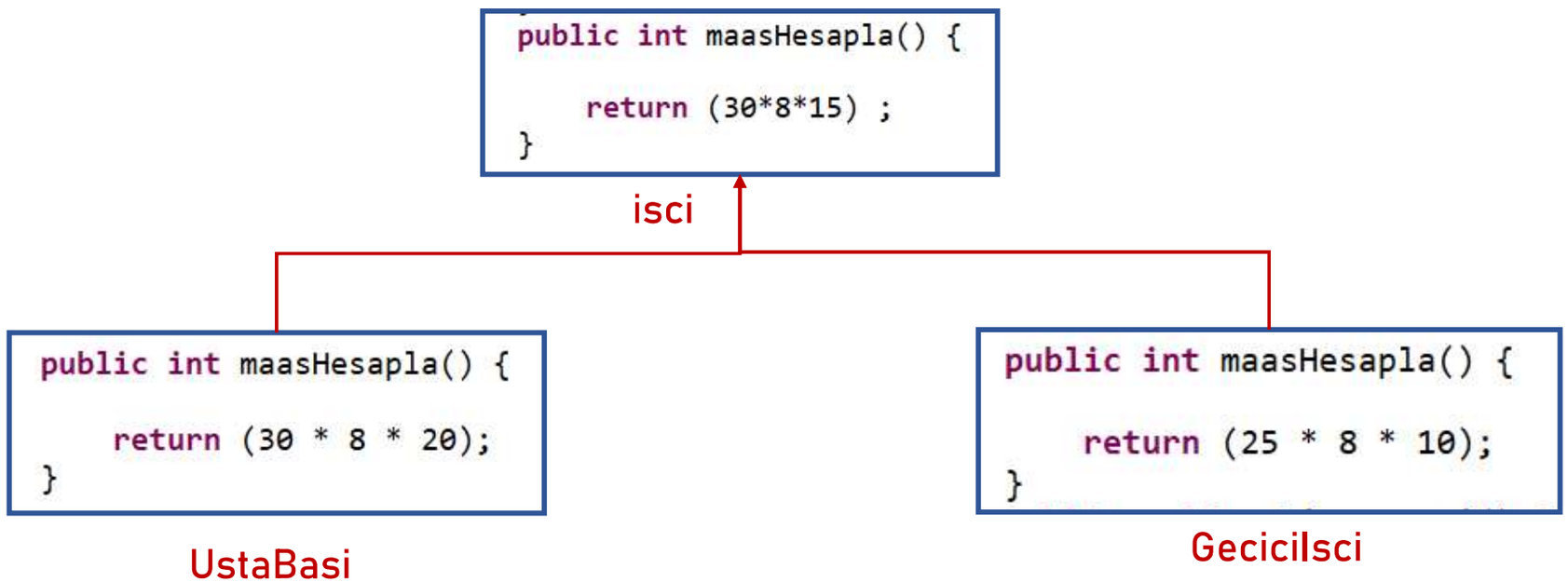
```java
public class GeciciIsci extends Isci {

    public static void main(String[] args) {

        GeciciIsci gi1 = new GeciciIsci();
        gi1.isim = "Faruk";
        gi1.soyisim = "Yanik";
        gi1.statu = "GeciciIsci";
        System.out.println(gi1.isim + " "+gi1.soyisim+" "+
        gi1.statu+ " "+ gi1.maasHesapla());
        gi1.calismaSaati();

    }
    public int maasHesapla() {

        return (25 * 8 * 10);
    }
    public void calismaSaati() {
        System.out.println("Gecici isciler haftalik 25 saat calisir");
    }
}
```

Overriding Methods

Method Overiding'i Nicin Kullaniriz ?

Overriding parent class'daki genel method'u degistirmeden child class'in kendine uygun method uretmesini saglar

Ornegimizde isci maasi hesaplanirken genel bir formul varken, child class olan Ustabasi ve Gecicilsci Class'lari kendilerine uygun maas hesaplama method'larina sahiptirler.



Overriding Kurallari

1) Method Signature'i *(isim ve parametreler)* ayni olmalidir.

2) Child class'daki method'un (overriding method) Access Modifier'i parent class'daki method'un (overridden) modifier'indan daha dar olamaz.

3) Overriding method covariant return type kullanmalidir.

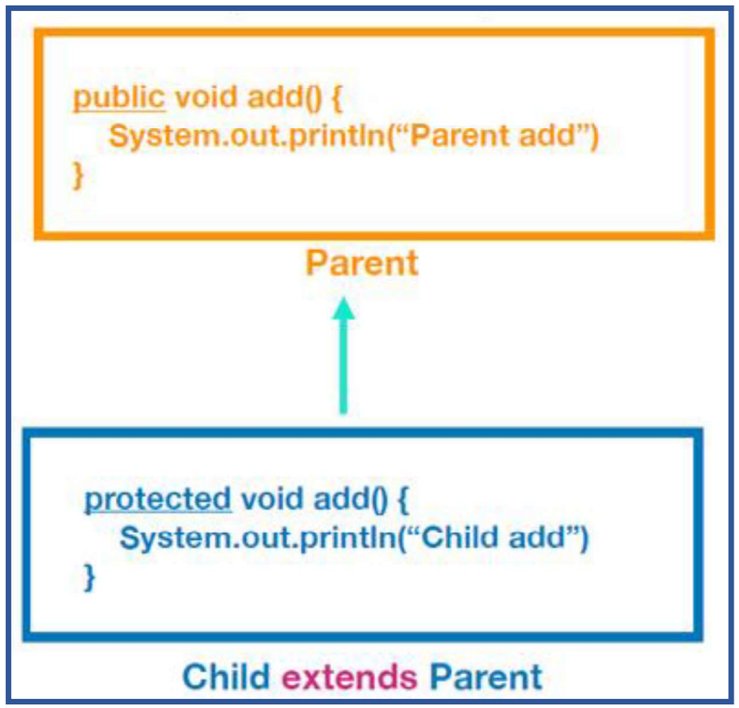
4) private, static and final method'lar overriding yapilamazlar

5 ve 6 sonra aciklanacak

5) Child class'daki method (overriding method), parent class'daki method'un (overridden method) throw edip etmedigine bakmaksizin compile time exception throw edebilir. Ancak parent class'da throw edilen exeption'dan daha genis olamaz

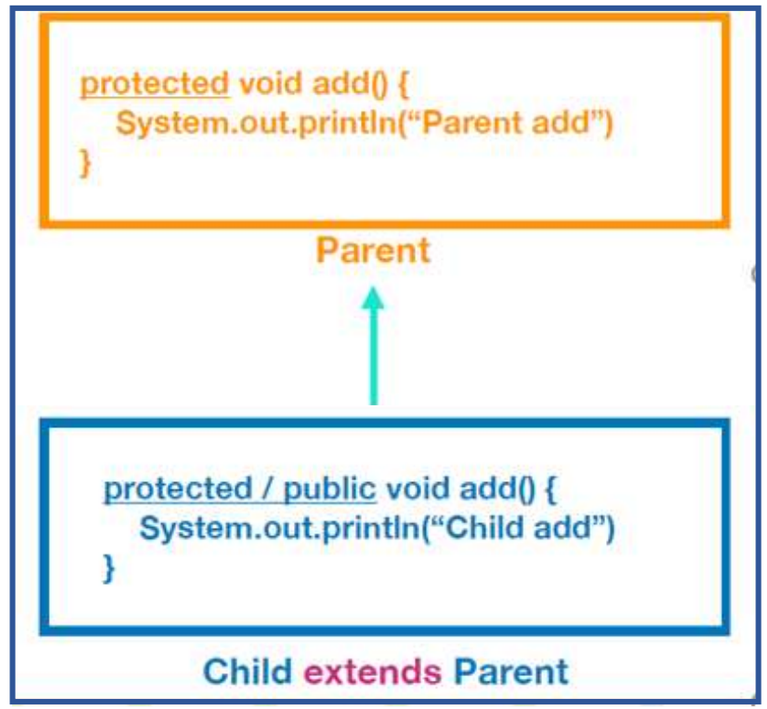
6) Eger abstract olmayan bir class abstract class'a extend ediyorsa veya bir interface'i implement ediyorsa abstract method'larin tamami override edilmelidir

Overriding Kurallari ? Return Type



Yanlis

Child Parent'i sinirlayamaz



Dogru

Child'in access modifier'i Parent ile ayni veya daha genis olmalidir

NOT : Private method'lar override edilemez..

Overriding Kurallari

Overriden ve overriding method'larin ikisini de kullanmak istersek child class'da (overriding method) super keyword'unu kullanabiliriz.

```
public class Lamb extends Animal {

    public void eat(){
        super.eat();
        System.out.println("Lambs eat grass");
    }

    public static void main(String[] args) {

        Lamb lamb = new Lamb();
        lamb.eat();

    }
}
```

Polymorphism

Polymorphism = Overloading + Overriding

- Poly “çok” morph ise “form”, “biçim” anlamlarını taşır. Bu ikisinin birleşimiyle oluşan “polymorphism” sözcüğü “çok biçimlilik” anlamına gelir.

- Özetle, oluşturulan nesnelerin gerektiğinde kılıktan kılığa girip başka bir nesneymiş gibi davranabilmesine polymorphism diyebiliriz. Bunlar program kodlarının yeniden kullanılabilmesi veya var olan kodun geliştirilebilmesi açısından çok önemlidir.

Polymorphism Turleri

- Method Overloading bir compile time (static) polymorphism'dir. Method Overloading sayesinde ayni isme, ayni body'e, farkli parametrelere sahip bir cok method uretip kullanabiliriz.

- Method Overriding bir run time (dynamic) polymorphism'dir. Method Overriding sayesinde ayni isme, ayni parametrelere'e, farkli body'e sahip bir cok method uretip kullanabiliriz.

Overloading vs Overriding

1) Overloading'de method signature degisir, Overriding'de degismez.

2) Overloading'de body istenirse degistirilebilir, Overriding'de body %99 degistirilir.

3) final, static ve private methodlar Overload edilebilir, ama Override edilemezler.

4) Overloading Compile Time Polymorphism (static)'dir, Overriding is Run Time Polymorphism'(dynamic)dir.

5) Overloading'de inheritance gerekmez, Overriding'de gerekir.

6) Overloading'de istedigimiz sekilde access modifier ve return type kullanabiliriz ama Overriding'de access modifier ve return type kullanma belli kurallara baglidir.

Polymorphism

1) “method signature” nedir? Hangi method’lar Java’ya gore aynidir ?

Method signature “method ismi” ve “parameter listesi”nden olusur.

Signature’I ayni olan method’lar Java’ya gore ayni method’dur.

2) Polymorphism nedir?

Polymorphism “overloading” ve “overriding”in birlesimidir.

3) “Overloading” ve “Overriding”in farki nedir ?

Overloading’de sadece parametreler degisir, overriding’de signature’a dokunulmaz sadece body degisir.

4) “Overriding”in faydasi nedir?

Coklu uygulama, reusability

Polymorphism

1) Asagidaki programdaki CTE nasil giderilebilir ? Program duzeltilip calistirildiginda konsolda ne yazdirir?

```
3  class ParentClass {
4      public void getDetails(String temp) {
5          System.out.println("Derived class " + temp);
6      }
7  }
8
9  public class Test01 extends ParentClass {
10
11     public int getDetails(String temp) {
12         System.out.println("Test class " + temp);
13         return 0;
14     }
15
16     public static void main(String[] args) {
17         Test01 obj = new Test01();
18         obj.getDetails("GFG");
19     }
20 }
```

a) Derived class GFG

b) Test class GFG

c) Compilation error

d) Runtime error

Polymorphism

2) Asagidaki programdaki CTE nasil giderilebilir ? Program duzeltilip calistirildiginda konsolda ne yazdirir?

```
3  class Derived {
4      public void getDetails() {
5          System.out.println("Derived class");
6      }
7  }
8
9  public class Test02 extends Derived {
10     protected void getDetails() {
11         System.out.println("Test class");
12     }
13
14     public static void main(String[] args) {
15         Derived obj = new Test02();
16         obj.getDetails();
17     }
18 }
```

a) Test class

b) Compilation error due to line xyz

c) Derived class

d) Compilation error due to access modifier

Polymorphism

2) Asagidaki programdaki CTE nasil giderilebilir ? Program duzeltilip calistirildiginda konsolda ne yazdirir?

Cevap :

```
3  class Derived {
4      public void getDetails() {
5          System.out.println("Derived class");
6      }
7  }
8
9  public class Test02 extends Derived {
10     public void getDetails() {
11         System.out.println("Test class");
12     }
13
14     public static void main(String[] args) {
15         Derived obj = new Test02();
16         obj.getDetails();
17     }
18 }
```

a) Test class

b) Compilation error due to line xyz

c) Derived class

d) Compilation error due to access modifier

Polymorphism

3) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
class Derived03 {
    public void getDetails() {
        System.out.printf("Derived class ");
    }
}

public class Test03 extends Derived03 {
    public void getDetails() {
        System.out.printf("Test class ");
        super.getDetails();
    }

    public static void main(String[] args) {
        Derived03 obj = new Test03();
        obj.getDetails();
    }
}
```

a) Test class Derived class
b) Derived class Test class
c) Compilation error
d) Runtime error

Polymorphism

4) Asagidaki programdaki CTE nasil giderilebilir ? Program duzeltilip calistirildiginda konsolda ne yazdirir?

```
3  class Derived04 {
4      protected final void getDetails() {
5          System.out.println("Derived class");
6      }
7  }
8
9  public class Test04 extends Derived04 {
10     protected final void getDetails() {
11         System.out.println("Test Class");
12     }
13
14     public static void main(String[] args) {
15         Derived04 obj = new Derived04();
16         obj.getDetails();
17     }
18 }
```

a) Derived class
b) Test class
c) Runtime error
d) Compilation error

Polymorphism

4) Asagidaki programdaki CTE nasil giderilebilir ? Program duzeltilip calistirildiginda konsolda ne yazdirir?

Cevap :

```
3  class Derived04 {
4      protected  void getDetails() {
5          System.out.println("Derived class");
6      }
7  }
8
9  public class Test04 extends Derived04 {
10     protected final void getDetails() {
11         System.out.println("Test Class");
12     }
13
14     public static void main(String[] args) {
15         Derived04 obj = new Derived04();
16         obj.getDetails();
17     }
18 }
```

a) Derived class
b) Test class
c) Runtime error
d) Compilation error

Polymorphism

5) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
class Person {
    public void talk() {
        System.out.println("First Program");
    }
}

class Student extends Person {
    public void talk() {
        System.out.println("Second Program");
    }
}

public class Test05 {

    public static void main(String[] args) {
        Person p = new Student();
        p.talk();
    }
}
```

Polymorphism

6) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
public class Test06 {
    public static void main(String[] args) {
        new C().create();
        new D().update();
        new R().read();
        new D().delete();
    }}
class C {
    public void create() { System.out.print("c");
    }}
class U {
    private void update() { System.out.print("u");
    }}
class R extends C {
    public void create() { System.out.print("C");
    }
    protected void read() { System.out.println("R");
    }}
class D extends U {
    void update() { System.out.println("U");
    }
    void delete() { System.out.println("D");
    }}
```

Polymorphism

7) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
class Super {
    public Integer getLength() {
        return new Integer(4);
    }
}

public class Test07 extends Super {
    public Integer getLength() {
        return (5);
    }

    public static void main(String[] args) {
        Super sooper = new Super();
        Test07 sub = new Test07();
        System.out.println(sooper.getLength().toString() + ", " + sub.getLength().toString());
    }
}
```

Polymorphism

8) Asagidaki program calistirildiginda konsolda ne yazdirir?

```
public class Test08 {

    public static void main(String[] args) {
        X x = new X();
        Y y = new Y();
        y.m2();
        x.m1();
        y.m1();
        x=y;
        x.m1();
    }}

    class X{
        public void m1() {
            System.out.println("m1, X class");
        }
    }
    class Y extends X{
        public void m1() {
            System.out.println("m1, Y class");
        }
        public void m2() {
            System.out.println("m2, Y class");
        }}
```

Polymorphism

9) Asagidaki program calistirildiginda konsolda ne yazdirir?

What will be the output of the following program?

```
public class Outer {
    public static void main(String args[]) {
        Computer mouse = new Laptop();
        System.out.println(mouse.getValue(100, 200));
    }
}
class NoteBook {
    int getValue(int a, int b) {
        if (a > b)
            return a;
        else
            return b;
    }
}
class Computer extends NoteBook {
    int getValue(int a, int b) {
        return a * b;
    }
}
class Laptop extends Computer {
    int getValue(int a, int b) {
        return b - a;
    }
}
```

Polymorphism

10) Asagidaki program calistirildiginda konsolda ne yazdirir?

What will be the output of the following program?

```
public class Product {
    public static void main(String[] args) {
        M m = new M();     M n = new N();
        M o = new O();     O oo = new O();
        m.product(3);     n.product(3);
        oo.product(3);
    }
}
class M {
    int product(int i) {
        int result = i * i;
        System.out.print("{" + i + ", " + result + "}~");
        return result;
    }
}
class N extends M {
    int product(int i) {
        int result = i + i;
        System.out.print("[" + i + ", " + result + "]~");
        return result;
    }
}
class O extends M {
    int product(int i) {
        int result = i * 2;
        System.out.print("(" + i + ", " + result + ")~");
        return result;
    }
}
```

Inheritance'da Data Type Kullanimi

Output nedir?

```
class Person {
    public Person() {
        System.out.println("Person Constructor");
    }

    public void talk() {
        System.out.println("First Program");
    }
}

class Student extends Person {
    public void talk() {
        System.out.println("Second Program");
    }
}

public class Test04 {

    public static void main(String[] args) {
        Person p = new Student();
        p.talk();
    }
}
```

```
Person Constructor
Second Program
```