



Principet e Avancuara të Gjuheve të Programimit

Lamir Shkurti, Can. PhD.
lamir.shkurti@gmail.com

SHKENCA KOMPJUTERIKE

- Trashëgimia është shtyllë e rëndësishme e OOP (Programimi i Orientuar në Objekte. Është mekanizmi në java me të cilin një klasë lejohet ta trashëgojë tiparet (fushat dhe metodot) e një klase tjetër.
- **Terminologji të rëndësishme:**
- **Super Klasa:** Klasa tiparet e së cilës trashëgohen njihet si super klasë (ose një klasë bazë apo një klasë prindërore).
- **Nënklasa:** Klasa që trashëgon klasën tjetër njihet si nën klasë (ose një klasë e prejardhur, klasë e zgjatur ose klasë fëmijë). Nënklasa mund të shtojë fushat dhe metodat e veta përveç fushave dhe metodove të **super** klasës.
- **Ripërdorimi:** Trashëgimia mbështet konceptin e "ripërdorimit", d.m.th. kur duam të krijojmë një klasë të re dhe tashmë ekziston një klasë që përfshin disa nga kodet që duam, ne mund të krijojmë klasën tonë të re nga klasa ekzistuese. Duke bërë këtë, ne jemi duke ripërdorur fushat dhe metodat e klasës ekzistuese.

QËLLIMI I TRASHËGIMISË



- Ideja prapa trashëgimisë në Java është që të kemi mundësi të krijojmë klasa të reja të ndërtuara mbi klasat ekzistuese.
- Kur trashëgojmë nga një klasë ekzistuese, mund të ripërdorim metodat dhe fushat e klasës prind. Për më tepër, ne gjithashtu mund të shtojmë metoda dhe fusha të reja në klasën e re.

SINTAKSA E JAVA TRASHËGIMISË

Sintaksa

```
class Subclass-name extends Superclass-name
{
    //methods and fields
}
```

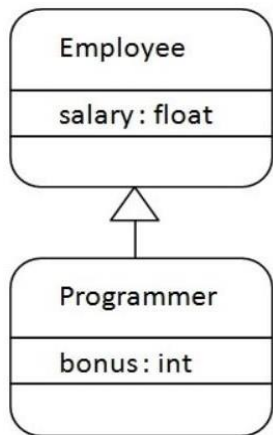
Fjala `extends` tregon që jemi duke krijuar një klasë të re që buron nga një klasë ekzistuese.

Kuptimi i fjalës "extends" është: të rrisë funksionalitetin.

Në terminologjinë Java, një klasë e cila trashëgohet quhet prind ose superklasë, dhe klasa e re quhet fëmijë ose nënklasë (Subclass).

SHEMBULL

Siç tregohet në figurë, Programeri është nënklasa dhe Punonjësi është superklasa. Marrëdhënia midis dy klasave është: Programeri IS-A Punonjës. Kjo do të thotë që Programeri është një lloj i Punonjësve.



```

public class Employee{
    float salary=4000.0f;
}
  
```

```

public class Programmer extends Employee{
    int bonus=1000;
    public static void main(String args[]){
        Programmer p=new Programmer();
        System.out.println("Programmer salary is:" + p.salary);
        System.out.println("Bonus of Programmer is:"+p.bonus);
    }
}
  
```

Në shembullin e mësipërm, objekti Programmer mund të ketë qasje në fushat e klasës vetanake, si dhe të klasës Employee, d.m.th: Ripërdorimi i kodit.

Output:

```

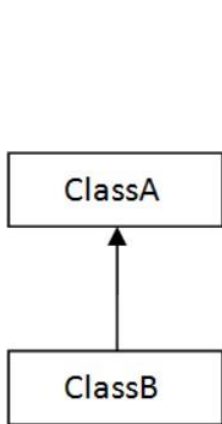
Programmer salary is: 4000.0
Bonus of programmer is: 1000
  
```

LLOJET E TRASHËGIMISË

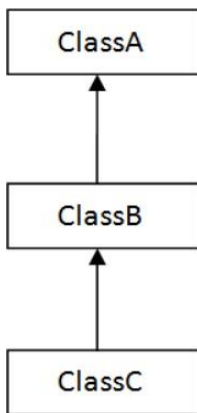
Në bazë të klasës, mund të ekzistojnë tri lloje të trashëgimisë në java: të vetme (single), shumë nivele (multilevel) dhe hierarkike (hierarchical).

Në java, trashëgimia e shumëfishtë dhe hibride mbështetet vetëm përmes ndërfaqes (interface).

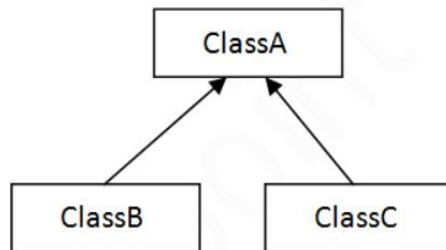
[Do të mësojmë më vonë për ndërfaqet \(interface\).](#)



1) Single



2) Multilevel



3) Hierarchical

Shënim: Trashëgimia e shumëfishtë nuk mbështetet nga Java (përmes klasës).

TRASHËGIMIA E SHUMËFISHTË

Nëse një klasë mund të trashëgojë veti të më shumë se një klase prind. Problemi ndodh kur ekzistojnë metoda me emërtim të njëjtë.

Me thirrjen e metodës, compiler nuk mund të përcaktojë se cila metodë e klasës do të thirret.

```
// First Parent class
```

```
class Parent1{  
    void fun(){  
        System.out.println("Parent1");  
    }  
}
```

```
// Second Parent Class
```

```
class Parent2 {  
    void fun () {  
        System.out.println("Parent2");  
    }  
}
```

```
// Error : Test is inheriting from multiple  
// classes
```

```
class Test extends Parent1, Parent2  
{  
    public static void main(String [] args)  
    {  
        Test t = new Test();  
        t.fun();  
    }  
}
```

Output : Compile Error

PROTECTED (MODIFIER)



- Dukshmëria (Visibility) e variablave dhe metodave tregon se cilat variabla apo metoda mund të trashëgohen e cilat jo.
- Variablat dhe metodat me dukshmëri të deklaruar si **public** mund të trashëgohen, ndërsa ato me dukshmëri **private** nuk mund të trashëgohen.
- Variablat e deklaruar si **public** nuk i përmbahen kërkesës për "mbërthim" (encapsulation)
- Ekziston edhe forma e tretë e dukshmërisë që ndihmon gjatë "inheritance" e njohur si : **protected**

PROTECTED



- **protected** mundëson që variabla apo metoda të trashëgohet nga superklasa në nënklasë.
- Dukshmëria e **protected** i përmbahet më tepër kërkesës për "mbërthim" se sa që ofron dukshmëria **public**
- Mirëpo "mbërthimi" në **protected** nuk ofrohet ashtu siç ofrohet përmes dukshmërisë **private**
- Metodatat dhe variablat **protected** në UML Diagram paraqiten duke përdorur karakterin apo simbolin #.

REFERENCA SUPER



- Konstruktorët edhe pse mund të kenë dukshmëri **public**, nuk mund të trashëgohen. Neve shpesh na nevojitet t'i qasemi konstruktorit të superklasës në mënyrë që t'i inicilizojmë variablat e superklasës
- Referenca **super** është fjalë e rezervuar, përdoret t'i referohemi superklasës dhe shpesh përdoret që të thirret konstruktori i superklasës

REFERENCA SUPER



- Konstruktori i nënklasës është pjesa ku thirret konstruktori i superklasës.
- Brenda konstruktorit të nënklasës, nëse veç thirret konstruktori i superklasës, atëherë kjo duhet të jetë vija e parë e kodit, pra vija e parë e kodit në këtë rast i qasemi përmes referencës **super**.
- Referenca **super**, mund të përdoret që të thirren variablat dhe metodat e superklasës nëse variablat e superklasës fshihen apo metodat mbishkruhen.

SHEMBULL



```
public class Personi{  
    String emri;  
    String mbiemri;  
    int moshë;  
  
    public Personi(String emri, String mbiemri, int moshë){  
        this.emri=emri;  
        this.mbiemri=mbiemri;  
        this.moshë=moshë;  
    }  
}
```

SHEMBULL

```
public class Studenti extends Personi{
    String drejtimi;
    public Studenti(String emri,String mbiemri,int mosha,String drejtimi){
        super(emri,mbiemri,mosha);
        this.drejtimi=drejtimi;
    }
    public String toString(){
        return emri+" "+mbiemri+" "+mosha+" Drejtimi:"+drejtimi;
    }
    public static void main(String [] args){
        Studenti s1=new Studenti("Filane","Fisteku1",19,"SHKI");
        Studenti s2=new Studenti("Filan","Fisteku1",20,"MEK");
        System.out.println(s1);
        System.out.println(s2);
    }
}
```

