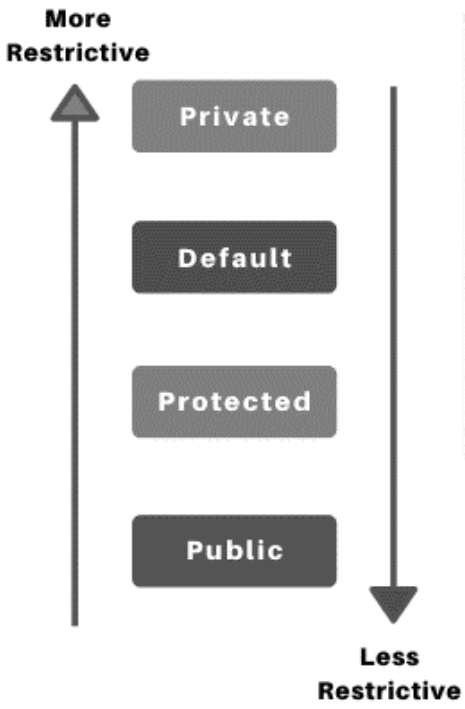
Shkenca Kompjuterike 2

Lavdim Menxhiqi - lavdim.menxhiqi@ubt-uni.net

Access Modifiers



		public	private	protected	default
Same Package	Class	YES	YES	YES	YES
	Sub class	YES	NO	YES	YES
	Non sub class	YES	NO	YES	YES
Different Package	Sub class	YES	NO	YES	NO
	Non sub class	YES	NO	NO	NO

Polimorfizmi

- Polimorfizëm paraqitet kur kemi shumë klasa që janë të lidhura me njëra-tjetrën përmes trashëgimisë.
- Polimorfizmi në Java është një koncept me të cilin ne mund të kryejmë një veprim të vetëm në mënyra të ndryshme.
- Polimorfizmi rrjedh nga 2 fjalë greke: poli dhe morfe.
  - "poli" do të thotë shumë dhe
  - "morfat" do të thotë forma.
- Ekzistojnë dy lloje të polimorfizmit në Java: compile-time polimorfizmi dhe runtime polimorfizmi.
- Ne mund të realizojmë polimorfizëm në java me anë të method overloading dhe method overriding.

Polimorfizmi Statik (compile-time)

Method Overloading është shembulli i përpilimit të polimorfizmit kohor (compile-time).

Method Overloading na lejon të kemi më shumë se një metodë me të njëjtin emër, nëse parametrat e metodave janë të ndryshëm në numër, rend dhe/ose lloje të të dhënave.

Method overloading është një nga mënyrat se si java mbështet polimorfizmin statik (compile-time).

Në shembullin në vazhdim do të kemi dy deklarime të së njëjtës metodë add ( ).

Se cila metodë add( ) do të thirret caktohet varësisht nga numri i parametrave gjatë compile-time.

Kjo është arsyeja që kjo njihet gjithashtu si polimorfizëm compile-time.

Polimorfizmi Statik (compile-time)

```
class SimpleCalculator
{
    int add(int a, int b)
    {
         return a+b;
    }
    int  add(int a, int b, int c)
    {
         return a+b+c;
    }
}
```

```
public class Demo
{
   public static void main(String args[])
   {
            SimpleCalculator obj = new SimpleCalculator();
        System.out.println(obj.add(10, 20));
        System.out.println(obj.add(10, 20, 30));
   }
}
```

Polimorfizmi Dinamik (Runtime)

Polimorfizmi dinamik është një proces në të cilin një thirrje për një metodë override përcaktohet gjatë kohës së interpretimit, prandaj quhet runtime polimorfizëm.

Në shembull në vazhdim kemi dy klasa ABC dhe XYZ.

- ABC është një klasë prindërore dhe XYZ është një klasë fëmijë.

Klasa femijë e bën overide metodën myMethod ( ) të klasës prind.

Polimorfizmi Dinamik (Runtime)

```
class ABC{
   public void myMethod(){
        System.out.println("Overridden Method");
   }
}
public class XYZ extends ABC{

   public void myMethod(){
        System.out.println("Overriding Method");
   }
   public static void main(String args[]){
        ABC obj = new XYZ();
        obj.myMethod();
   }
}
```

Output:

```
Overriding Method
```

Klasat Abstrakte

- Abstraksioni i të dhënave është procesi i fshehjes së detajeve të caktuara dhe shfaqja e informacionive thelbësore.
- Abstraksioni mund të arrihet përmes klasave Abstrakte ose përmes Interface-ave.
- Fjala kyçe "abstract" është një non-access modifiers, i përdorur për klasa dhe metoda:

  Klasa abstrakte është një klasë e kufizuar që nuk mund të përdoret për të krijuar objekte (nuk kemi akses në të, ajo duhet të trashëgohet nga një klasë tjetër).

  Metoda abstrakte mund të përdoret vetëm në një klasë abstrakte dhe nuk ka trup (body). Body (implementimi) sigurohet nga nënklasa.

Klasat Abstrakte

Një klasë abstrakte mund të ketë metoda abstrakte dhe të rregullta.

```
abstract class Animal {
  public abstract void animalSound();
  public void sleep() {
    System.out.println("Zzz");
  }
}
```

Nga shembulli i mësipërm, nuk është e mundur të krijojmë një objekt të klasës Animal.

```
Animal myObj = new Animal();
```

Klasat Abstrakte

Për të hyrë në klasën abstrakte, ajo duhet të trashëgohet nga një klasë tjetër.

```
// Abstract class
abstract class Animal {
  // Abstract method (nuk ka body)
  public abstract void animalSound();
  // Metode e rregullt
  public void sleep() {
    System.out.println("Zzz");
  }
}

class Wolf extends Animal {
  public void animalSound() {
    System.out.println("The wolf says: auu auu");
  }
}
```

```
class KlasaKryesore {
  public static void main(String[] args) {
    Wolf wolf = new Wolf();
    wolf.animalSound();
    wolf.sleep();
  }
}
```

Pse na duhet një klasë abstrakte?

Le të themi se kemi një klasë Animal që ka një metode sound ( ) dhe nënklasat (trashëgiminë) e saj si Qeni, Luani, Kali, Macja etj.

Meqenëse tingulli i kafshës ndryshon nga një kafshë në tjetrën, nuk ka asnjë kuptim për ta implementuar këtë metodë në klasën prindërore.

Çdo nënklasë duhet të mbishkruaj (override) këtë metodë për ti dhënë detajet e veta, pasi që klasa Luani do të ketë tjetër ulërimë në këtë metodë dhe klasa e Qenit do të ketë tjetër ulërimë.

Kështu, deklarimi si abstrakte e kësaj metode do të ishte zgjedhja më e mirë pasi duke e bërë abstrakte këtë metodë, ne i detyrojmë të gjitha nënklasat të implementojnë këtë metodë, gjithashtu nuk duhet të japim ndonjë implementim të kësaj metode në klasën prind .

Klasat Abstrakte - Pikat për tu mbajtur mend

- Një klasë abstrakte duhet të deklarohet me fjalën "abstract".
- Mund të ketë metoda abstrakte dhe jo-abstrakte.
- Nuk mund të krijohet instanca e saj.
- Mund të ketë konstruktorë dhe metoda statike.
- Mund të ketë metoda final të cilat do të detyrojnë nënklasën të mos ndryshojë trupin e metodës.