Nasleđivanje

Nasleđivanje

Prvi, ujedno i najznačajniji princip OOP-a je **nasleđivanje**. Ovaj princip omogućava klasi da nasledi sva stanja i ponašanja neke druge klase.

Konkretno, podrazumevamo da uvedimo novi tip koji je proširen novim atributima i metodama u odnosu na neki već postojeći tip.

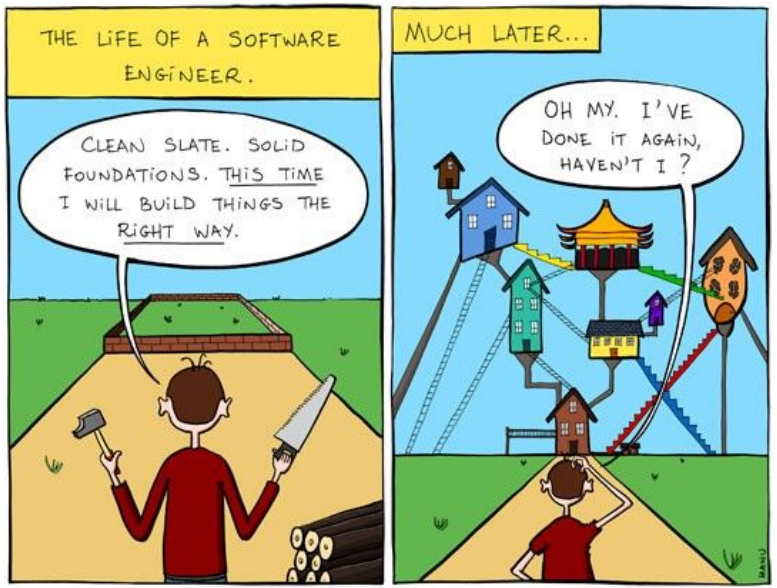
Primer: class **TerenskoVozilo**(**PrevoznoSredstvo**)

Prednosti nasleđivanja

Glavna prednost jeste izbegavanje pravljenja grešaka, kao i umanjenje količine koda.

Pretpostavimo da želimo da napravimo 3 klase životinja: **Pas**, **Macka** i **Mis**. Ukoliko znamo da sve životinje moraju da se kreću, da se hrane i da se oglašavaju, za svaku klasu bismo definisali identične metode i atribute.

U ovakvim situacijama, ideja je da sva svojstva koja su zajednička za ove tipove, definišemo u klasi koja će biti bazna klasa.

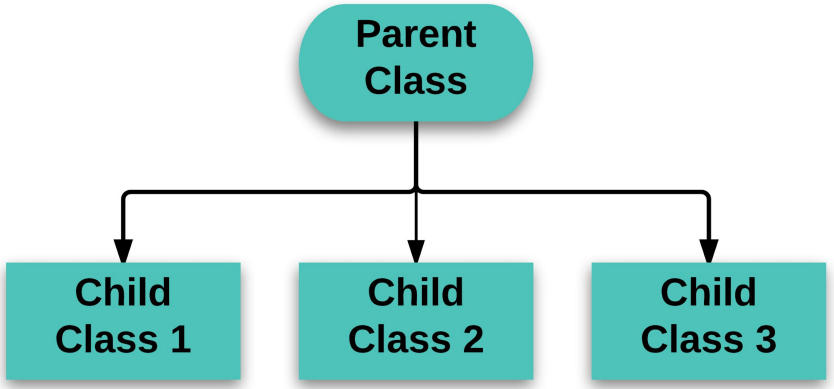


Roditelj i dete klase

Važno je naglasiti razliku između klase koja nasleđuje atribute i metode i klase od koje se nasleđuje.

Osnovna klasa koja se nalazi na vrhu hijerarhije, odnosno klasa iz koje se druge klase izvode naziva se **bazna klasa**, **natklasa** ili **roditeljska klasa**.

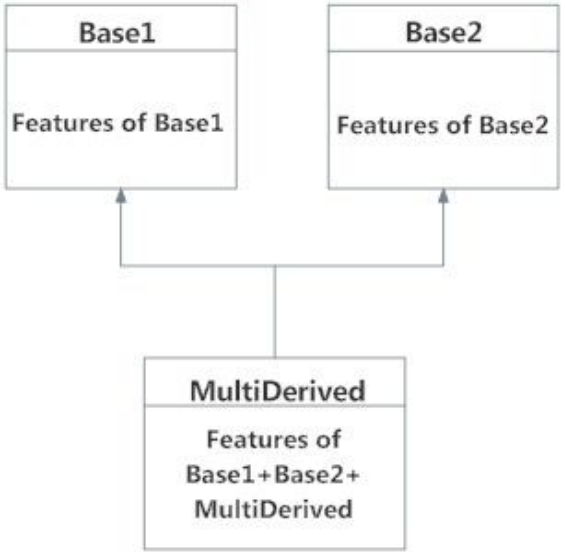
Klasa koja nastaje izvođenjem neke druge klase i proširivanjem novim svojstvima (atributima i metodama) naziva se **izvedena klasa**, **potklasa** ili **dete klasa**.



Višestruko nasleđivanje

Nekada se javi potreba da konkretan tip nasledi stanja i ponašanja od dva ili više tipova.

Uzmimo sledeći slučaj u obzir. Postoje 3 klase: **Covek**, **Sin** i **Vozac**. Klasa **Sin** je izvedena klasa klase **Covek**. Postavlja se pitanje, može li nečiji sin koji ima odgovarajući broj godina da postane vozač?



Da bismo obezbedili usvajanje svojstava od dva ili više drugih tipova koristimo višestruko nasleđivanje.

Primer: **class Sin**(**Covek**, **Vozac**)

Funkcija isinstance()

Funkcija **isinstance()** je predefinisana Python metoda koja služi za proveru tipa. Konkretno omogućava da proverimo da li je neka vrednost instanca odgovarajuće klase.

Ona prima dva argumenta gde je prvi argument sam objekat koji se pita, a drugi argument tip odnosno klasa.

Primer: **isinstance**(**zivotinja**, **Pas**)

Ova funkcija često se koristi u OOP paradigmi i u zavisnosti od jezika ima svoju definiciju.

Njena najčešća primena je kada hoćemo da izvršimo kontrolu toka na osnovu tipa neke vrednosti.

Primer: Samo u slučaju da je životinja pas, neka trči po lopticu.

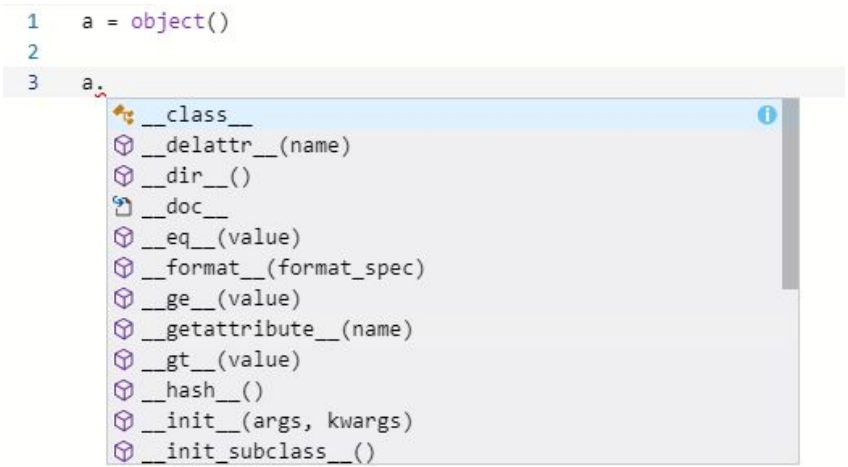
Object klasa

U svim OOP jezicima, na vrhu hijerarhije postoji klasa od koje se sve druge klase izvode. Klasa koja je natklasa svih klasa naziva se **object** klasa.

Kada definišemo klasu, iako ne navodimo da naša klasa nasleđuje object, to se dešava u pozadini. Ovakvo nasleđivanje nazivamo implicitno nasleđivanje.

Tip object ima obavezno neparametrizovan konstruktor.

Primer: **a = object()**



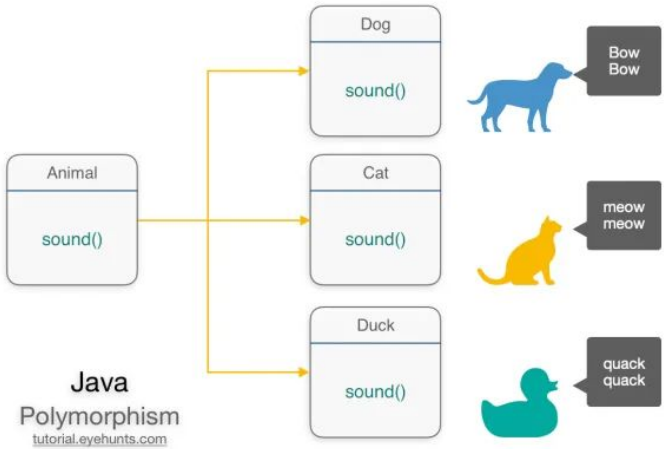
Polimorfizam

Polimorfizam

Nasleđivanje postavlja osnovu za naredni princip OOP-a, a to je polimorfizam. **Polimorfizam** (višeobličnost) je mogućnost da se jedan tipom označavaju drugi tipovi koji pripadaju istoj grani.

Uzmimo primer da imamo 3 klase: **Pas**, **Macka** i **Zivotinja**. Na prvu loptu jasno je da su tipovi **Pas** i **Macka** nasleđeni od tipa **Zivotinja**.

Pod pretpostavkom da možemo da sve objekte možemo da označimo nekim tipom, tip **Macka** možemo posmatrati i kao tip **Zivotinja** i kao tip **Macka**. Identično važi za klasu **Pas**.



Primer polimorfizma u Java jeziku

```
public class Zivotinja{
}

public class Macka extends Zivotinja{
}

public class Pas extends Zivotinja{
}
```

```
public class Main{
	public static void main(String args[]){
		Zivotinja zivotinje[] = new Zivotinje[3];
		Pas p1 = new Pas();
		Pas p2 = new Pas();
		Macka m1 = new Macka();
		zivotinje[0] = p1;
		zivotinje[1] = p2;
		zivotinje[2] = m;

		for(Zivotinja z: zivotinje){
			System.out.println(z);
		}
	}
}
```

Forsiranje polimorfizma u Python-u

Korišćenjem funkcija možemo simulirati polimorfizam u Python-u. Kako argumentima u okviru funkcije možemo zadavati očekivani tip, navodimo sledeći primer.

```
class Zivotinja:
      pass

class Macka(Zivotinja):
      pass

class Pas(Zivotinja):
      pass
```

```
def jaSamTipa(zivotinja=Zivotinja):
      if isinstance(zivotinja, Macka):
            print("Jesam macka.")
      else:
            print("Nisam macka.")


z1 = Macka()
z2 = Zivotinja()
z3 = Pas()

jaSamTipa(z1)
jaSamTipa(z2)
jaSamTipa(z3)
```