Klasių paveldėjimas (inheritance)

Tai klasės sugebėjimas paveldėti protėvių klasės duomenis ir metodus (**subklasė/vaikas** paveldi iš **superklasės/tėvo**).

Java neturi daugialypio paveldėjimo (netiesiogiai tai galima išspręsti per interfeisus).

Vaikas paveldi visus "matomus" (ne *private*) tėvo kintamuosius ir metodus. Tai vienos krypties paveldėjimas – tėvo klasės objektams vaiko kintamieji ir metodai nepasiekiami (nematomi).

final tipo klasės paveldėti negalima.

Tėvo tipo objektui galima priskirti vaiko tipo objektą (atvirkštinis priskirimas iššaukia *java.lang.ClassCastException* situaciją.

Konstruktoriai nėra nei paveldimi, nei perdengiami.

Atributas **super**

Naudojamas parodyti jog vaikas kreipiasi į tėvą. Naudojama :

- kreipiniui į tėvo klasės konstruktorių (jis turi būti pirmuoju vaiko konstruktoriaus sakiniu):

  super(parametrų-sąrašas);

- kreipiniui į tėvo klasės kintamąjį/metodą (kai sutampa vardai) :

  super.kintamasis / **super.**metodas

PVZ :

```
class A {
    int x;
}
class B extends A {
    int x;
}
class C extends B {
    int x;
    // Šioje C klasėje galime pasiekti visų klasių kintamuosius x
    x = 5;                  //  x iš C klasės
    this.x = 5;             //  x iš C klasės
    ((B)this).x = 5;        //  x iš B klasės
    super.x = 5;            //  x iš B klasės
    ((A)this).x = 5;        //  x iš A klasės
    super.super.x = 5;      //  Klaida
}
```

Analogiškai būtų ir metodams.

Konstruktoriai vykdomi hierarchine tvarka – nuo pagrindinės klasės atgal iki vaiko klasės.

Metodų perdengimas (Overriding = Late binding)

Tai polimorfizmo atvejas, kai tėvo ir vaiko klasė turi vienodus metodus. Būtinos sąlygos : sutampa ne tik metodų vardai, bet jų antraštes ir grąžinamų reikšmių tipai. Metodo pasirinkimą nusako tipas objekto, kuriam šis metodas kviečiamas (jei objektas tėvo tipo, tai bus kviečiamas jos metodas; kitaip – vaiko). Tai atliekama vykdymo metu.

Nepersidengia :

- konstruktoriai;
- **final** tipo metodai;
- statiniai metodai (stainį metodą galima "paslėpti" tik statiniu metodu).

P a v y z d y s

```
/**   Metodų perdengimas (Overriding = Late binding)  ****
      Ploto skaičiavimo metodo įvairioms figūroms perdengimas
       Shema :                    Plokste
                                  /      \
                        Staciakampis   Trikampis
**/
class Plokste {
   protected double ilgis; // private buti negali !
   protected double plotis;
   Plokste () { //  sis konstruktorius reikalingas klasei Trikampis
            This(0.0, 0.0);
   }
   Plokste (double a, double b) { //  konstruktorius
            ilgis  = a;
            plotis = b;
   }
   double plotas() { // ploto skaiciavimo metodas
            System.out.println("Algoritmas dar nesugalvotas");
            return -1;
   }
   static void kasAs() {
            System.out.println("As esu plokstele");
   }
}
class Staciakampis extends Plokste {
   Staciakampis(double a, double b) {
            super(a, b); // kreipinys į konstruktorių Plokste,
                         // nes konstruktoriai nepersidengia
   }
   double plotas() {
            System.out.println("Staciakampis:");
            return ilgis * plotis;
   }
   double perimetras() { // tik sios klases metodas
            return 2*ilgis + 2*plotis;
   }

  static void kasAs() {
            System.out.println("As esu staciakampiukas");
  }
}
class Trikampis extends Plokste {
   Trikampis(double a, double b) {
            // kitas konstruktoriaus variantas - atkartojamas konstruktorius Plolste
            ilgis = a;
            plotis = b;
   }
   double plotas() {
            System.out.println("Trikampis:");
            return (ilgis * plotis) / 2;
```

```
  }
}

class LateTestas {
  public static void main(String args[]) {
            Plokste      plo = new Plokste(5, 5);
            Staciakampis sta = new Staciakampis(7, 8);
            Trikampis    tri = new Trikampis(9, 9);
            Plokste figura;
            //// A.  Per super-klases tipo kintamaji
            System.out.println("      G r u p e  A ");
            figura = plo;
            figura.kasAs();
            System.out.println("  Plotas = " + figura.plotas());
            figura = sta;
            figura.kasAs(); // kvies statini is Plokste
            sta.kasAs();  // kvies statini is Staciakampis
            Staciakampis.kasAs();  // kvies statini is Staciakampis
            System.out.println("  Plotas = " + figura.plotas());
            //System.out.println("  Per " + figura.perimetras()); // Nerasta:
            //Error #: 300 : method perimetras() not found in class Plokste
            figura = tri;
            System.out.println("  Plotas = " + figura.plotas());
            //// B. Tiesiogiai
            System.out.println("      G r u p e  B ");
            System.out.println("  Plotas =  " + plo.plotas());
            System.out.println("  Plotas =  " + sta.plotas());
            System.out.println("  Perimetras  " + sta.perimetras());
            System.out.println("  Plotas =  " + tri.plotas());
            //// C.
            System.out.println("      G r u p e  C ");
            sta = (Staciakampis) plo;
            System.out.println("  Plotas - nevykdo " + sta.plotas());
             // Vykdymo metu java.lang.ClassCastException !
  }
}

/* Rezultatai :
     G r u p e  A
As esu plokstele
Algoritmas dar nesugalvotas
  Plotas = -1.0
As esu plokstele
As esu staciakampiukas
As esu staciakampiukas
Staciakampis:
  Plotas = 56.0
Trikampis:
  Plotas = 40.5
     G r u p e  B
Algoritmas dar nesugalvotas
          Plotas =  -1.0
          Staciakampis:
            Plotas =  56.0
            Perimetras  30.0
          Trikampis:
            Plotas =  40.5
               G r u p e  C
          java.lang.ClassCastException:
Plokste
             at
LateTestas.main(LateTestas.java:81)
          Exception in thread "main"
          */
```