```
Mapper.from(A.class)
      .to(B.class)
      .use( <AnotherMapper> )                                  // früher .useMapper()
      .map(A::getP1).to(B::getP2)                              // früher reassign()
      .map(A::getP1).to(B::getP2)
                    .with( <f(P1->P2)> )                       // früher replace()
                    .skipWhenNull()                            // butSkipWhenNull() ist optional
      .mapThis().to(B::getP3)
                .with( <f(A->P3)> )                            // früher set() mit Funktion f(A->P3)
      .mapCollection(A::getList<P6>).to(B::getList<P7>)
                                    .with( <f(P6->P7)>  )
                                    .skipWhenNull()            // früher replaceCollection()
      .set( Supplier<P4> ).to(B::getP4)                        // ist so geblieben
      .set( <P4>         ).to(B::getP4)                        // ist so geblieben
      .omitInSource(A::getP5)                                  // ist so geblieben
      .omitInDestination(B::getP5)                             // ist so geblieben
      .omitOthers()                                            // ist so geblieben
      .omitOthersSource()                                      // früher omitOtherSourceProperties()
      .omitOthersDestination();                                // früher omitOtherDestinationProperties()

class A {
     P1 p1;
     List<P6> list;
     P5 p5;
     // Getter/Setter & Constructor...
}

class B {
     P2 p2;
     P3 p3;
     List<P7> list;
     P4 p4;
     P5 p5;
     // Getter/Setter & Constructor...
}
```