

Modul Pemrograman Berbasis Objek (PBO)

Inheritance Dan Abstrak Class

Salah satu bagian yang tidak terpisahkan dari OOP adalah inheritance. Konsep inheritance akan banyak digunakan ketika membuat kodingan berorientasi objek.

Lalu apa yang dimaksud dengan inheritance ?

Kalian bisa membayangkan suatu keluarga harmonis yang terdiri dari orang tua dan anak. Seperti layaknya keluarga yang harmonis, orang tua mengajarkan anaknya tentang pengalaman hidup dan tingkah laku mereka. Contohnya, seorang ibu memiliki kemampuan untuk memasak dan mengajarkan anak perempuan nya kemampuan memasak yang dimiliki oleh ibu tersebut. Contoh lain nya, seorang ayah memiliki ilmu dalam manajemen waktu dan mengajarkan anak nya ilmu tersebut.

Hal tersebut lah yang bisa kita sebut sebagai inheritance atau pewarisan. Mudah nya, inheritance adalah bagaimana orang tua mewarisi sifat/kemampuan yang dimiliki nya kepada anak nya.

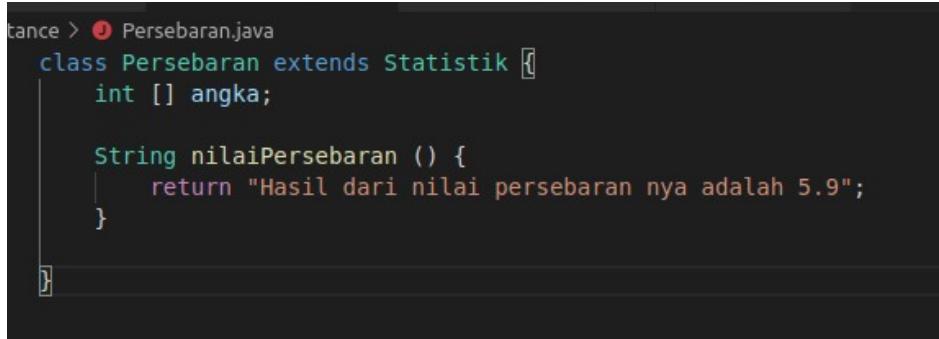
Dalam oop sendiri, kita dapat membuat suatu kelas orang tua atau yang bisa disebut parent dengan berbagai method dan properti lalu menurunkan method dan properti tersebut kepada class anak atau yang biasa disebut child. Pada class child, class anak mewarisi berbagai method atau properti dari class parent, dan dapat menggunakan kemampuan method dan properti tersebut.

Mari kita lihat pada contoh implementasi kodingan nya. Buatlah suatu folder baru bebas dengan nama apa saja, lalu pada folder tersebut buatlah file baru bernama Statistik.java lalu tulis kode dibawah ini ke dalam file tersebut.

```
Inheritance > Statistik.java
1  class Statistik {
2
3      String mean() {
4          return "Hasil mean nya adalah 4.5";
5      }
6
7      void median () {
8          System.out.println("Nilai median adalah 6");
9      }
10
11     String modus() {
12         return "Modus nya adalah 2.5";
13     }
14
15 }
```

class Statistik akan kita gunakan sebagai class parent, dan mempunyai 3 method yaitu mean, median, dan modus. Class tersebut akan mewariskan ke 3 method tersebut kepada child nya.

Masih di folder yang sama, buatlah file baru lagi bernama Persebaran.java lalu tuliskan kode dibawah ini pada file tersebut



```
tance > Persebaran.java
class Persebaran extends Statistik {
    int [] angka;

    String nilaiPersebaran () {
        return "Hasil dari nilai persebaran nya adalah 5.9";
    }
}
```

Class persebaran akan kita gunakan sebagai child atau anak dari class Statistik. Lalu, mungkin kalian dapat melihat diatas ada suatu kode yang belum pernah kita lihat sebelum nya yaitu :

“class Persebaran extends Statistik”

Untuk menurunkan kemampuan class statistik kepada class persebaran, kita dapat menggunakan extends disamping nama class anak nya yaitu persebaran, lalu menuliskan nama orang tua/parent yang akan mewarisi sifat dan kemampuan nya kepada anak nya.

Cukup dengan menambahkan extends kita sudah menurunkan kemampuan class Statistik kepada class Persebaran. Semua yang dimiliki oleh oleh class Statistik baik itu properti maupun method dapat digunakan oleh class Persebaran tanpa perlu membuat kode yang sama pada class persebaran.

Nah, jika kalian melihat pada class Persebaran, terdapat properti angka dan method nilaiPersebaran. angka dan nilaiPersebaran adalah kemampuan yang berasal dari class Persebaran tersebut bukan di turunkan oleh class Statistik.

Mari kita coba buat 1 lagi class yang akan menjadi anak dari class statistik. Buatlah file baru bernama NilaiTengah.java didalam folder yang sama lalu masukan kode dibawah ini pada file tersebut

```
Inheritance > NilaiTengah.java
1  class NilaiTengah extends Statistik {
2
3      String nilaiTengah () {
4          return "Hasil dari nilai tengah adalah 2.3";
5      }
6
7  }
```

Kita telah menurunkan kemampuan class Statistik kepada class NilaiTengah. Sedangkan method nilaiTengah adalah kemampuan class tersebut bukan hasil turunan dari class Statistik.

Sekarang kita akan membuat class utama yang akan menjalankan class Persebaran dan NilaiTengah. Buatlah file bernama Main.java dan tulis kode dibawah ini

```
Inheritance > Main.java
1  class Main {
2      public static void main (String args []) {
3          Persebaran pertama = new Persebaran();
4
5          System.out.println(pertama.nilaiPersebaran());
6          System.out.println(pertama.mean());
7
8          NilaiTengah kedua = new NilaiTengah();
9
10         System.out.println(kedua.nilaiTengah());
11         kedua.median();
12         System.out.println(kedua.modus());
13     }
14 }
```

Pada kode diatas, kita membuat sebuah variable yang akan menjadi objek class persebaran bernama pertama. Dapat kita lihat diatas, variabel pertama memanggil method nilaiPersebaran yang dimiliki nya, lalu memanggil method mean hasil pewarisan dari class statistik. Tapi, jika kalian lihat kode pada class Persebaran, tidak ada kode method mean.

```
Inheritance > NilaiTengah.java
1  class NilaiTengah extends Statistik {
2
3      String nilaiTengah () {
4          return "Hasil dari nilai tengah adalah 2.3";
5      }
6
7 }
```

Lalu, dari mana method mean tersebut? Yap, dari class statistik.

```
Inheritance > Statistik.java
1  class Statistik {
2
3      String mean() {
4          return "Hasil mean nya adalah 4.5";
5      }
6
7      void median () {
8          System.out.println("Nilai median adalah 6");
9      }
10
11     String modus() {
12         return "Modus nya adalah 2.5";
13     }
14
15 }
```

Method mean yang dipanggil oleh objek bernama pertama adalah hasil pewarisan dari class Statistik kepada class Persebaran. class Persebaran tidak membuat method mean pada penulisan kodennya melainkan mewarisi dari class Statistik.

Sedangkan method nilaiPersebaran adalah method yang dibuat langsung pada class Persebaran bukan dari hasil penurunan class Statistik.

Penurunan ini hanya berlaku dari class Parent ke child bukan child ke parent. Oleh karena itu class Statistik tidak dapat menjalankan apapun yang dibuat oleh child nya, sebaliknya child nya dapat menjalankan apapun yang berada pada class Statistik.

Mari kita lihat hasil pada kode yang kita buat di atas di dalam terminal

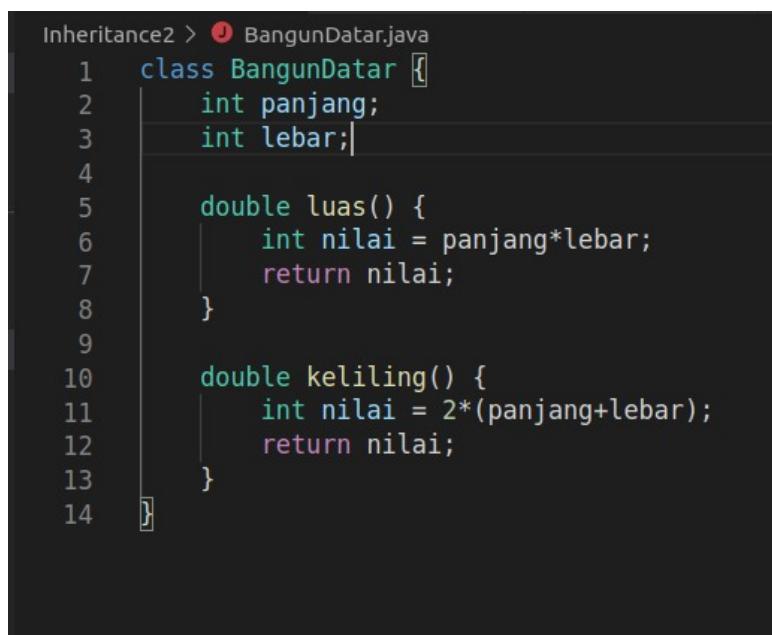
```
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance$ java Main
Hasil dari nilai persebaran nya adalah 5.9
Hasil mean nya adalah 4.5
Hasil dari nilai tengah adalah 2.3
Nilai median adalah 6
Modus nya adalah 2.5
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance$
```

Notes : jika kalian melihat kembali kode pada class Main, kita membuat objek juga untuk class NilaiTengah yang bernama kedua. Penjelasan dari cara kerja objek tersebut sama seperti dengan objek pertama yang mengandung class Persebaran. Kita tidak akan membahas objek yang bernama kedua.

Itulah konsep dari inheritance. Sekarang terdapat pertanyaan berikut :

“Bagaimana semisal, anak mewarisi sifat orang tua nya namun tidak sama persis dengan sifat yang diwarisi. Semisal ibu mengajari anak nya memasak nasi goreng ayam, tetapi anak nya tidak terlalu menyukai ayam pada nasi goreng tersebut dan mengganti nya dengan kambing. Kemampuan membuat nasi goreng nya didapat dari ibu, tetapi anak nya membuat suatu hal baru pada resep tersebut, apakah hal tersebut dapat diterapkan pada oop ? ” Tentu saja bisa

Mari buat folder baru bebas dengan nama apa saja, dan buat file baru pada folder tersebut bernama BangunDatar.java, dan tuliskan kode dibawah ini



```
Inheritance2 > BangunDatar.java
1  class BangunDatar {
2      int panjang;
3      int lebar;
4
5      double luas() {
6          int nilai = panjang*lebar;
7          return nilai;
8      }
9
10     double keliling() {
11         int nilai = 2*(panjang+lebar);
12         return nilai;
13     }
14 }
```

class BangunDatar akan kita jadikan sebagai parent, dan memiliki method luas dan keliling serta properti panjang dan lebar. Rumus method luas dan keliling yang dibuat pada class BangunDatar adalah rumus persegi panjang.

Kita akan membuat class Persegi yang akan menjadi child yang mewarisi class BangunDatar, tetapi kita tidak ingin menggunakan rumus method luas dan keliling dari class BangunDatar karena menggunakan rumus persegi panjang bukan rumus persegi. Kita dapat mengganti rumus pada method luas dan keliling sesuai yang diinginkan oleh class Persegi. Bagaimana caranya ? Mari kita lihat langsung implementasi nya pada Class Persegi

Pada folder yang sama dengan class BangunDatar, buatlah file baru bernama Persegi.java dan tuliskan kode dibawah ini

```
tance2 > Persegi.java
class Persegi extends BangunDatar {
    double sisi;

    public Persegi(double sisi) {
        this.sisi = sisi;
    }

    @Override
    double luas() {
        double nilai = this.sisi * this.sisi;
        return nilai;
    }

    @Override
    double keliling() {
        double nilai = 4 * sisi;
        return nilai;
    }
}
```

Terdapat kata Override pada kode diatas dan kita juga kembali menulis method luas dan keliling pada class Persegi. Lalu, apa arti dari override dan kenapa kita menulis kembali method luas dan keliling yang sudah diwariskan oleh class BangunDatar?

Jawaban nya, override kita gunakan untuk mengganti rumus persegi panjang yang terdapat pada method luas dan keliling hasil pewarisan class BangunDatar menjadi rumus persegi. Dengan override kita dapat mengganti logic dari method yang diwariskan oleh class parent, menjadi sesuai class child nya.

Oleh karena itu, kita kembali menuliskan method luas dan keliling pada class Persegi dengan mengubah logic yang awal nya rumus persegi panjang menjadi rumus persegi.

Sekarang, mari kita buat file baru bernama Main.java dan tuliskan kode dibawah ini pada file tersebut

```
Inheritance2 > Main.java
1  class Main {
2      public static void main(String args[]) {
3          Persegi persegi = new Persegi(5);
4          double luasPersegi = persegi.luas();
5          double kelilingPersegi = persegi.keliling();
6
7          System.out.println("Luas Persegi adalah = " + luasPersegi);
8          System.out.println("Keliling persegi adalah = " + kelilingPersegi);
9
10     }
11 }
```

Lalu, jalankan kode tersebut pada terminal



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance2$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance2$ java Main
Luas Persegi adalah = 25.0
Keliling persegi adalah = 20.0
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance2$
```

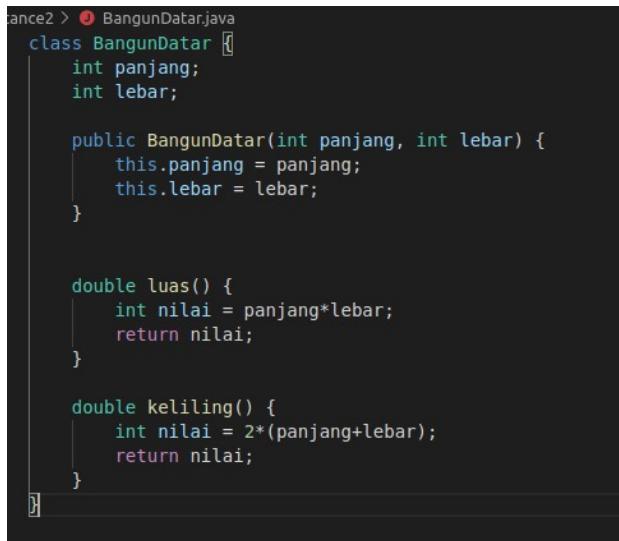
Kita dapat melihat hasil dari kode yang telah kita buat. Luas dan keliling persegi yang dipanggil pada class Main dan ditampilkan pada terminal merupakan method luas dan keliling dari class Persegi yang sudah dirubah rumus nya dengan rumus persegi.

Sekarang, terdapat pertanyaan lagi.

“Pada pembuatan nasi goreng, ibu mewariskan resep nasi goreng ayam kepada anak nya, dan anak nya mengubah resep tersebut menjadi nasi goreng kambing. Suatu hari, anak nya ingin membuat nasi goreng ayam, bukan nasi goreng kambing. Apakah si anak masih dapat membuat nasi goreng ayam ? ” Jawaban nya bisa, karena walaupun anak nya mengubah resep tersebut, anak nya masih mengingat dengan jelas bagaimana membuat nasi goreng ayam

Apakah kita dapat mengimplementasikan hal tersebut pada oop? Tentu saja bisa, mari kita lihat implementasi nya langsung pada kodingan.

Pada file BangunDatar.java, tambahkan kode dibawah ini



```
ance2 > BangunDatar.java
class BangunDatar {
    int panjang;
    int lebar;

    public BangunDatar(int panjang, int lebar) {
        this.panjang = panjang;
        this.lebar = lebar;
    }

    double luas() {
        int nilai = panjang*lebar;
        return nilai;
    }

    double keliling() {
        int nilai = 2*(panjang+lebar);
        return nilai;
    }
}
```

Kita menambahkan constructor pada BangunDatar.java yang bertugas untuk mengisi variabel panjang dan lebar. Lalu, bagaimana mengisi constructor BangunDatar pada class child nya ? Kita akan menggunakan “super”

Buka file Persegi.java, dan tambahkan kode berikut di dalam file tersebut

```
Pertemuan 6' > Inheritance2 > Persegi.java
1  class Persegi extends BangunDatar {
2      double sisi;
3
4      public Persegi(double sisi, int panjang, int lebar) {
5          super(panjang, lebar);
6          this.sisi = sisi;
7      }
8
9      @Override
10     double luas() {
11         double nilai = this.sisi * this.sisi;
12         return nilai;
13     }
14
15     double luasOriginal() {
16         double nilai = super.luas();
17
18         return nilai;
19     }
20
21     @Override
22     double keliling() {
23         double nilai = 4 * sisi;
24
25         return nilai;
26     }
27 }
```

Kita menambahkan parameter panjang dan lebar pada constructor persegi, lalu menambahkan kata super dengan isian parameter panjang dan lebar. Kata super disini akan merujuk constructor pada parent nya class Persegi yaitu constructor class BangunDatar. Isian parameter dari super tersebut, juga akan mengisi parameter construstor class BangunDatar juga.

Pada kode pada class Persegi kita juga menambahkan method baru bernama luasOriginal. Dan jika kita jeli, terdapat kata super.luas(). Kata super akan merujuk kepada parentnya class Persegi yaitu BangunDatar.

super.luas() memiliki arti bahwa kita ingin menggunakan method luas dari class BangunDatar bukan method luas yang sudah kita ganti dengan rumus persegi. Walaupun kita telah merubah rumus method luas yang diwariskan oleh class BangunDatar dengan rumus luas persegi, kita masih bisa mengakses rumus luas original dari class BangunDatar dengan menggunakan kata kunci bernama “super”.

Mari kita buat 1 child lagi yang akan mewarisi class BangunDatar.

Buatlah file baru pada folder yang sama dengan class BangunDatar dan namakan file tersebut dengan PersegiPanjang.java . Masukan kode dibawah ini pada file tersebut

```
heritance2 > PersegiPanjang.java
1  class PersegiPanjang extends BangunDatar {
2      double panjang = 10;
3      double lebar = 20;
4
5      public PersegiPanjang(int panjang, int lebar) {
6          super(panjang, lebar);
7      }
8
9      double luasBangunDatar() {
10         return super.luas();
11     }
12
13     double luasPersegiPanjang() {
14         double nilai = this.panjang * this.lebar;
15
16         return nilai;
17     }
18 }
```

Pada kode diatas, kita membuat constructor PersegiPanjang dan diisi dengan super(panjang, lebar) yang akan merujuk kepada constructor class BangunDatar

Lalu pada method luasBangunDatar, kita menggunakan method super.luas() yang nantinya juga akan merujuk method luas dari class BangunDatar

Sedangkan method luasPersegiPanjang adalah method yang berisi rumus luas persegi panjang yang bukan hasil warisan dari class BangunDatar.

Mari kita lihat hasil perubahan kode yang kita lakukan. Buka file Main.java, lalu tambahkan kode dibawah ini

```
nuan 6' > Inheritance2 > Main.java
class Main {
    public static void main(String args[]) {
        Persegi persegi = new Persegi(5, 10, 2);
        double luasPersegi = persegi.luas();
        double luasOriginal = persegi.luasOriginal();
        double kelilingPersegi = persegi.keliling();

        System.out.println("Luas Persegi adalah = " + luasPersegi);
        System.out.println("Luas Original Bangun datar pada class Persegi adalah = " + luasOriginal);
        System.out.println("Keliling persegi adalah = " + kelilingPersegi);

        PersegiPanjang persegiPanjang = new PersegiPanjang(10, 5);
        double luasBangunDatar = persegiPanjang.luasBangunDatar();
        double luasPersegiPanjang = persegiPanjang.luasPersegiPanjang();

        System.out.println("Luas Bangun Datar adalah = " + luasBangunDatar);
        System.out.println("Luas Persegi Panjang adalah = " + luasPersegiPanjang);
    }
}
```

Dapat kita lihat pada kode diatas, kita menambahkan argumen dengan nilai 10 dan 2 pada objek persegi yang nantinya akan mengisi constructor dari class BangunDatar. Kita juga menjalankan method luasOriginal yang berisi method luas dari warisan class BangunDatar yang belum diubah menjadi rumus luas persegi.

Kita juga menambahkan objek persegiPanjang yang menampung class PersegiPanjang dan menjalankan method yang terdapat pada class tersebut.

Mari kita jalankan kodingan yang telah kita buat pada terminal

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL 1: ba

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance2$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance2$ java Main
Luas Persegi adalah = 25.0
Luas Original Bangun datar pada class Persegi adalah = 20.0
Keliling persegi adalah = 20.0
Luas Bangun Datar adalah = 50.0
Luas Persegi Panjang adalah = 200.0
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/Inheritance2$
```

Kita telah menjalankan kodingan yang telah kita buat pada terminal.

Sekarang masih ada 1 pertanyaan terakhir

“Seorang ayah mengatakan kepada anak-anaknya bahwa mereka harus menjadi seorang yang dermawan. Tetapi, ayah tersebut tidak menjelaskan secara detail dermawan yang seperti apa yang dimaksud alias masih abstrak pengertian dermawan nya.

Anak pertama menganggap bahwa dermawan yang dimaksud adalah dengan memberikan sedekah ke orang yang membutuhkan. Anak kedua berpikir bahwa dermawan yang dimaksud adalah memberikan jajan kepada teman-teman nya yang tidak mendapatkan uang jajan.

Kedua anak pun melaporkan kedermawanan mereka kepada ayahnya. Ayahnya mengatakan bahwa kedermawanan yang kalian lakukan berbeda, tetapi keduanya adalah perilaku yang benar dan tidak ada yang salah. Lalu, salah seorang anak bertanya. Kok bisa kedua nya benar? Sang ayah pun menjawab, ayah ingin kalian mendefinisikan kedermawanan sesuai yang kalian percayai bukan hasil dari kata-kata ayah, karena itu ayah menganggap kedermawanan yang kalian lakukan adalah benar walaupun caranya berbeda. “

Wah, cerita yang mengharukan sekali bukan :D

Sekarang apa relasi nya kepada inheritance ini ? Mari kita berkenalan dengan abstrak pada inheritance.

Kita tahu bahwa konsep inheritance adalah mewariskan kemampuan/sifat dari orang tua ke anaknya. Tetapi kita juga mengetahui bahwa orang tua terkadang hanya memberikan nasihat tanpa mendetailkan maksud dari nasihat tersebut agar anaknya dapat memahami nasihat tersebut secara mandiri. Hal ini dapat kita lakukan di OOP

Pada OOP, kita dapat membuat suatu abstrak class yang nantinya mewariskan hanya nama-nama method saja tetapi tidak memiliki rumus/logika/algoritma pada method tersebut. Yang akan membuat logic adalah class child nya. Masing-masing class child yang mewarisi method dari abstrak class akan mendetailkan method tersebut sesuai dengan kebutuhan mereka. Mari kita lihat langsung caranya pada kodingan

Kita akan menggunakan kembali class BangunDatar, Persegi, dan PersegiPanjang. Tetapi kali ini kita akan membuat didalam folder baru. Kalian bisa copy paste file tersebut didalam folder baru yang kalian buat lalu merubah masing-masing file sesuai implementasi abstrak class yang akan kita kerjakan dibawah ini atau kalian dapat membuat kembali file tersebut dari awal.

Pada class BangunDatar, masukan kode dibawah ini

```
ok > BangunDatar.java
abstract class BangunDatar {
    int panjang;
    int lebar;

    abstract double luas();

    abstract double keliling();
}
```

Kata abstrak pada kode diatas, mengindikasikan bahwa kita membuat sebuah class abstrak dan membuat method abstrak. Kita tidak mendefinisikan bagaimana method tersebut bekerja. Yang akan mendefinisikan bagaimana method itu bekerja adalah class child nya

Pada class Persegi, masukan kode dibawah ini

```
> Persegi.java
class Persegi extends BangunDatar {
    double sisi;

    public Persegi(double sisi) {
        this.sisi = sisi;
    }

    @Override
    double luas() {
        double nilai = this.sisi * this.sisi;
        return nilai;
    }

    @Override
    double keliling() {
        double nilai = 4 * sisi;
        return nilai;
    }
}
```

Kalian bisa lihat perubahan pada kode diatas? Tidak ada perubahan spesifik, hanya perubahan kecil

Untuk mendetailkan bagaimana luas dan keliling yang diwariskan pada class parent bekerja pada class child, kita cukup menambahkan override dan method nya, lalu kita menuliskan kode pada method tersebut sesuai yang diinginkan class child nya.

Tidak ada hal yang baru disini, perbedaanya hanya jika pada sebelum nya kita menggunakan override untuk mengubah rumus method luas dan keliling pada class BangunDatar menjadi persegi sesuai kebutuhan class Persegi.

Kali ini, kita menggunakan override untuk mendetailkan method yang diwarisi pada class abstrak menjadi sesuai dengan kebutuhan class child tersebut. Pada class Persegi, kita mendetailkan method luas dan keliling sesuai dengan rumus persegi.

Sekarang, kita lakukan hal yang sama pada class PersegiPanjang

```
ak > ① PersegiPanjang.java
class PersegiPanjang extends BangunDatar {
    double panjang;
    double lebar;

    public PersegiPanjang(int panjang, int lebar) {
        this.panjang = panjang;
        this.lebar = lebar;
    }

    @Override
    double luas() {
        double nilai = this.panjang * this.lebar;

        return nilai;
    }

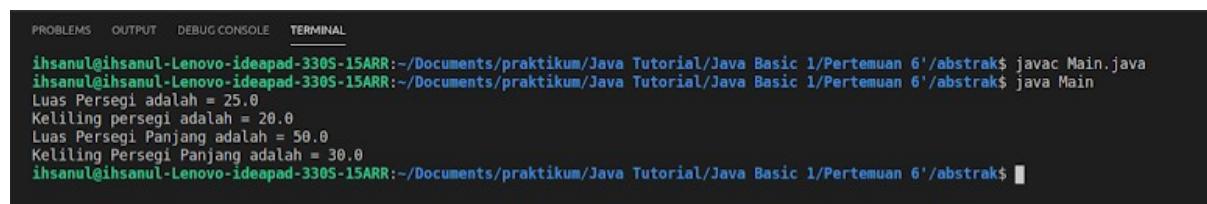
    @Override
    double keliling() {
        double nilai = 2 * (panjang+lebar);

        return nilai;
    }
}
```

Kita mendefinisikan method luas dan keliling sesuai dengan rumus persegi panjang.

Sekarang pada file Main.java, tuliskan kode dibawah ini dan jalankan

```
strak > ① Main.java
1  class Main {
2      public static void main(String args[]) {
3          Persegi persegi = new Persegi(5);
4          double luasPersegi = persegi.luas();
5          double kelilingPersegi = persegi.keliling();
6
7          System.out.println("Luas Persegi adalah = " + luasPersegi);
8          System.out.println("Keliling persegi adalah = " + kelilingPersegi);
9
10         PersegiPanjang persegiPanjang = new PersegiPanjang(10, 5);
11         double luasPersegiPanjang = persegiPanjang.luas();
12         double luasKelilingPanjang = persegiPanjang.keliling();
13
14         System.out.println("Luas Persegi Panjang adalah = " + luasPersegiPanjang);
15         System.out.println("Keliling Persegi Panjang adalah = " + luasKelilingPanjang);
16     }
17 }
```



The screenshot shows a terminal window with the following text:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/abstrak$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/abstrak$ java Main
Luas Persegi adalah = 25.0
Keliling persegi adalah = 20.0
Luas Persegi Panjang adalah = 50.0
Keliling Persegi Panjang adalah = 30.0
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/abstrak$
```

Kode yang telah kita buat berhasil dijalankan. Kita telah berhasil mengimplementasikan abstrak class.

Modul inheritance dan abstrak telah selesai sampai sini.

Notes : Penjelasan dibawah ini diluar materi inheritance dan abstrak, tetapi masih berkaitan dengan materi ini, maka materi dibawah ini tetap kita bahas pada modul ini.

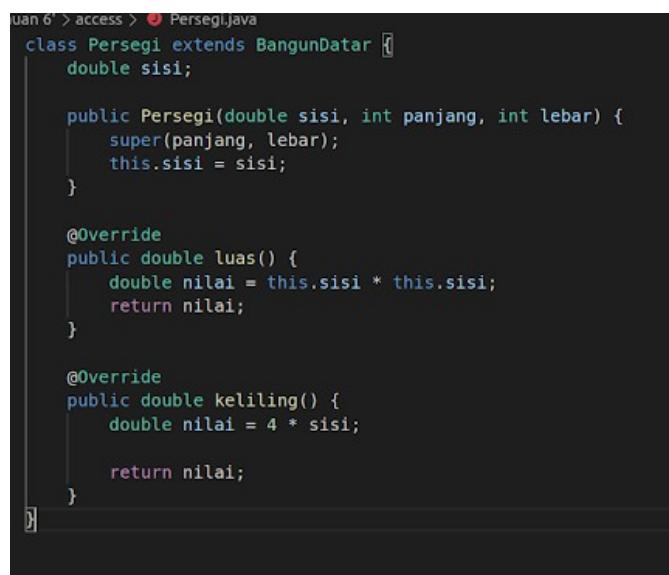
Sekarang kita akan berkenalan dengan access modifier

Access modifier adalah bagaimana kita bisa mengakses suatu class, method, dan properti. Kita bisa membuat sebuah method dan properti hanya bisa diakses class tersebut. Atau kita bisa juga membuat method dan properti nya bisa diakses class tersebut dan class turunan nya. Dan kita juga bisa membuat method dan properti yang dibuat pada suatu class bisa diakses oleh class lain nya tanpa ada batasan

Terdapat 3 access modifier :

- Public, method/properti/class yang memiliki access modifier public maka mereka dapat diakses dimana saja, baik itu turunan nya, menjadi objek, dipanggil di class lain, dan lain sebagai nya
- Private, method/properti/class yang memiliki access modifier private maka mereka hanya dapat diakses di class tersebut
- Protected, method/properti/class yang memiliki access modifier protected maka hanya dapat diakses di class tersebut, turunan class tersebut, dan package class tersebut.

Mari kita lihat contoh pada kodingan



```
uan 6' > access > Persegi.java
class Persegi extends BangunDatar {
    double sisi;

    public Persegi(double sisi, int panjang, int lebar) {
        super(panjang, lebar);
        this.sisi = sisi;
    }

    @Override
    public double luas() {
        double nilai = this.sisi * this.sisi;
        return nilai;
    }

    @Override
    public double keliling() {
        double nilai = 4 * sisi;
        return nilai;
    }
}
```

Kodingan diatas menunjukan bahwa constructor, method luas, dan method keliling memiliki access modifier publik, oleh karena itu kita dapat mengakses constructor dan method nya dimana saja, termasuk class lain.

```
Jan 6' > access > Main.java
class Main {
    public static void main(String args[]) {
        Persegi persegi = new Persegi(5, 10, 2);
        double luasPersegi = persegi.luas();
        double kelilingPersegi = persegi.keliling();

        System.out.println("Luas Persegi adalah = " + luasPersegi);
        System.out.println("Keliling persegi adalah = " + kelilingPersegi);
    }
}
```

diantara kita mengakses method luas dan keliling pada class Main, hal ini dapat kita lakukan karena access modifier pada method luas dan keliling adalah publik.

```
Jan 6' > access > PersegiPanjang.java
class PersegiPanjang extends BangunDatar {
    private double panjang = 10;
    private double lebar = 20;

    public PersegiPanjang(int panjang, int lebar) {
        super(panjang, lebar);
    }

    double luasBangunDatar() {
        return super.luas();
    }

    double luasPersegiPanjang() {
        double nilai = this.panjang * this.lebar;

        return nilai;
    }
}
```

Pada contoh class PersegiPanjang kita membuat properti dari panjang dan lebar memiliki access modifier private. Sehingga, properti tersebut hanya bisa diakses oleh class tersebut, tidak bisa diakses oleh class lain

```
Jan 6' > access > Main.java
class Main {
    public static void main(String args[]) {
        Persegi persegi = new Persegi(5, 10, 2);
        double luasPersegi = persegi.luas();
        double kelilingPersegi = persegi.keliling();

        System.out.println("Luas Persegi adalah = " + luasPersegi);
        System.out.println("Keliling persegi adalah = " + kelilingPersegi);

        PersegiPanjang persegiPanjang = new PersegiPanjang(10, 5);
        PersegiPanjang.panjang = 10;
    }
}
```

Kita lihat di atas, objek persegiPanjang berusaha mengakses properti panjang. Hal ini tidak bisa dilakukan, karena acces modifier dari properti panjang adalah private. Program akan error

```
emuan 6' > access > BangunDatar.java
class BangunDatar {
    int panjang;
    int lebar;

    public BangunDatar(int panjang, int lebar) {
        this.panjang = panjang;
        this.lebar = lebar;
    }

    protected double luas() {
        int nilai = panjang*lebar;
        return nilai;
    }

    protected double keliling() {
        int nilai = 2*(panjang+lebar);
        return nilai;
    }
}
```

Pada contoh diatas, kita menggunakan access modifier protected pada method luas dan keliling di class BangunDatar, protected menandakan bahwa kita dapat menggunakan method tersebut hanya pada class tersebut, class yang menjadi turunan nya/child nya, dan package pada tersebut

```
uan 6' > access > PersegiPanjang.java
class PersegiPanjang extends BangunDatar {
    private double panjang = 10;
    private double lebar = 20;

    public PersegiPanjang(int panjang, int lebar) {
        super(panjang, lebar);
    }

    double luasBangunDatar() {
        return super.luas();
    }

    double luasPersegiPanjang() {
        double nilai = this.panjang * this.lebar;
        return nilai;
    }
}
```

Dapat kita lihat pada kodingan diatas, kita memanggil super.luas yang mengindikasikan bahwa kita memanggil method luas pada class BangunDatar. Hal ini diperbolehkan karena PersegiPanjang adalah class child/turunan dari class BangunDatar.

```
Pertemuan 6' > access > Main.java
1  class Main {
2      public static void main(String args[]) {
3          Persegi persegi = new Persegi(5, 10, 2);
4          double luasPersegi = persegi.luas();
5          double kelilingPersegi = persegi.keliling();
6
7          System.out.println("Luas Persegi adalah = " + luasPersegi);
8          System.out.println("Keliling persegi adalah = " + kelilingPersegi);
9
10         PersegiPanjang persegiPanjang = new PersegiPanjang(10, 5);
11
12         // PersegiPanjang.panjang = 10;
13
14         double luasBangunDatar = persegiPanjang.luasBangunDatar();
15         double luasPersegiPanjang = persegiPanjang.luasPersegiPanjang();
16
17         System.out.println("Luas Bangun Datar adalah = " + luasBangunDatar);
18         System.out.println("Luas Persegi Panjang adalah = " + luasPersegiPanjang);
19     }
}
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/access$ javac Main.java
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/access$ java Main
Luas Persegi adalah = 25.0
Keliling persegi adalah = 20.0
Luas Bangun Datar adalah = 50.0
Luas Persegi Panjang adalah = 200.0
ihsanul@ihsanul-Lenovo-ideapad-330S-15ARR:~/Documents/praktikum/Java Tutorial/Java Basic 1/Pertemuan 6'/access$
```

Diatas adalah kodingan class Main yang menjalankan class Persegi dan PersegiPanjang yang diberi access Modifier.

Method panjang pada objek persegiPanjang dijadikan komentar karena ketika dijalankan akan menghasilkan error yang disebabkan access modifier pada method tersebut adalah private.

Tugas

1. Buat atau cari studi kasus tentang inheritance, implementasikan studi kasus tersebut dalam bentuk kode
2. Screenshot kode dan hasil kode yang dijalankan. Masukan hasil screenshot tersebut dan jelaskan kode, hasil kode, dan studi kasus yang dibuat.

Tugas dikirim pada elen, dengan format sebagai berikut

1. Pada penamaan file gunakan format berikut

Praktikum_keberapa_Judul Praktikum_Nama_Nim

Contoh :

Praktikum01_Perkenalan-Java_IhsanulFikriAbiyyu_0220318021

2. Format laporan bertipe pdf
3. Pada cover praktikum, masukan beberapa hal dibawah ini
 - Praktikum ke berapa
 - Judul Praktikum
 - Nama
 - Nim
4. Tenggat waktu mengerjakan adalah 2 pekan semenjak tugas diberikan
5. Diperbolehkan berkelompok tetapi hanya antara yang tidak punya laptop/komputer dengan yang punya laptop/komputer. Selain itu tidak boleh