

# **LAPORAN 10**

## **Pemrograman Berorientasi Objek**

**“Interface”**

Dosen Pengampu : **Paulina Heruningsih Prima Rosa.**



**DIBUAT OLEH :**

Nama : Yohanis Celvin D.P.U.Pati  
NIM : 245314033

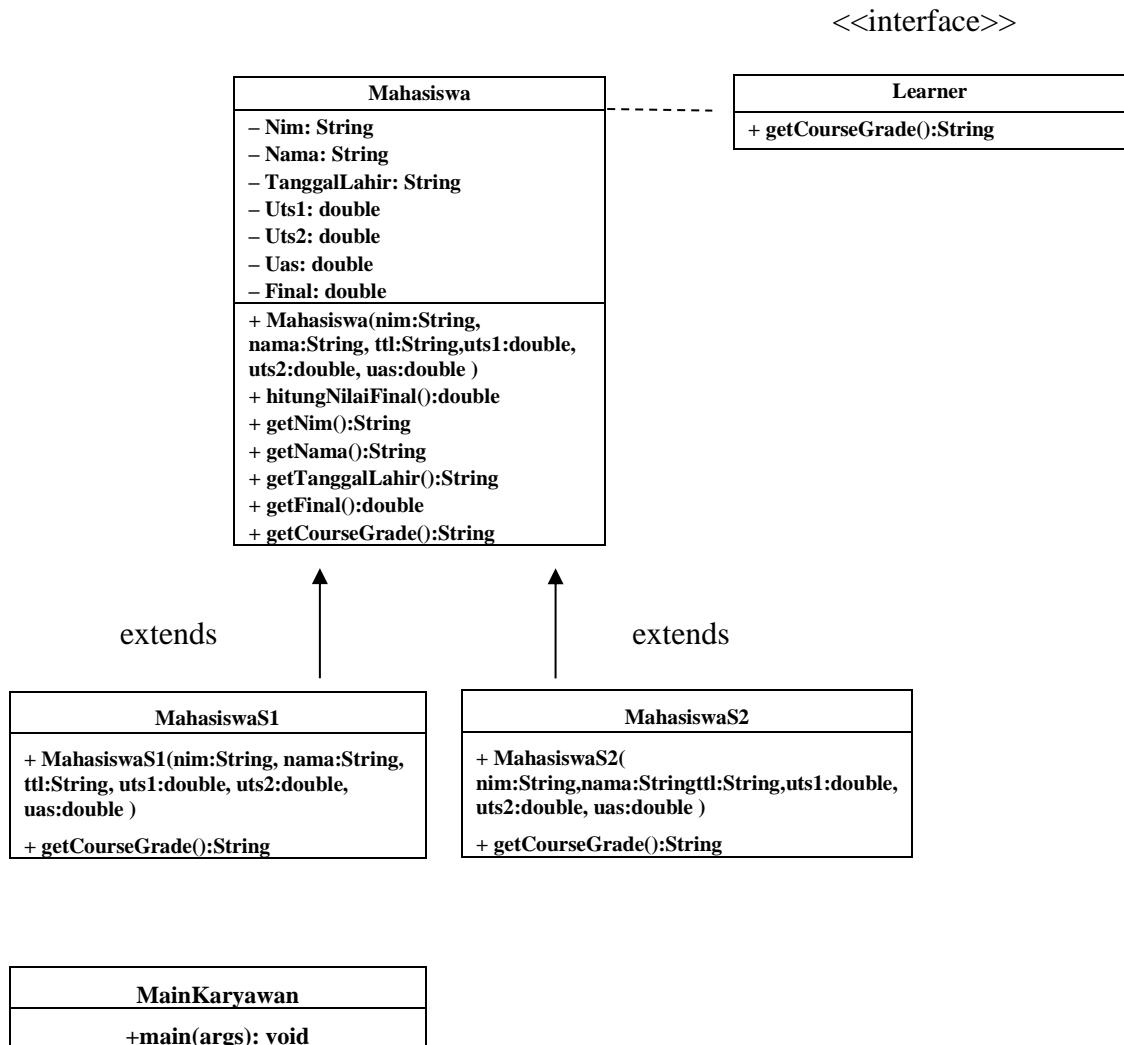
**KELAS : BP**

**PROGRAM STUDI INFORMATIKA  
FAKULTAS SAINS DAN TEKNOLOGI  
UNIVERSITAS SANATA DHARMA  
YOGYAKARTA  
2024**

# Kasus 1

## A. DIAGRAM CLASS

Penggambaran diagram kelas



## B. LISTING PROGRAM

```

Source History Help File Edit View Insert Tools Window Help
1 * Author Calvin Pati
2
3 class Mahasiswa implements Learner {
4
5     protected String Nim; // Nomor Tanda Mahasiswa
6     protected String Nama; // Nama lengkap mahasiswa
7     protected String TanggalLahir; // Format DD-MM-YYYY
8     protected double Uts1; // Nilai Ujian Tengah Semester 1 (0-100)
9     protected double Uts2; // Nilai Ujian Tengah Semester 2 (0-100)
10    protected double Uas; // nilai ujian Akhir semester (0-100)
11    protected double Final; // nilai akhir yang ditentukan oleh UTS1, UTS2, dan UAS
12
13    public Mahasiswa(String Nim, String Nama, String TanggalLahir,
14                      double Uts1, double Uts2, double Uas) {
15        this.Nim = Nim;
16        this>Nama = Nama;
17        this.TanggalLahir = TanggalLahir;
18        this.Uts1 = Uts1;
19        this.Uts2 = Uts2;
20        this.Uas = Uas;
21        this.Final = Final();
22    }
23
24    @Override
25    public void hitungNilaiFinal() { // untuk nittng nilai final secara otomatis
26        if (return nilai akhir dengan bobot 30% UTS1, 30% UTS2, dan 40% UAS
27        punto omnihitungNilaiFinal() {
28            return (Uts1 * 0.3) + (Uts2 * 0.3) + (0.4 * Uas);
29        }
30    }
31
32    // Implementasi metode dari interface Learner
33    @Override
34    public String getCourseGrade() {
35        if (Final >= 85) return "A";
36        else if (Final >= 70) return "B";
37        else if (Final >= 56) return "C";
38        else if (Final >= 45) return "D";
39        else return "E";
40    }
41
42    // getter untuk memakses data privat
43    public String getNim() { return Nim; }
44    public String getNama() { return Nama; }
45    public String getTanggalLahir() { return TanggalLahir; }
46    public double getFinal() { return Final; }
47
48 }
49
50
51 /**
52 * 
53 * @author Calvin Pati
54 */
55 class MahasiswaS1 extends Mahasiswa {
56
57     // Konstruktor panggil konstruktor kelas induk
58     public MahasiswaS1(String Nim, String Nama, String TanggalLahir,
59                         double Uts1, double Uts2, double Uas) {
60         super(Nim, Nama, TanggalLahir, Uts1, Uts2, Uas);
61     }
62
63
64    // Override metode getCourseGrade()
65    @Override
66    public String getCourseGrade() {
67        return super.getCourseGrade();
68    }
69 }
70
71
72 import java.util.Scanner;
73
74 /**
75 * 
76 * @author Calvin Pati
77 */
78 import java.util.Scanner;
79 public class MainMahasiswa {
80
81     public static void main(String[] args) {
82         Scanner input = new Scanner(System.in);
83
84         // Input Data Mahasiswa S1
85         System.out.println("==> INPUT DATA MAHASISWA S1 ==>");
86         System.out.print("Masukkan NIM : ");
87         String nim1 = input.nextLine();
88         System.out.print("Masukkan Nama : ");
89         String namal = input.nextLine();
90         System.out.print("Masukkan Tanggal Lahir : ");
91         String tll = input.nextLine();
92         System.out.print("Masukkan Nilai UTS1 : ");
93         double uts1_1 = input.nextDouble();
94         System.out.print("Masukkan Nilai UTS2 : ");
95         double uts2_1 = input.nextDouble();
96         System.out.print("Masukkan Nilai UAS : ");
97         double uas1 = input.nextDouble();
98         input.nextLine(); // buang newline
99
100        // objek MahasiswaS1
101        MahasiswaS1 mhs1 = new MahasiswaS1(nim1, namal, tll, uts1_1, uts2_1, uas1);
102
103        // Data Mahasiswa S2
104        System.out.println("\n==> INPUT DATA MAHASISWA S2 ==>");
105        System.out.print("Masukkan NIM : ");
106        String nim2 = input.nextLine();
107        System.out.print("Masukkan Nama : ");
108        String naga2 = input.nextLine();
109
110        System.out.print("Masukkan Tanggal Lahir : ");
111        String tll2 = input.nextLine();
112        System.out.print("Masukkan Nilai UTS1 : ");
113        double uts1_2 = input.nextDouble();
114        System.out.print("Masukkan Nilai UTS2 : ");
115        double uts2_2 = input.nextDouble();
116        System.out.print("Masukkan Nilai UAS : ");
117        double uas2 = input.nextDouble();
118
119        //objek MahasiswaS2
120        MahasiswaS2 mhs2 = new MahasiswaS2(nim2, nama2, tll2, uts1_2, uts2_2, uas2);
121
122        System.out.println("==> HASIL PENGHITUNGAN NILAI ==>");
123        System.out.println("NIM1: " + mhs1.getNim());
124        System.out.println("Nilai: " + mhs1.getFinal());
125        System.out.println("Nama: " + mhs1.getNama());
126        System.out.println("Tanggal Lahir: " + mhs1.getTanggalLahir());
127        System.out.println("Grade: " + mhs1.getCourseGrade());
128
129        System.out.println("NIM2: " + mhs2.getNim());
130        System.out.println("Nama: " + mhs2.getNama());
131        System.out.println("Tanggal Lahir: " + mhs2.getTanggalLahir());
132        System.out.println("Nilai: " + mhs2.getFinal());
133        System.out.println("Grade: " + mhs2.getCourseGrade());
134
135        input.close();
136    }
137
138
139 }
140
141
142 /**
143 * 
144 * @author Calvin Pati
145 */
146 class MahasiswaS2 extends Mahasiswa {
147
148     // Konstruktor untuk panggil konstruktor kelas induk
149     public MahasiswaS2(String Nim, String Nama, String TanggalLahir,
150                         double Uts1, double Uts2, double Uas) {
151         super(Nim, Nama, TanggalLahir, Uts1, Uts2, Uas);
152     }
153
154
155    // Override metode getCourseGrade()
156    @Override
157    public String getCourseGrade() {
158        if (Final >= 85) return "A";
159        else if (Final >= 70) return "B";
160        else if (Final >= 56) return "C";
161        else if (Final >= 45) return "D";
162        else return "E";
163    }
164
165 }
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
8010
8011
8012
8013
8014
8015
8016
8017
8018
8019
8020
8021
8022
8023
8024
8025
8026
8027
8028
8029
8030
8031
8032
8033
8034
8035
8036
8037
8038
8039
80310
80311
80312
80313
80314
80315
80316
80317
80318
80319
80320
80321
80322
80323
80324
80325
80326
80327
80328
80329
80330
80331
80332
80333
80334
80335
80336
80337
80338
80339
80340
80341
80342
80343
80344
80345
80346
80347
80348
80349
80350
80351
80352
80353
80354
80355
80356
80357
80358
80359
80360
80361
80362
80363
80364
80365
80366
80367
80368
80369
803610
803611
803612
803613
803614
803615
803616
803617
803618
803619
803620
803621
803622
803623
803624
803625
803626
803627
803628
803629
803630
803631
803632
803633
803634
803635
803636
803637
803638
803639
803640
803641
803642
803643
803644
803645
803646
803647
803648
803649
803650
803651
803652
803653
803654
803655
803656
803657
803658
803659
803660
803661
803662
803663
803664
803665
803666
803667
803668
803669
8036610
8036611
8036612
8036613
8036614
8036615
8036616
8036617
8036618
8036619
8036620
8036621
8036622
8036623
8036624
8036625
8036626
8036627
8036628
8036629
8036630
8036631
8036632
8036633
8036634
8036635
8036636
8036637
8036638
8036639
8036640
8036641
8036642
8036643
8036644
8036645
8036646
8036647
8036648
8036649
8036650
8036651
8036652
8036653
8036654
8036655
8036656
8036657
8036658
8036659
8036660
8036661
8036662
8036663
8036664
8036665
8036666
8036667
8036668
8036669
80366610
80366611
80366612
80366613
80366614
80366615
80366616
80366617
80366618
80366619
80366620
80366621
80366622
80366623
80366624
80366625
80366626
80366627
80366628
80366629
80366630
80366631
80366632
80366633
80366634
80366635
80366636
80366637
80366638
80366639
80366640
80366641
80366642
80366643
80366644
80366645
80366646
80366647
80366648
80366649
80366650
80366651
80366652
80366653
80366654
80366655
80366656
80366657
80366658
80366659
80366660
80366661
80366662
80366663
80366664
80366665
80366666
80366667
80366668
80366669
803666610
803666611
803666612
803666613
803666614
803666615
803666616
803666617
803666618
803666619
803666620
803666621
803666622
803666623
803666624
803666625
803666626
803666627
803666628
803666629
803666630
803666631
803666632
803666633
803666634
803666635
803666636
803666637
803666638
803666639
803666640
803666641
803666642
803666643
803666644
803666645
803666646
803666647
803666648
803666649
803666650
803666651
803666652
803666653
803666654
803666655
803666656
803666657
803666658
803666659
803666660
803666661
803666662
803666663
803666664
803666665
803666666
803666667
803666668
803666669
8036666610
8036666611
8036666612
8036666613
8036666614
8036666615
8036666616
8036666617
8036666618
8036666619
8036666620
8036666621
8036666622
8036666623
8036666624
8036666625
8036666626
8036666627
8036666628
8036666629
8036666630
8036666631
8036666632
8036666633
8036666634
8036666635
8036666636
8036666637
8036666638
8036666639
8036666640
8036666641
8036666642
8036666643
8036666644
8036666645
8036666646
8036666647
8036666648
8036666649
8036666650
8036666651
8036666652
8036666653
8036666654
8036666655
8036666656
8036666657
8036666658
8036666659
8036666660
8036666661
8036666662
8036666663
8036666664
8036666665
8036666666
8036666667
8036666668
8036666669
80366666610
80366666611
80366666612
80366666613
80366666614
80366666615
80366666616
80366666617
80366666618
80366666619
80366666620
80366666621
80366666622
80366666623
80366666624
80366666625
80366666626
80366666627
80366666628
80366666629
80366666630
80366666631
80366666632
80366666633
80366666634
80366666635
80366666636
80366666637
80366666638
80366666639
80366666640
80366666641
80366666642
80366666643
80366666644
80366666645
80366666646
80366666647
80366666648
80366666649
80366666650
80366666651
80366666652
80366666653
80366666654
80366666655
80366666656
80366666657
80366666658
80366666659
80366666660
80366666661
80366666662
80366666663
80366666664
80366666665
80366666666
80366666667
80366666668
80366666669
803666666610
803666666611
803666666612
803666666613
803666666614
803666666615
803666666616
803666666617
803666666618
803666666619
803666666620
803666666621
803666666622
803666666623
803666666624
803666666625
803666666626
803666666627
803666666628
803666666629
803666666630
803666666631
803666666632
803666666633
803666666634
803666666635
803666666636
803666666637
803666666638
803666666639
803666666640
803666666641
803666666642
803666666643
803666666644
803666666645
803666666646
803666666647
803666666648
803666666649
803666666650
803666666651
803666666652
803666666653
803666666654
803666666655
803666666656
803666666657
803666666658
803666666659
803666666660
803666666661
803666666662
803666666663
803666666664
803666666665
803666666666
803666666667
803666666668
803666666669
8036666666610
8036666666611
8036666666612
8036666666613
8036666666614
8036666666615
8036666666616
8036666666617
8036666666618
8036666666619
8036666666620
8036666666621
8036666666622
8036666666623
8036666666624
8036666666625
8036666666626
8036666666627
8036666666628
8036666666629
8036666666630
8036666666631
8036666666632
8036666666633
8036666666634
8036666666635
8036666666636
8036666666637
8036666666638
8036666666639
8036666666640
8036666666641
8036666666642
8036666666643
8036666666644
8036666666645
8036666666646
8036666666647
8036666666648
8036666666649
8036666666650
8036666666651
8036666666652
8036666666653
8036666666654
8036666666655
8036666666656
8036666666657
8036666666658
8036666666659
8036666666660
8036666666661
8036666666662
8036666666663
8036666666664
8036666666665
8036666666666
8036666666667
8036666666668
8036666666669
80366666666610
80366666666611
80366666666612
80366666666613
80366666666614
80366666666615
80366666666616
80366666666617
80366666666618
80366666666619
80366666666620
80366666666621
80366666666622
80366666666623
80366666666624
80366666666625
80366666666626
80366666666627
80366666666628
80366666666629
80366666666630
80366666666631
80366666666632
80366666666633
80366666666634
80366666666635
80366666666636
80366666666637
80366666666638
80366666666639
80366666666640
80366666666641
80366666666642
80366666666643
80366666666644
80366666666645
80366666666646
80366666666647
80366666666648
80366666666649
80366666666650
80366666666651
80366666666652
80366666666653
80366666666654
80366666666655
80366666666656
80366666666657
80366666666658
80366666666659
80366666666660
80366666666661
80366666666662
80366666666663
80366666666664
80366666666665
80366666666666
80366666666667
80366666666668
80366666666669
803666666666610
803666666666611
803666666666612
803666666666613
803666666666614
803666666666615
803666666666616
803666666666617
8036666666666
```

## C. OUTPUT Unit Test nya

```

cd C:\Users\Windows\OneDrive\Documents\NetBeansProjects\Modul10PBO; "JAVA_HOME=C:\\Program Files\\Java\\jdk-23" cmd /c
Scanning for projects...
-----< com.mycompany:Modul10PBO >-----
Building Modul10PBO 1.0-SNAPSHOT
  from pom.xml
-----[ jar ]-----
--- resources:3.3.1:resources (default-resources) @ Modul10PBO ---
skip non existing resourceDirectory C:\Users\Windows\OneDrive\Documents\NetBeansProjects\Modul10PBO\src\main\resources
--- compiler:3.13.0:compile (default-compile) @ Modul10PBO ---
Recompiling the module because of changed source code.
Compiling 11 source files with javac [debug release 23] to target\classes
--- exec:3.1.0:exec (default-cli) @ Modul10PBO ---
--- INPUT DATA MAHASISWA S1 ---
Masukan NIM : 245413029
Masukan Nama : Daniel
Masukan Tanggal Lahir : 21-05-2003
Masukan Nilai UTSL : 90
Masukan Nilai UTSL : 80
Masukan Nilai UAS : 79
--- INPUT DATA MAHASISWA S2 ---
Masukan NIM : 235413021
Masukan Nama : Dani
Masukan Tanggal Lahir : 13-07-2002
Masukan Nilai UTSL : 99
Masukan Nilai UTSL : 87
Masukan Nilai UAS : 87
--- HASIL PERHITUNGAN NILAI ---
MAHASISWA S1:
NIM : 245413029
Nama : Daniel
Tanggal Lahir : 21-05-2003
Nilai Final : 82.6
Grade : B
MAHASISWA S2:
NIM : 235413021
Nama : Dani
Tanggal Lahir : 13-07-2002
Nilai Final : 90.6
Grade : A
BUILD SUCCESS
Total time: 01:32 min
Finished at: 2025-05-24T14:46:55+07:00
-----
```

## D. ANALISA

- Interface Learner

No	Syntax	penjelasan
1	import java.util.Scanner;	// impor class Scanner dari Java Utility Library untuk input dari pengguna.
2	interface Learner { String getCourseGrade(); }	//Interface Learner mendefinisikan kontrak berupa method getCourseGrade() yang harus diimplementasikan oleh setiap kelas yang mengimplementasikannya.

- **Class Mahasiswa**

No	Syntax	penjelasan
1	class Mahasiswa implements Learner	// Kelas induk Mahasiswa mengimplementasikan Learner, yang artinya wajib menyediakan implementasi getCourseGrade().
2	protected String Nim; ... protected double Final;	// Atribut yang menyimpan data mahasiswa (NIM, nama, tanggal lahir, nilai ujian). Dideklarasikan protected agar bisa diakses oleh subclass-nya (MahasiswaS1 dan MahasiswaS2).
3	public Mahasiswa(...){ ... }	//Konstruktor Mahasiswa yang menerima data input dan langsung menghitung nilai akhir (Final).
4	public double hitungNilaiFinal()	//Method untuk menghitung nilai akhir dari UTS1 (30%), UTS2 (30%), dan UAS (40%).
5	public String getCourseGrade()	//Implementasi dari method interface Learner. Memberikan konversi nilai akhir menjadi huruf untuk Mahasiswa S1.
6	public String getNim() { ... } dsb.	//Getter method untuk mengakses atribut Nim, Nama, TanggalLahir, dan Final dari luar class.

- **Class MahasiswaS1**

No	Syntax	penjelasan
1	class MahasiswaS1 extends Mahasiswa	// Subclass MahasiswaS1 mewarisi dari class Mahasiswa.
2	super(Nim, Nama, TanggalLahir, Uts1, Uts2, Uas);	//Konstruktor subclass yang memanggil konstruktor superclass untuk inisialisasi data.
3	@Override public String getCourseGrade()	//Method ini override method dari Mahasiswa, tapi hanya memanggil ulang (super.getCourseGrade()), karena logikanya sama.

- **Class MahasiswaS2**

No	Syntax	penjelasan
1	class MahasiswaS2 extends Mahasiswa	// Subclass MahasiswaS2, yang juga mewarisi Mahasiswa, tetapi mengubah cara konversi nilai.
2	public MahasiswaS2(String Nim, String Nama, String TanggalLahir, double Uts1, double Uts2, double Uas)	//Konstruktor MahasiswaS2 yang menggunakan parameter untuk mengisi data mahasiswa dan langsung memanggil konstruktor superclass dengan super(...).
3	super(Nim, Nama, TanggalLahir, Uts1, Uts2, Uas);	//Memanggil konstruktor dari Mahasiswa agar data mahasiswa S2 terinisialisasi dengan baik.
4	@Override public String getCourseGrade()	//Override method getCourseGrade() dengan standar penilaian lebih tinggi.
5	if (Final >= 85) return "A"; ...	//Logika konversi nilai Mahasiswa S2. Nilai minimal untuk A lebih tinggi dibandingkan Mahasiswa S1 (85 vs 80).

- **Class MainMahasiswa**

No	Syntax	penjelasan
1	Scanner input = new Scanner(System.in);	// Membuat objek Scanner untuk menerima input dari user melalui terminal.
2	System.out.println("== INPUT DATA MAHASISWA S1 ==");	//Menampilkan judul sebagai petunjuk kepada user untuk memasukkan data Mahasiswa S1.
3	String nim1 = input.nextLine(); ... double uas1 = input.nextDouble();	//Membaca input data Mahasiswa S1 (NIM, nama, tanggal lahir, nilai UTS1, UTS2, dan UAS).
4	input.nextLine();	//Menghapus karakter newline (\n) yang tertinggal setelah membaca angka.
5	MahasiswaS1 mhs1 = new MahasiswaS1(...);	//Membuat objek Mahasiswa S1 dengan data input.
6	System.out.println("== INPUT DATA MAHASISWA S2 ==");	//Menampilkan petunjuk untuk input Mahasiswa S2.
7	String nim2 = input.nextLine(); ... double uas2 = input.nextDouble();	//Membaca input untuk Mahasiswa S2.
8	MahasiswaS2 mhs2 = new MahasiswaS2(...);	//Membuat objek Mahasiswa S2 dengan data input yang telah diberikan.

9	System.out.println("\n==== HASIL PERHITUNGAN NILAI ===");	//Menampilkan header output hasil perhitungan.
10	System.out.println("MAHASISWA S1: ...")	//Menampilkan detail data Mahasiswa S1, nilai akhirnya, dan grade-nya.
11	System.out.println("MAHASISWA S2: ...")	//Menampilkan detail data Mahasiswa S2, nilai akhirnya, dan grade-nya.
12	input.close();	//Menutup objek Scanner untuk membersihkan resource.

➤ **Penjelasan alur jalannya program, penggunaan, dan manfaat interface dalam program:**

1) Kelas Learner (Interface)

Kelas ini adalah kontrak khusus yang diimplementasi dari kelas-kelas apapun yang mau dianggap sebagai “pembelajaran”

Fungsinya untuk memaksa kelas turunan (Mahasiswa) supaya menyediakan cara untuk mengubah nilai numerik menjadi grade huruf (A-E).

2) Kelas Mahasiswa (Supperclass)

Kelas ini adalah kelas dasar untuk MahasiswaS1 dan MahasiswaS2, dan kelas ini mengimplementasi interface Learner, jadi wajib menyediakan implementasi getCourseGrade().

Fungsionalnya:

- Konstruktor untuk mengisi semua atribut saat objek dibuat.
- hitungNilaiFinal() untuk menghitung nilai akhir dari 30% UTS1, 30% UTS2, 40% UAS.
- getCourseGrade() untuk mengonversi nilai akhir menjadi huruf berdasarkan standar mahasiswa S1.

3) Kelas MahasiswaS1

Kelas ini adalah subclass dari Mahasiswa yang mewakili jenjang S1, dan kelas ini menggunakan metode Override, metode getCourseGrede(), meskipun logikanya sama dengan kelas induk, tujuannya supaya bisa menyesuaikan jika nanti kriteria penilaian S1 berubah, dan kelas ini tetap memakai logikakonversi nilai default ( $A \geq 80$ ,  $B \geq 70$ , dst).

#### 4) Kelas MahasiswaS2

Kelas ini juga subclass dari Mahasiswa dan kelas ini mewakili mahasiswa jenjang S2, kelas ini juga memakai metode override metode getCourseGrade() dengan logika yang lebih ketat, misalnya: A  $\geq$  85 (lebih tinggi dari S1) dan B  $\geq$  70, C  $\geq$  56, dst.

Perubahan logika ini memungkinkan adanya penyesuaian standar akademik untuk tingkat pendidikan yang lebih tinggi.

#### 5) Kelas MainMahasiswa (Fungsi main())

Kelas ini berfungsi sebagai titik awal program (entry point), fungsinya untuk Meminta input dari user untuk dua mahasiswa: satu S1 dan satu S2, Membuat objek MahasiswaS1 dan MahasiswaS2 berdasarkan input, Menampilkan data mahasiswa beserta nilai final dan grade-nya, dan output yang ditampilkan memberi gambaran hasil konversi nilai tiap jenjang.

### ➤ Penggunaan dan Manfaat Interface

1) Interface Learner interface dipakai di baris ini:

```
interface Learner {  
    String getCourseGrade();  
}
```

Yang kemudian diimplementasikan oleh:

```
class Mahasiswa implements Learner {  
    ...  
}
```

No	Manfaat	penjelasan
1	Polimorfisme	// Dengan interface Learner, kita bisa membuat koleksi objek Learner (misalnya array) yang berisi objek MahasiswaS1 dan MahasiswaS2, lalu memanggil getCourseGrade() tanpa peduli tipenya.

2	Reusabilitas dan Fleksibilitas	//Jika nanti dibuat jenis mahasiswa baru (misalnya MahasiswaDiploma), cukup implementasikan Learner, tanpa mengubah kode lama.
3	Pemisahan Kontrak dan Implementasi	//Interface mendefinisikan "apa yang harus dilakukan", bukan "bagaimana". Implementasi diserahkan ke masing-masing kelas.
4	Kode Lebih Modular	//Kode menjadi terstruktur dan modular karena logika konversi nilai bisa diubah per kelas tanpa menyentuh interface.

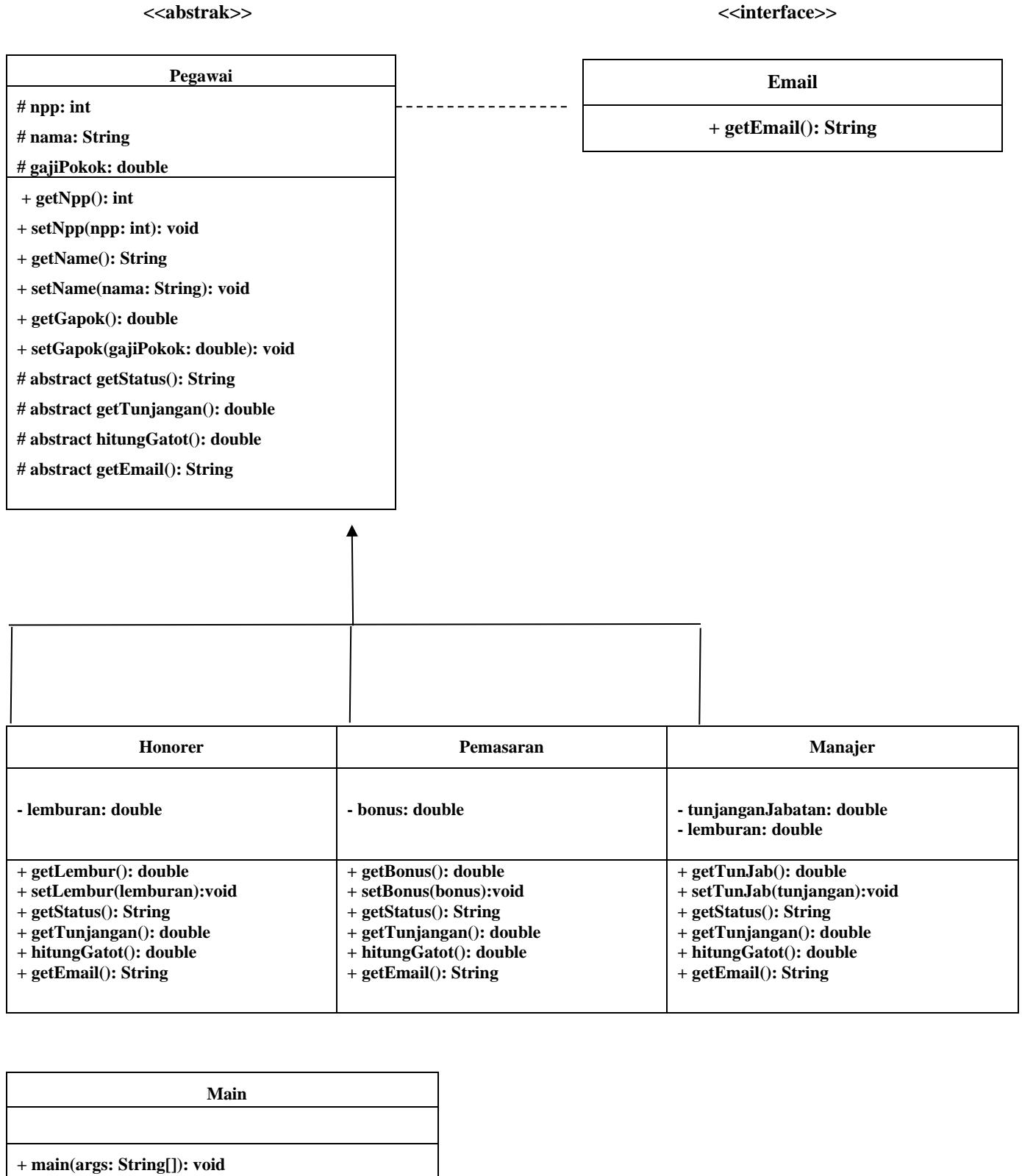
## ➤ Perbandingan

aspek	Versi Tanpa Interface	Versi Dengan Interface (Learner)
Struktur Kelas	Hanya ada inheritance (kelas induk Mahasiswa, dan subclass MahasiswaS1/MahasiswaS2).	Menggunakan inheritance + <b>interface Learner</b> yang mengharuskan method getCourseGrade() diimplementasikan.
Keterikatan antar kelas	Kelas MahasiswaS1 dan MahasiswaS2 bergantung langsung ke kelas Mahasiswa.	Kelas-kelas tetap tergantung ke Mahasiswa, tetapi kontrak perilaku (nilai → grade) ditetapkan oleh Learner.
Polimorfisme (berbasis interface)	Tidak mendukung penggunaan objek secara polymorphic berdasarkan perilaku (getCourseGrade()).	Mendukung polimorfisme berbasis interface, misalnya kita bisa buat: List<Learner> daftarMahasiswa = new ArrayList<>(); lalu

		isinya bisa MahasiswaS1, MahasiswaS2, dst.
Ekstensi program	Sulit dikembangkan jika ingin menambahkan jenis mahasiswa baru (misal: D3, mahasiswa exchange, dll), karena harus bergantung ke kelas Mahasiswa.	Lebih mudah dikembangkan: tinggal buat kelas baru yang mengimplementasikan Learner saja, tidak wajib mewarisi Mahasiswa.
Keterbukaan terhadap perubahan (Open/Closed Principle)	Jika logika getCourseGrade() diubah, perlu menyentuh langsung kelas Mahasiswa atau turunannya.	Logika perubahan dapat dilakukan per kelas (terpisah) tanpa mengubah interface Learner.
Konsistensi perilaku	Tidak ada jaminan semua subclass memiliki metode getCourseGrade().	Ada <b>kontrak wajib</b> yang menjamin semua kelas yang "belajar" pasti punya getCourseGrade().
Reusabilitas	Terbatas pada hierarki warisan (extends).	Tinggi – berbagai kelas bisa mengimplementasikan interface Learner, meskipun tidak berhubungan secara hierarki kelas.
Kode lebih fleksibel?	Tidak fleksibel.	Ya, fleksibel dan scalable.

## Kasus 2

### A. DIAGRAM CLASS



## B. LISTING PROGRAM

```
Source History [C] Calvin S. [S] [E] [D] [F] [G] [H] [I] [L] [J] [K] [M] [N] [O] [P] [Q] [R] [U] [V] [W] [X] [Y] [Z] [B] [C] [D] [E] [F] [G] [H] [I] [L] [M] [N] [O] [P] [Q] [R] [U] [V] [W] [X] [Y] [Z]
```

9 \* Author Clevati Pati

10 \*/

11 class Konselor extends Pegawai {

12 protected double lembur(); // Jumlah lembur

13 // Untuk lembur

14 public double getLembur() {

15 return lembur();

16 }

17 // Setter lemburan

18 public void setLemburan(double lembur) {

19 this.lembur = lembur;

20 }

21 // Implementasi status

22 @Override

23 String getstatus() {

24 return "Konselor";

25 }

26 // Implementasi tunjangan

27 @Override

28 double getTunjangan() {

29 return lembur();

30 }

31 // Implementasi total gaji

32 @Override

33 double hitungGaji() {

34 return getGajikot + lembur();

35 }

36 // Implementasi email: nippkonselor.usd.ac.id

37 @Override

38 public String getEmail() {

39 public String getEmail() {

40 return getNip() + "konselor.usd.ac.id";

41 }

42 }

```
9  * @author Calvin Pati
10 */
11 class Penasaran extends Pegawai {
12     protected double bonus; // Besar bonus
13     // Getter bonus
14     public double getBonus() {
15         return bonus;
16     }
17     // Setter bonus
18     public void setBonus(double bonus) {
19         this.bonus = bonus;
20     }
21     // Implementasi status
22     @Override
23     String getStatus() {
24         return "Penasaran";
25     }
26     // Implementasi tunjangan
27     @Override
28     double getTunjangan() {
29         return bonus;
30     }
31     // Implementasi total gaji
32     @Override
33     double hitungGaji() {
34         return getGajip() + bonus;
35     }
36     // Implementasi email: nip@marketing.usd.ac.id
37     @Override
38     public String getEmail() {
39         public String getEmail() {
40             return getNip() + "marketing.usd.ac.id";
41         }
42     }
}
```

```
6  /*
7  * @author Calvin Pati
8  */
9
10 interface Email {
11     String getEmail(); // inisialisasi metode untuk email
12 }
13
14 
```

```
5      * Author Celvin Pati
6
7  Class Manager extends Pegawai {
8      protected double tunjanganJabatan; // Tunjangan jabatan
9      protected double lembur; // Lembur
10     // Getter tunjangan jabatan
11     public double getTunj Jab() {
12         return tunjanganJabatan;
13     }
14     // Setter tunjangan jabatan
15     public void setTunj Jab(double tunjanganJabatan) {
16         this.tunjanganJabatan = tunjanganJabatan;
17     }
18     // Getter lembur
19     public double getLembur() {
20         return lembur;
21     }
22     // Setter lembur
23     public void setLembur(double lembur) {
24         this.lembur = lembur;
25     }
26     // Implementasi status
27     @Override
28     String getstatus() {
29         return "Manajer";
30     }
31     // Implementasi tunjangan = tunjanganJabatan + lembur
32     @Override
33     double getTunjangan() {
34         return tunjanganJabatan + lembur;
35     }
36     // Implementasi total gaji
37     @Override
38     double hitungGaji() {
39         return getGapok() + tunjanganJabatan + lembur;
40     }
41     // Implementasi email: nifpmanajer.usd.ac.id
42     @Override
43     public String getEmail() {
44         return getNpp() + "@manajer.usd.ac.id";
45     }
46     // Implementasi nama
47     @Override
48     public String getName() {
49         return "Manager";
50     }
51 }
```

## C. OUTPUT

```

--- resources:3.3.1:resources (default-resources) @ Modul10PBO ---
skip non existing resourceDirectory C:\Users\Windows\OneDrive\Documents\NetBeansProjects\Modul10PBO\src\main\resources

--- compiler:3.13.0:compile (default-compile) @ Modul10PBO ---
Nothing to compile - all classes are up to date.

--- exec:3.1.0:exec (default-cli) @ Modul10PBO ---
Masukkan jumlah pegawai: 2

Pilih jenis pegawai:
1. Manajer
2. Pemasaran
3. Honorer
Pilihan: 1
NPF : 2200
Nama : cia
Gaji Pokok : 2000000
Tunj. Jabatan : 500
Lembur : 400

Pilih jenis pegawai:
1. Manajer
2. Pemasaran
3. Honorer
Pilihan: 2
NPF : 300
Nama : delta
Gaji Pokok : 1000000
Bonus : 300

NO NPF NAMA STATUS GAIK TULJANGAN GAJI TOTAL EMAIL
1. 2200 cia Manajer 2000000,00 500,00 2000900,00 2200@manajer.usd.ac.id
2. 300 delta Pemasaran 1000000,00 300,00 1000300,00 300@marketing.usd.ac.id

BUILD SUCCESS
-----
Total time: 01:07 min
Finished at: 2025-05-24T16:37:58+07:00
-----
```

## D. ANALISA

- Interface Email

No	Syntax	penjelasan
1	interface Email { String getEmail(); }	//Interface Email hanya mendefinisikan satu method getEmail() yang wajib diimplementasikan oleh setiap kelas yang mengimplementasikannya. Tujuannya adalah untuk memastikan bahwa setiap objek pegawai memiliki alamat email sesuai format masing-masing.

- **Class Abstrak Pegawai**

No	Syntax	penjelasan
1	abstract class Pegawai implements Email { ... }	// Mendefinisikan kelas abstrak Pegawai yang mewarisi interface Email. Tidak dapat diinstansiasi secara langsung.
2	protected int npp; protected String nama; protected double gajiPokok;	//Atribut dasar dari setiap pegawai: nomor pegawai, nama, dan gaji pokok.
3	public int getNpp() { return npp; }	//Getter untuk mengambil nilai atribut npp.
4	public void setNpp(int npp) { this.npp = npp; }	//Setter untuk menetapkan nilai npp.
5	abstract String getStatus();	//Method abstrak untuk mendapatkan status jabatan, wajib diimplementasikan oleh subclass.
6	abstract double getTunjangan();	//Method abstrak untuk mendapatkan tunjangan total, wajib diimplementasikan oleh subclass.
7	abstract double hitungGatot();	//Method abstrak untuk menghitung total gaji (gaji pokok + tunjangan).
8	public abstract String getEmail();	//Method abstrak dari interface Email, mengembalikan alamat email pegawai.

- **Class Honorer**

No	Syntax	penjelasan
1	class Honorer extends Pegawai { ... }	// Kelas Honorer adalah subclass dari Pegawai.
2	protected double lemburan;	//Menyimpan nilai lembur untuk pegawai honorer.

3	public double getLembur() { return lemburan; }	//Getter untuk mengambil jumlah lemburan.
4	public void setLembur(double lemburan) { this.lemburan = lemburan; }	//Setter untuk menetapkan jumlah lemburan.
5	String getStatus() { return "Honorer"; }	//Implementasi method abstrak, mengembalikan status pegawai "Honorer".
6	double getTunjangan() { return lemburan; }	//Implementasi method abstrak, tunjangan hanya dari lemburan.
7	double hitungGaji() { return getGapok() + lemburan; }	//Menghitung total gaji = gaji pokok + lembur.
8	public String getEmail() { return getNpp() + "@honorer.usd.ac.id"; }	//Format email khusus untuk pegawai honorer.

- **Kelas Pemasaran**

No	Syntax	penjelasan
1	class Pemasaran extends Pegawai { ... }	// Subclass dari Pegawai khusus untuk divisi pemasaran.
2	protected double bonus;	//Menyimpan nilai bonus pemasaran.
3	public double getBonus() { return bonus; }	//Getter untuk mengambil nilai bonus.
4	public void setBonus(double bonus) { this.bonus = bonus; }	//Setter untuk menetapkan nilai bonus.
5	String getStatus() { return "Pemasaran"; }	//Menyatakan status pegawai sebagai "Pemasaran".
6	double getTunjangan() { return bonus; }	//Tunjangan hanya berasal dari bonus.

7	double hitungGatot() { return getGapok() + bonus; }	//Total gaji = gaji pokok + bonus.
8	public String getEmail() { return getNpp() + "@marketing.usd.ac.id"; }	//Email khusus untuk pegawai pemasaran.

- Kelas Manajer

No	Syntax	penjelasan
1	class Manajer extends Pegawai { ... }	// Subclass dari Pegawai khusus untuk posisi manajerial.
2	protected double tunjanganJabatan; protected double lemburan;	//Menyimpan tunjangan jabatan dan lembur.
3	public double getTunJab() { return tunjanganJabatan; }	//Getter untuk tunjangan jabatan.
4	public void setTunJab(double tunjanganJabatan) { this.tunjanganJabatan = tunjanganJabatan; }	//Setter tunjangan jabatan.
5	public double getLembur() { return lemburan; }	//Getter lembur.
6	public void setLembur(double lemburan) { this.lemburan = lemburan; }	//Setter lembur.
7	String getStatus() { return "Manajer"; }	//Status sebagai "Manajer".
8	double getTunjangan() { return tunjanganJabatan + lemburan; }	//Tunjangan berasal dari jabatan + lembur.
9	double hitungGatot() { return getGapok() + tunjanganJabatan + lemburan; }	//Total gaji manajer.
10	public String getEmail() { return getNpp() + "@manajer.usd.ac.id"; }	//Email khusus untuk manajer.

➤ **Penjelasan alur jalannya program, penggunaan, dan manfaat interface dalam program:**

1) Inisialisasi Program

Program dimulai dari method main di dalam kelas Main. Di sini pengguna diminta menginput jumlah pegawai yang akan didata.

2) Perulangan Input Data Pegawai

Program melakukan loop sebanyak jumlah pegawai yang dimasukkan.

Dalam setiap iterasi, pengguna akan memilih jenis pegawai:

- Manajer
- Pemasaran
- Honorer

3) Input dan Penyimpanan Data Pegawai

Sesuai pilihan:

- Objek pegawai dari kelas yang sesuai (Manajer, Pemasaran, atau Honorer) dibuat.
- Data seperti npp, nama, gaji pokok, dan tunjangan khusus diinputkan oleh pengguna.
- Objek tersebut disimpan ke dalam array Pegawai[].

4) Output Hasil Data Pegawai

Setelah semua data dimasukkan, program akan mencetak tabel yang berisi:

- Nomor pegawai
- NPP
- Nama
- Status (jabatan)
- Gaji pokok
- Tunjangan
- Total gaji
- Email

## ➤ Penggunaan interface dalam program

### 1) Interface Email

```
interface Email {  
    String getEmail();  
}
```

- Digunakan sebagai kontrak bahwa semua kelas pegawai harus memiliki method getEmail().
- Diimplementasikan oleh kelas Pegawai, dan diteruskan ke semua turunannya (Manajer, Pemasaran, Honorer).
- Setiap kelas memberikan format email yang berbeda.

## ➤ Manfaat interface dalam program

Manfaat	Penjelasan
Konsistensi	Memastikan bahwa semua jenis pegawai memiliki method getEmail() meskipun formatnya berbeda.
Polimorfisme	Memungkinkan pemanggilan getEmail() tanpa peduli tipe objeknya (Manajer, Honorer, dll.) selama objek tersebut bertipe Email.
Fleksibilitas	Jika di masa depan ada jenis pegawai baru (misalnya Freelancer), cukup tambahkan kelas yang implement Email. Tidak perlu ubah kode lama.
Pemisahan Tanggung Jawab	Memisahkan logika utama data pegawai (NPP, nama, gaji, dll) dari tanggung jawab membuat email.
Keterbacaan & Maintainability	Lebih mudah dibaca dan dipelihara karena struktur program modular dan memiliki kontrak yang jelas.

## ➤ Perbandingan

Aspek	Program Tanpa Interface	Program Dengan Interface (Email)
Struktur Kode	Semua method didefinisikan langsung di kelas turunan atau abstrak.	Menggunakan interface Email untuk mendefinisikan kontrak getEmail().
Abstraksi Email	Method getEmail() tetap ada, tapi tidak dijamin dimiliki semua kelas.	Semua kelas pegawai <b>wajib</b> memiliki getEmail() karena terikat kontrak interface.
Konsistensi Method	Bisa saja kelas baru lupa menambahkan method getEmail().	Interface memastikan semua kelas pegawai punya method getEmail().
Polimorfisme	Tidak bisa menggunakan instanceof Email atau casting ke interface.	Bisa memanfaatkan polymorphism, misal: (Email) obj.getEmail()
Prinsip OOP (SOLID)	Kurang mendukung prinsip Interface Segregation & Dependency Inversion.	Lebih dekat dengan prinsip-prinsip OOP, terutama Interface Segregation.