

Nama : Fiza Rahmatus Sholikha
Kelas : SIB 2E
No. Absen : 07

Praktikum 1. Basic OOP

Langkah	Keterangan
1	<p>Kelas adalah blueprint atau cetak biru yang mendefinisikan struktur dan perilaku suatu objek. Kelas berisi atribut (data) dan metode (fungsi) yang berkaitan dengan objek tersebut. Objek, di sisi lain, adalah instance konkret dari suatu kelas, memiliki nilai nyata untuk atribut dan mampu menjalankan metode yang didefinisikan dalam kelas. Dalam PHP, Anda dapat membuat kelas dengan kata kunci class dan kemudian membuat objek dari kelas tersebut dengan kata kunci new. Berikut adalah contoh sederhana:</p>
2	Buatlah folder <code>dasarWeb/JS12_OOP</code> dan file baru didalamnya, dengan nama <code>oop.php</code> .
3	Ketikkan ke dalam file <code>oop.php</code> tersebut kode di bawah ini.
4	<pre> 1 <?php 2 2 references 0 implementations Windsurf: Refactor Explain 3 class Car{ 4 3 references 5 public \$brand; 6 7 1 reference 0 overrides Windsurf: Refactor Explain 8 public function startEngine(): void{ 9 echo "Engine started"; 10 } 11 12 13 \$car1 = new Car(); 14 \$car1->brand = "Toyota"; 15 16 \$car2 = new Car(); 17 \$car2->brand = "Honda"; 18 19 \$car1->startEngine(); 20 echo \$car2->brand; 21 ?> 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 159 160 161 162 163 164 165 166 167 168 169 169 170 171 172 173 174 175 176 177 178 179 179 180 181 182 183 184 185 186 187 187 188 189 189 190 191 192 193 194 195 196 197 197 198 199 199 200 201 202 203 204 205 206 207 207 208 209 209 210 211 212 213 214 215 215 216 217 217 218 218 219 219 220 220 221 221 222 222 223 223 224 224 225 225 226 226 227 227 228 228 229 229 230 230 231 231 232 232 233 233 234 234 235 235 236 236 237 237 238 238 239 239 240 240 241 241 242 242 243 243 244 244 245 245 246 246 247 247 248 248 249 249 250 250 251 251 252 252 253 253 254 254 255 255 256 256 257 257 258 258 259 259 260 260 261 261 262 262 263 263 264 264 265 265 266 266 267 267 268 268 269 269 270 270 271 271 272 272 273 273 274 274 275 275 276 276 277 277 278 278 279 279 280 280 281 281 282 282 283 283 284 284 285 285 286 286 287 287 288 288 289 289 290 290 291 291 292 292 293 293 294 294 295 295 296 296 297 297 298 298 299 299 300 300 301 301 302 302 303 303 304 304 305 305 306 306 307 307 308 308 309 309 310 310 311 311 312 312 313 313 314 314 315 315 316 316 317 317 318 318 319 319 320 320 321 321 322 322 323 323 324 324 325 325 326 326 327 327 328 328 329 329 330 330 331 331 332 332 333 333 334 334 335 335 336 336 337 337 338 338 339 339 340 340 341 341 342 342 343 343 344 344 345 345 346 346 347 347 348 348 349 349 350 350 351 351 352 352 353 353 354 354 355 355 356 356 357 357 358 358 359 359 360 360 361 361 362 362 363 363 364 364 365 365 366 366 367 367 368 368 369 369 370 370 371 371 372 372 373 373 374 374 375 375 376 376 377 377 378 378 379 379 380 380 381 381 382 382 383 383 384 384 385 385 386 386 387 387 388 388 389 389 390 390 391 391 392 392 393 393 394 394 395 395 396 396 397 397 398 398 399 399 400 400 401 401 402 402 403 403 404 404 405 405 406 406 407 407 408 408 409 409 410 410 411 411 412 412 413 413 414 414 415 415 416 416 417 417 418 418 419 419 420 420 421 421 422 422 423 423 424 424 425 425 426 426 427 427 428 428 429 429 430 430 431 431 432 432 433 433 434 434 435 435 436 436 437 437 438 438 439 439 440 440 441 441 442 442 443 443 444 444 445 445 446 446 447 447 448 448 449 449 450 450 451 451 452 452 453 453 454 454 455 455 456 456 457 457 458 458 459 459 460 460 461 461 462 462 463 463 464 464 465 465 466 466 467 467 468 468 469 469 470 470 471 471 472 472 473 473 474 474 475 475 476 476 477 477 478 478 479 479 480 480 481 481 482 482 483 483 484 484 485 485 486 486 487 487 488 488 489 489 490 490 491 491 492 492 493 493 494 494 495 495 496 496 497 497 498 498 499 499 500 500 501 501 502 502 503 503 504 504 505 505 506 506 507 507 508 508 509 509 510 510 511 511 512 512 513 513 514 514 515 515 516 516 517 517 518 518 519 519 520 520 521 521 522 522 523 523 524 524 525 525 526 526 527 527 528 528 529 529 530 530 531 531 532 532 533 533 534 534 535 535 536 536 537 537 538 538 539 539 540 540 541 541 542 542 543 543 544 544 545 545 546 546 547 547 548 548 549 549 550 550 551 551 552 552 553 553 554 554 555 555 556 556 557 557 558 558 559 559 560 560 561 561 562 562 563 563 564 564 565 565 566 566 567 567 568 568 569 569 570 570 571 571 572 572 573 573 574 574 575 575 576 576 577 577 578 578 579 579 580 580 581 581 582 582 583 583 584 584 585 585 586 586 587 587 588 588 589 589 590 590 591 591 592 592 593 593 594 594 595 595 596 596 597 597 598 598 599 599 600 600 601 601 602 602 603 603 604 604 605 605 606 606 607 607 608 608 609 609 610 610 611 611 612 612 613 613 614 614 615 615 616 616 617 617 618 618 619 619 620 620 621 621 622 622 623 623 624 624 625 625 626 626 627 627 628 628 629 629 630 630 631 631 632 632 633 633 634 634 635 635 636 636 637 637 638 638 639 639 640 640 641 641 642 642 643 643 644 644 645 645 646 646 647 647 648 648 649 649 650 650 651 651 652 652 653 653 654 654 655 655 656 656 657 657 658 658 659 659 660 660 661 661 662 662 663 663 664 664 665 665 666 666 667 667 668 668 669 669 670 670 671 671 672 672 673 673 674 674 675 675 676 676 677 677 678 678 679 679 680 680 681 681 682 682 683 683 684 684 685 685 686 686 687 687 688 688 689 689 690 690 691 691 692 692 693 693 694 694 695 695 696 696 697 697 698 698 699 699 700 700 701 701 702 702 703 703 704 704 705 705 706 706 707 707 708 708 709 709 710 710 711 711 712 712 713 713 714 714 715 715 716 716 717 717 718 718 719 719 720 720 721 721 722 722 723 723 724 724 725 725 726 726 727 727 728 728 729 729 730 730 731 731 732 732 733 733 734 734 735 735 736 736 737 737 738 738 739 739 740 740 741 741 742 742 743 743 744 744 745 745 746 746 747 747 748 748 749 749 750 750 751 751 752 752 753 753 754 754 755 755 756 756 757 757 758 758 759 759 760 760 761 761 762 762 763 763 764 764 765 765 766 766 767 767 768 768 769 769 770 770 771 771 772 772 773 773 774 774 775 775 776 776 777 777 778 778 779 779 780 780 781 781 782 782 783 783 784 784 785 785 786 786 787 787 788 788 789 789 790 790 791 791 792 792 793 793 794 794 795 795 796 796 797 797 798 798 799 799 800 800 801 801 802 802 803 803 804 804 805 805 806 806 807 807 808 808 809 809 810 810 811 811 812 812 813 813 814 814 815 815 816 816 817 817 818 818 819 819 820 820 821 821 822 822 823 823 824 824 825 825 826 826 827 827 828 828 829 829 830 830 831 831 832 832 833 833 834 834 835 835 836 836 837 837 838 838 839 839 840 840 841 841 842 842 843 843 844 844 845 845 846 846 847 847 848 848 849 849 850 850 851 851 852 852 853 853 854 854 855 855 856 856 857 857 858 858 859 859 860 860 861 861 862 862 863 863 864 864 865 865 866 866 867 867 868 868 869 869 870 870 871 871 872 872 873 873 874 874 875 875 876 876 877 877 878 878 879 879 880 880 881 881 882 882 883 883 884 884 885 885 886 886 887 887 888 888 889 889 890 890 891 891 892 892 893 893 894 894 895 895 896 896 897 897 898 898 899 899 900 900 901 901 902 902 903 903 904 904 905 905 906 906 907 907 908 908 909 909 910 910 911 911 912 912 913 913 914 914 915 915 916 916 917 917 918 918 919 919 920 920 921 921 922 922 923 923 924 924 925 925 926 926 927 927 928 928 929 929 930 930 931 931 932 932 933 933 934 934 935 935 936 936 937 937 938 938 939 939 940 940 941 941 942 942 943 943 944 944 945 945 946 946 947 947 948 948 949 949 950 950 951 951 952 952 953 953 954 954 955 955 956 956 957 957 958 958 959 959 960 960 961 961 962 962 963 963 964 964 965 965 966 966 967 967 968 968 969 969 970 970 971 971 972 972 973 973 974 974 975 975 976 976 977 977 978 978 979 979 980 980 981 981 982 982 983 983 984 984 985 985 986 986 987 987 988 988 989 989 990 990 991 991 992 992 993 993 994 994 995 995 996 996 997 997 998 998 999 999 1000 1000 1001 1001 1002 1002 1003 1003 1004 1004 1005 1005 1006 1006 1007 1007 1008 1008 1009 1009 1010 1010 1011 1011 1012 1012 1013 1013 1014 1014 1015 1015 1016 1016 1017 1017 1018 1018 1019 1019 1020 1020 1021 1021 1022 1022 1023 1023 1024 1024 1025 1025 1026 1026 1027 1027 1028 1028 1029 1029 1030 1030 1031 1031 1032 1032 1033 1033 1034 1034 1035 1035 1036 1036 1037 1037 1038 1038 1039 1039 1040 1040 1041 1041 1042 1042 1043 1043 1044 1044 1045 1045 1046 1046 1047 1047 1048 1048 1049 1049 1050 1050 1051 1051 1052 1052 1053 1053 1054 1054 1055 1055 1056 1056 1057 1057 1058 1058 1059 1059 1060 1060 1061 1061 1062 1062 1063 1063 1064 1064 1065 1065 1066 1066 1067 1067 1068 1068 1069 1069 1070 1070 1071 1071 1072 1072 1073 1073 1074 1074 1075 1075 1076 1076 1077 1077 1078 1078 1079 1079 1080 1080 1081 1081 1082 1082 1083 1083 1084 1084 1085 1085 1086 1086 1087 1087 1088 1088 1089 1089 1090 1090 1091 1091 1092 1092 1093 1093 1094 1094 1095 1095 1096 1096 1097 1097 1098 1098 1099 1099 1100 1100 1101 1101 1102 1102 1103 1103 1104 1104 1105 1105 1106 1106 1107 1107 1108 1108 1109 1109 1110 1110 1111 1111 1112 1112 1113 1113 1114 1114 1115 1115 1116 1116 1117 1117 1118 1118 1119 1119 1120 1120 1121 1121 1122 1122 1123 1123 1124 1124 1125 1125 1126 1126 1127 1127 1128 1128 1129 1129 1130 1130 1131 1131 1132 1132 1133 1133 1134 1134 1135 1135 1136 1136 1137 1137 1138 1138 1139 1139 1140 1140 1141 1141 1142 1142 1143 1143 1144 1144 1145 1145 1146 1146 1147 1147 1148 1148 1149 1149 1150 1150 1151 1151 1152 1152 1153 1153 1154 1154 1155 1155 1156 1156 1157 1157 1158 1158 1159 1159 1160 1160 1161 1161 1162 1162 1163 1163 1164 1164 1165 1165 1166 1166 1167 1167 1168 1168 1169 1169 1170 1170 1171 1171 1172 1172 1173 1173 1174 1174 1175 1175 1176 1176 1177 1177 1178 1178 1179 1179 1180 1180 1181 1181 1182 1182 1183 1183 1184 1184 1185 1185 1186 1186 1187 1187 1188 1188 1189 1189 1190 1190 1191 1191 1192 1192 1193 1193 1194 1194 1195 1195 1196 1196 1197 1197 1198 1198 1199 1199 1200 1200 1201 1201 1202 1202 1203 1203 1204 1204 1205 1205 1206 1206 1207 1207 1208 1208 1209 1209 1210 1210 1211 1211 1212 1212 1213 1213 1214 1214 1215 1215 1216 1216 1217 1217 1218 1218 1219 1219 1220 1220 1221 1221 1222 1222 1223 1223 1224 1224 1225 1225 1226 1226 1227 1227 1228 1228 1229 1229 1230 1230 1231 1231 1232 1232 1233 1233 1234 1234 1235 1235 1236 1236 1237 1237 1238 1238 1239 1239 1240 1240 1241 1241 1242 1242 1243 1243 1244 1244 1245 1245 1246 1246 1247 1247 1248 1248 1249 1249 1250 1250 1251 1251 1252 1252 1253 1253 1254 1254 1255 1255 1256 1256 1257 1257 1258 1258 1259 1259 1260 1260 1261 1261 1262 1262 1263 1263 1264 1264 1265 1265 1266 1266 1267 1267 1268 1268 1269 1269 1270 1270 1271 1271 1272 1272 1273 1273 1274 1274 1275 1275 1276 1276 1277 1277 1278 1278 1279 1279 1280 1280 1281 1281 1282 1282 1283 1283 1284 1284 1285 1285 1286 1286 1287 1287 1288 1288 1289 1289 1290 1290 1291 1291 1292 1292 1293 1293 1294 1294 1295 1295 1296 1296 1297 1297 1298 1298 1299 1299 1300 1300 1301 1301 1302 1302 1303 1303 1304 1304 1305 1305 1306 1306 1307 1307 1308 1308 1309 1309 1310 1310 1311 1311 1312 1312 1313 1313 1314 1314 1315 1315 1316 1316 1317 1317 1318 1318 1319 1319 1320 1320 1321 1321 1322 1322 1323 1323 1324 1324 1325 1325 1326 1326 1327 1327 1328 1328 1329 1329 1330 1330 1331 1331 1332 1332 1333 1333 1334 1334 1335 1335 1336 1336 1337 1</pre>

```

1  <?php
2   references | 2 implementations | Windsurf: Refactor | Explain
3   class Animal{
4     5 references
5     protected $name;
6     0 references | 0 overrides | Windsurf: Refactor | Explain | Generate F...
7     public function __construct($name){
8       $this->name = $name;
9     }
10
11    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate F...
12    public function eat(): void{
13      echo $this->name . " is eating.<br>";
14    }
15
16    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate F...
17    class Cat extends Animal{
18      1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate F...
19      public function meow(): void{
20        echo $this->name . " says meow!<br>";
21      }
22
23    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate F...
24    class Dog extends Animal{
25      1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate F...
26      public function bark(): void{
27        echo $this->name . " says woof!<br>";
28      }
29
30    $cat = new Cat(name: "Whiskers");
31    $dog = new Dog(name: "Buddy");
32
33    $cat->eat();
34    $dog->sleep();
35
36    $cat->meow();
37    $dog->bark();

```

localhost/dasarWeb/JS12_OOP/inheritance.php

Whiskers is eating.
Buddy is sleeping.
Whiskers says meow!
Buddy says woof!

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. ([soal no 1.2](#))
 Pada kode ini terdapat class Animal yang merupakan parent class dengan atribut protected (\$name) dan method (__construct, eat, sleep). Class Cat dan Dog adalah child class (anak) yang extends (mewarisi) semua properti dan method dari Animal. Hasilnya, objek \$cat dan \$dog bisa menggunakan method dari Animal (seperti eat() dan sleep()) sekaligus method spesifik mereka sendiri (meow() dan bark())

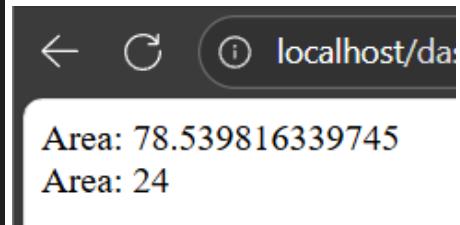
7

[Polymorphism](#) adalah konsep dalam pemrograman berorientasi objek yang memungkinkan objek dari class yang berbeda untuk merespon pada pemanggilan metode dengan cara yang sama. Ini dapat diwujudkan dalam PHP melalui penggunaan antarmuka (interface) dan penggunaan overriding metode. Dengan polymorphism, Anda dapat memperlakukan objek dari class yang berbeda dengan cara yang seragam.

Berikut adalah contoh sederhana penggunaan polymorphism dalam PHP menggunakan antarmuka:

```

1 <?php
2 interface Shape{
3     public function calculateArea();
4 }
5
6 class Circle implements Shape{
7     private $radius;
8
9     public function __construct($radius){
10         $this->radius = $radius;
11     }
12
13     public function calculateArea(){
14         return pi() * pow($this->radius, 2);
15     }
16 }
17
18 class Rectangle implements Shape{
19     private $width;
20     private $height;
21
22     public function __construct($width, $height){
23         $this->width = $width;
24         $this->height = $height;
25     }
26
27     public function calculateArea(){
28         return $this->width * $this->height;
29     }
30 }
31
32 function printArea(Shape $shape){
33     echo "Area: " . $shape->calculateArea() . "<br>";
34 }
35
36 $circle = new Circle(5);
37 $rectangle = new Rectangle(4, 6);
38
39 printArea($circle);
40 printArea($rectangle);
41 ?>
```



Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. ([soal no 1.3](#))
 Pada kode ini memiliki interface Shape menetapkan "kontrak" bahwa setiap class yang mengimplementasikannya harus memiliki method calculateArea(). Class Circle dan Rectangle sama-sama mengimplementasikan Shape, tetapi dengan implementasi calculateArea() yang berbeda. Fungsi printArea() dapat menerima objek apapun yang bertipe Shape (baik Circle maupun Rectangle) dan memanggil method calculateArea()

8

Encapsulation adalah salah satu konsep dalam pemrograman berorientasi objek (OOP) yang mengizinkan pembungkusan (encapsulation) properti dan metode dalam sebuah class sehingga akses ke mereka dapat dikontrol. Hal ini dapat membantu dalam menerapkan prinsip-prinsip pengelolaan akses dan memastikan bahwa properti dan metode yang mungkin berubah di kemudian hari tidak merusak integritas class atau program secara keseluruhan.

Berikut adalah contoh sederhana encapsulation dalam PHP:

```

1 <?php
3 references | 0 implementations | Windsurf: Refactor | Explain
2 class Car{
2 references
3 | private $model;
3 references
4 | private $color;
5
3 references | 0 overrides | Windsurf: Refactor | Explain | Generate Function
6 public function __construct($model, $color){
7     $this->model = $model;
8     $this->color = $color;
9 }
10
1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function
11 public function getModel(): mixed{
12     return $this->model;
13 }
14
1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function
15 public function setColor($color): void{
16     $this->color = $color;
17 }
18
2 references | 0 overrides | Windsurf: Refactor | Explain | Generate Function
19 public function getColor(): mixed{
20     return $this->color;
21 }
22 }
23
24 $car = new Car(model: "Toyota", color: "Blue");
25 echo "Model: " . $car->getModel() . "<br>";
26 echo "Color: " . $car->getColor() . "<br>";
27
28 $car->setColor(color: "Red");
29 echo "Updated Color: " . $car->getColor() . "<br>";
30 ?>

```

Model: Toyota
Color: Blue
Updated Color: Red

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. ([soal no 1.4](#))

Pada kode ini terdapat atribut \$model dan \$color pada class Car disetel sebagai private, artinya tidak bisa diakses langsung dari luar class. Untuk mengakses atau mengubah nilainya, method public (disebut getter dan setter) seperti getModel() dan setColor() harus digunakan. Dengan cara ini, data tidak bisa diubah langsung dari luar kelas (\$objek->nama tidak bisa diakses langsung), melainkan harus lewat metode yang sudah disediakan. Tujuannya adalah melindungi data, mencegah perubahan yang tidak diinginkan, dan menjaga agar struktur program tetap aman serta konsisten..

9

Abstraction adalah salah satu konsep dasar dalam pemrograman berorientasi objek (OOP) yang memungkinkan Anda menyembunyikan detail internal dan hanya mengekspos fungsionalitas yang diperlukan. Ini membantu dalam menciptakan class dan metode yang bersifat umum dan fleksibel, memungkinkan pengguna untuk berinteraksi dengan objek tanpa perlu mengetahui implementasi internalnya.

Berikut adalah contoh sederhana abstraksi dalam PHP menggunakan abstract class dan method:

```

1 <?php
2 abstract class Shape {
3     abstract public function calculateArea();
4 }
5
6 class Circle extends Shape {
7     private $radius;
8
9     public function __construct($radius){
10         $this->radius = $radius;
11     }
12
13     public function calculateArea(){
14         return pi() * pow($this->radius, 2);
15     }
16 }
17
18 class Rectangle extends Shape {
19     private $width;
20     private $height;
21
22     public function __construct($width, $height){
23         $this->width = $width;
24         $this->height = $height;
25     }
26
27     public function calculateArea(){
28         return $this->width * $this->height;
29     }
30 }
31
32 $circle = new Circle(5);
33 $rectangle = new Rectangle(4, 6);
34
35 echo "Area of Circle: " . $circle->calculateArea() . "<br>";
36 echo "Area of Rectangle: " . $rectangle->calculateArea() . "<br>";
37 ?>

```

← ⌂ ⓘ localhost/dasarWeb

Area of Circle: 78.539816339745
Area of Rectangle: 24

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.5)
Pada kode ini, class Shape didefinisikan sebagai abstract dan memiliki abstract method calculateArea(). Ini berarti class Shape tidak bisa dibuat objeknya secara langsung dan hanya berfungsi sebagai blueprint dasar. Class anak seperti Circle dan Rectangle yang extends Shape wajib menyediakan implementasi dari method calculateArea(). Dengan abstraksi, kita menyembunyikan detail implementasi dan hanya menampilkan kerangka umum agar program lebih mudah dikembangkan tanpa harus tahu bagaimana setiap fungsi bekerja di dalamnya.

10

Interface adalah konsep dalam pemrograman berorientasi objek yang memungkinkan definisi kontrak atau kerangka yang harus diikuti oleh class-class yang mengimplementasikannya. Interface tidak memiliki implementasi sendiri, tetapi hanya menyediakan deklarasi metode dan properti yang harus diimplementasikan oleh class yang menggunakaninya. Hal ini memungkinkan untuk mencapai polimorfisme tanpa memerlukan pewarisan tunggal, sehingga sebuah class dapat mengimplementasikan beberapa interface.

Berikut adalah contoh penggunaan interface dalam PHP:

```

1  <?php
2   6 references | 5 implementations | Windsurf: Refactor | Explain
3   interface Shape {
4     4 references | 5 overrides
5     public function calculateArea();
6   }
7
8   1 reference | 3 implementations | Windsurf: Refactor | Explain
9   interface Color {
10    1 reference | 1 override
11    public function getColor();
12  }
13
14  3 references | 0 implementations | Windsurf: Refactor | Explain
15  class Circle implements Shape, Color {
16    6 references
17    private $radius;
18    2 references
19    private $color;
20
21    3 references | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
22    public function __construct($radius, $color){
23      $this->radius = $radius;
24      $this->color = $color;
25    }
26
27    4 references | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
28    public function calculateArea(): float|int{
29      return pi() * pow(num: $this->radius, exponent: 2);
30    }
31
32    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
33    public function getColor(): mixed{
34      return $this->color;
35    }
36  }
37
38  $circle = new Circle(radius: 5, color: "Blue");
39  echo "Area of Circle: " . $circle->calculateArea() . "<br>";
40  echo "Color of Circle: " . $circle->getColor() . "<br>";
41 ?>

```

localhost/dasarWeb
Area of Circle: 78.539816339745
Color of Circle: Blue

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.6)
Pada kode ini, ada dua interface, Shape dengan method calculateArea dan Color dengan method getColor. Class Circle mengimplementasikan keduanya implements Shape, Color, sehingga class Circle diwajibkan untuk menyediakan implementasi untuk semua method dari kedua interface tersebut. Interface digunakan agar beberapa kelas berbeda dapat memiliki perilaku yang sama tanpa harus saling mewarisi.

Constructors dan **destructors** adalah metode khusus dalam pemrograman berorientasi objek (OOP) yang digunakan dalam PHP untuk menginisialisasi dan membersihkan objek. **Constructor** adalah metode yang dipanggil secara otomatis ketika objek baru dibuat, sedangkan **destructor** adalah metode yang dipanggil secara otomatis ketika objek dihapus atau tidak lagi digunakan.

Constructor (Metode Pembuat)

Constructor menggunakan nama khusus `__construct` dalam PHP. Constructor ini akan dipanggil secara otomatis setiap kali objek baru dibuat dari class yang mengandung constructor tersebut.

Destructor (Metode Penghancur)

Destructor menggunakan nama khusus `__destruct` dalam PHP. Destructor ini akan dipanggil secara otomatis ketika objek dihapus atau program selesai dieksekusi.

Berikut adalah contoh constructor dan destructor:

10

```

1 <?php
2 references | 0 implementations | Windsurf: Refactor | Explain
2 class Car{
3     2 references
4     private $brand;
5
6         2 references | 0 overrides | Windsurf: Refactor | Explain | Generate
7     public function __construct($brand){
8         $this->brand = $brand;
9     }
10
11     1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate
12     public function getBrand(): mixed{
13         return $this->brand;
14     }
15
16     0 references | 0 overrides | Windsurf: Refactor | Explain | Generate
17     public function __destruct(){
18         echo "The car is being destroyed.<br>";
19     }
20
21 }
22
23 $car = new Car(brand: "Toyota");
24 echo "Brand: " . $car->getBrand() . "<br>";
25 ?>
26
27

```

← ⏪ ⓘ localhost/dasarWeb/jS12_OOP/ConsDestructors.php

Brand: Toyota
The car is being destroyed.

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.7)
Pada kode ini, public function `__construct()` adalah constructor yang otomatis dipanggil saat sebuah objek baru (`new Car(...)`) dibuat, digunakan di sini untuk menginisialisasi properti `$brand`. Sebaliknya, public function `__destruct()` adalah destructor yang otomatis dipanggil saat objek akan dihapus atau di akhir eksekusi skrip, yang di sini digunakan untuk mencetak pesan "The car is being destroyed.".

Encapsulation and Access Modifiers

Encapsulation adalah salah satu konsep utama dalam pemrograman berorientasi objek (OOP), dan itu melibatkan pembungkusan data (variabel) dan metode (fungsi) dalam sebuah class. Ini membantu dalam menyembunyikan implementasi internal suatu class dan hanya mengekspos fungsionalitas yang diperlukan. Access modifiers adalah bagian dari encapsulation yang memungkinkan Anda mengontrol tingkat akses ke properti dan metode dalam sebuah class.

PHP memiliki tiga access modifiers utama yang dapat digunakan dalam class:

12

Public: Properti atau metode yang dideklarasikan sebagai public dapat diakses dari luar class, sehingga mereka bersifat terbuka untuk diakses dari mana saja.

Protected: Properti atau metode yang dideklarasikan sebagai protected hanya dapat diakses dari dalam class itu sendiri dan dari class turunannya (inheritance).

Private: Properti atau metode yang dideklarasikan sebagai private hanya dapat diakses dari dalam class itu sendiri. Mereka tidak dapat diakses dari luar class, bahkan oleh class turunannya.

Berikut adalah contoh penggunaan access modifiers dalam PHP:

12

```
1 <?php
2     3 references | 2 implementations | Windsurf: Refactor | Explain
3     class Animal {
4         7 references
5         public $name;
6         2 references
7         protected $age;
8         2 references
9         private $color;
10
11     1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment
12     public function __construct($name, $age, $color){
13         $this->name = $name;
14         $this->age = $age;
15         $this->color = $color;
16     }
17
18     1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment
19     public function getName(): mixed{
20         return $this->name;
21     }
22     1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comment
23     protected function getAge(): mixed{
24         return $this->age;
25     }
26     1 reference | Windsurf: Refactor | Explain | Generate Function Comment
27     private function getColor(): mixed{
28         return $this->color;
29     }
30 }
```



Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.8)
Pada kode ini, ethod public (getName) dapat diakses dari mana saja. Method protected (getAge) dan private (getColor) tidak dapat diakses dari luar class. Inilah mengapa kode menghasilkan Fatal Error saat mencoba memanggil getAge() dari global scope, membuktikan bahwa akses terproteksi tidak bisa dipanggil dari luar

Praktikum 2. CRUD dengan OOP

Langkah	Keterangan
1	Buat file baru pada dasarWeb/JS12_OOP/database.php . Ketikkan kode seperti di bawah ini.
2	<pre> 1 <?php 2 class Database{ 3 private \$host = "localhost"; 4 private \$username = "postgrees"; 5 private \$password = "12345678"; 6 private \$database = "prakwebdb"; 7 public \$conn; 8 9 public function connect(){ 10 \$conn = pg_connect("host=\$this->host dbname=\$this->database user=\$this->username password=\$this->password"); 11 return \$conn; 12 13 if (!\$conn) { 14 die("Koneksi gagal: " . pg_last_error()); 15 } 16 } 17 } 18 ?></pre>
3	Buat file baru pada dasarWeb/JS12_OOP/crud.php . Ketikkan kode seperti di bawah ini.
4	<pre> 1 <?php 2 require_once 'database.php'; 3 4 class Crud { 5 private \$db; 6 7 public function __construct() { 8 \$this->db = new Database(); 9 } 10 11 // Create 12 public function create(\$jabatan, \$keterangan) { 13 \$query = "INSERT INTO jabatan (jabatan, keterangan) VALUES ('\$jabatan', '\$keterangan')"; 14 \$result = pg_query(\$this->db->conn->query(\$query)); 15 return \$result; 16 } 17 18 // Read 19 public function read() { 20 \$query = "SELECT * FROM jabatan"; 21 \$result = pg_query(\$this->db->conn->query(\$query)); 22 23 \$data = []; 24 \$numRows = pg_num_rows(\$result); 25 if (\$numRows > 0) { 26 while (\$row = pg_fetch_assoc(\$result)) { 27 \$data[] = \$row; 28 } 29 } 30 return \$result; 31 } 32 33 // Read by ID 34 public function readById(\$id) { 35 \$query = "SELECT * FROM jabatan WHERE id = '\$id'"; 36 \$result = pg_query(\$this->db->conn->query(\$query)); 37 38 \$numRows = pg_num_rows(\$result); 39 if (\$numRows == 1) { 40 \$row = pg_fetch_assoc(\$result); 41 return \$row; 42 } else { 43 return null; 44 } 45 } 46 47 // Update 48 public function update(\$id, \$jabatan, \$keterangan) { 49 \$query = "UPDATE jabatan SET jabatan = '\$jabatan', keterangan = '\$keterangan' WHERE id = '\$id'"; 50 \$result = pg_query(\$this->db->conn->query(\$query)); 51 return \$result; 52 } 53 54 // Delete 55 public function delete(\$id) { 56 \$query = "DELETE FROM jabatan WHERE id = '\$id'"; 57 \$result = pg_query(\$this->db->conn->query(\$query)); 58 return \$result; 59 } 60 }</pre>

5	Buat file baru pada dasarWeb/JS12_OOP/index.php . Ketikkan kode seperti di bawah ini.
6	<pre> 1 <?php 2 require_once 'Crud.php'; 3 4 \$crud = new Crud(); 5 6 if (\$_SERVER['REQUEST_METHOD'] === 'POST') { 7 \$jabatan = \$_POST['jabatan']; 8 \$keterangan = \$_POST['keterangan']; 9 \$crud->create(\$jabatan, \$keterangan); 10 } 11 12 if (isset(\$_GET['action']) && \$_GET['action'] === 'delete') { 13 \$id = \$_GET['id']; 14 \$crud->delete(\$id); 15 } 16 17 \$tampil = \$crud->read(); 18 ?> 19 20 <!DOCTYPE html> 21 <html lang="en"> 22 <head> 23 <meta charset="UTF-8"> 24 <meta name="viewport" content="width=device-width, initial-scale=1.0"> 25 <title>CRUD Jabatan</title> 26 <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css"> 27 </head> 28 <body> 29 <div class="container mt-5"> 30 <button type="button" class="btn btn-success mb-3" data-toggle="modal" data-target="#tambahModal">Tambah</button> 31 <table class="table"> 32 <thead> 33 <tr> 34 <th>ID</th> 35 <th>Jabatan</th> 36 <th>Keterangan</th> 37 <th>Aksi</th> 38 </tr> 39 </thead> 40 <tbody> 41 <?php 42 foreach (\$tampil as \$show) { 43 echo "<tr>"; 44 echo "<td>{\$show['id']}</td>"; 45 echo "<td>{\$show['jabatan']}</td>"; 46 echo "<td>{\$show['keterangan']}</td>"; 47 echo "<td>"; 48 echo "Edit"; 49 echo "Delete"; 50 echo "</td>"; 51 echo "</tr>"; 52 } 53 > 54 </tbody> 55 </table> 56 </div> 57 58 <div class="modal fade" id="tambahModal" tabindex="-1" role="dialog" aria-labelledby="exampleModallLabel" aria-hidden="true"> 59 <div class="modal-dialog" role="document"> 60 <div class="modal-content"> 61 <div class="modal-header"> 62 <h5 class="modal-title" id="exampleModallLabel">Tambah Data Jabatan</h5> 63 <button type="button" class="close" data-dismiss="modal" aria-label="Close"> 64 &times; 65 </button> 66 </div> 67 <div class="modal-body"> 68 <form method="POST" action=""> 69 <div class="form-group"> 70 <label for="name">Jabatan:</label> 71 <input type="text" class="form-control" id="jabatan" name="jabatan" required> 72 </div> 73 <div class="form-group"> 74 <label for="email">Keterangan:</label> 75 <textarea name="keterangan" class="form-control" id="keterangan" cols="30" rows="10" required></textarea> 76 </div> 77 <button type="submit" class="btn btn-primary">Tambah</button> 78 </form> 79 </div> 80 </div> 81 </div> 82 </div> 83 84 <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script> 85 <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script> 86 </body> 87 </html> 88 </pre>
7	Buat file baru dasarWeb/JS12_OOP/edit.php . Ketikkan kode seperti di bawah ini.

```

1 <?php
2 require_once 'crud.php';
3
4 $crud = new Crud();
5
6 $id = $_GET['id'];
7
8 $stampil = $crud->readById($id);
9
10 if ($_SERVER['REQUEST_METHOD'] == "POST") {
11     $jabatan = $_POST['jabatan'];
12     $keterangan = $_POST['keterangan'];
13     $crud->update($id, $jabatan, $keterangan);
14
15     header("Location: index.php");
16     exit;
17 }
18 ?>
19
20 <!DOCTYPE html>
21 <html lang="en">
22 <head>
23     <meta charset="UTF-8">
24     <meta name="viewport" content="width=device-width, initial-scale=1.0">
25     <title>Edit Jabatan</title>
26     <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
27 </head>
28 <body>
29     <div class="container mt-5">
30         <h2>Edit Jabatan</h2>
31         <form method="post" action="">
32             <div class="form-group">
33                 <label for="jabatan">Jabatan:</label>
34                 <input type="text" class="form-control" id="jabatan" name="jabatan" value=<?php echo $stampil['jabatan']; ?> required>
35             </div>
36             <div class="form-group">
37                 <label for="keterangan">Keterangan:</label>
38                 <textarea name="keterangan" class="form-control" id="keterangan" cols="30" rows="10" required><?php echo $stampil['keterangan']; ?></textarea>
39             </div>
40             <input type="hidden" name="id" value=<?php echo $stampil['id']; ?>>
41             <button type="submit" class="btn btn-primary">Simpan</button>
42         </form>
43     </div>
44
45     <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
46     <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
47 </body>
48 </html>

```

8

Jalankan code pada praktikum 2. Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 2.1)

The figure consists of three screenshots of a web application demonstrating CRUD operations on job positions.

- Screenshot 1 (Top):** A table titled "Tambah" (Add) showing one record: ID 2, Jabatan HR, Keterangan merekrut pegawai baru. Action buttons: Edit (blue) and Delete (red).
- Screenshot 2 (Middle):** An "Edit Jabatan" form. The Jabatan field contains "web developer". The Keterangan field contains "membuat website". A "Simpan" (Save) button is at the bottom.
- Screenshot 3 (Bottom):** The same table as Screenshot 1, now showing the updated record: ID 2, Jabatan web developer, Keterangan membuat website. Action buttons: Edit (blue) and Delete (red).

ID	Jabatan	Keterangan	Aksi
2	web developer	membuat website	Edit Delete
5	Cyber Security Engineer	bertanggungjawab atas kemanan data	Edit Delete

Penjelasan:

- database.php berisi class Database yang tugasnya hanya mengelola koneksi ke database PostgreSQL
- crud.php berisi class Crud yang menangani semua logika operasi database (INSERT, SELECT, UPDATE, DELETE) sebagai method-nya. Class ini menggunakan class Database untuk mendapatkan koneksi
- index.php adalah halaman utama yang menampilkan data (Read), menangani penambahan data (Create), dan penghapusan data (Delete). Halaman ini menggunakan object dari class Crud untuk berinteraksi dengan database.
- edit.php adalah halaman terpisah untuk mengubah data berdasarkan ID yang dikirim, yang juga menggunakan method dari class Crud