**LAPORAN PRAKTIKUM**

**PERTEMUAN-6**

Diajukan untuk memenuhi salat satu tugas praktikum Mata kuliah Pemrograman Berorientasi Objek



**Disusun Oleh:**

**Daiva Raditya Pradipa (231511039)**

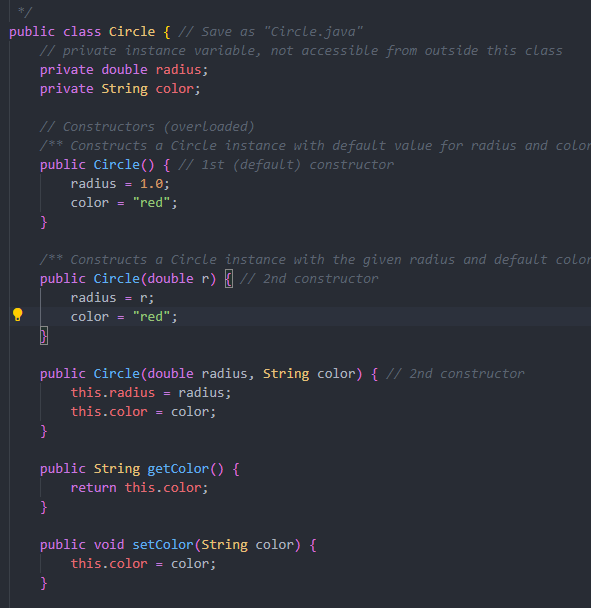
**Jurusan Teknik Komputer dan Informatika**

**Program Studi D-3 Teknik Informatika**

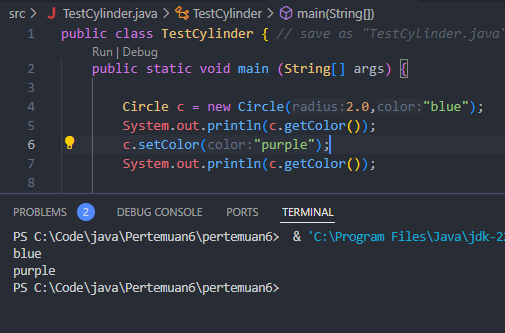
**Politeknik Negeri Bandung**

**2024**

1. Task 1.1



Hasil akhir program



Penjelasan:

Pada task 1.1, kita diminta untuk menambahkan atribut baru bernama color dengan tipe data String, yang berfungsi untuk menyimpan warna dari objek class SHape. Selain itu, kita diminta membuat constructor tambahan dengan parameter tambahan untuk color, sehingga saat membuat objek, kita bisa langsung menentukan warnanya. Kemudian, kita harus menambahkan metode getter dan setter untuk atribut color. Getter akan digunakan untuk mengambil nilai color, sedangkan setter berfungsi untuk mengubah atau menetapkan nilai baru pada atribut color

Masalah:-

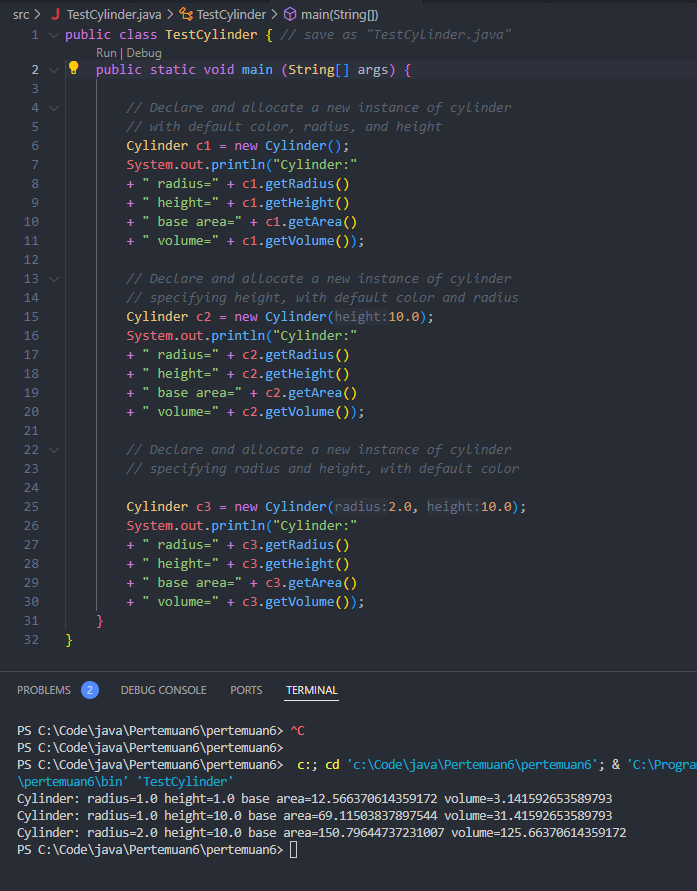
Solusi:-

Teman yang membantu: -

1. Task 1.2



Hasil akhir program



Penjelasan:

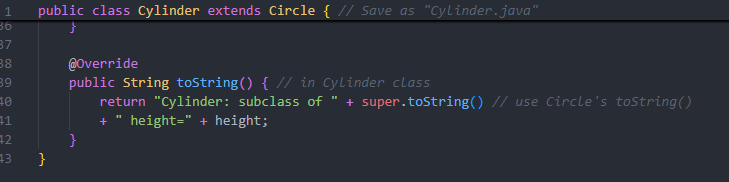
Pada task 1.2, kita diminta untuk mengubah implementasi dari method getArea yang ada pada superclass Circle di dalam subclass Cylinder. Dalam subclass Cylinder, rumus yang digunakan untuk menghitung luas permukaan (surface area) adalah Surface Area=2π×radius×height+2×base area. Untuk menerapkan perubahan ini, kita bisa menggunakan konsep method overriding, di mana kita menulis ulang method getArea di subclass Cylinder agar sesuai dengan rumus tersebut dengan terlebih dahulu menuliskan syntax @Override untuk mengindikasikan bahwa method tersebut berasal dari superclass yang akan didefinisikan ulah dan diubah implementasinya. Dengan demikian, saat objek Cylinder memanggil getArea, hasilnya akan dihitung berdasarkan rumus luas permukaan tabung, bukan luas lingkaran seperti di superclass Circle.

Masalah:-

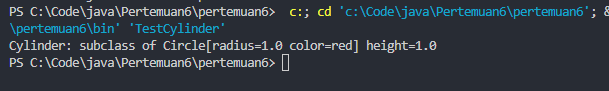
Solusi:-

Teman yang membantu: -

1. Task 1.3



Hasil akhir program



Penjelasan:

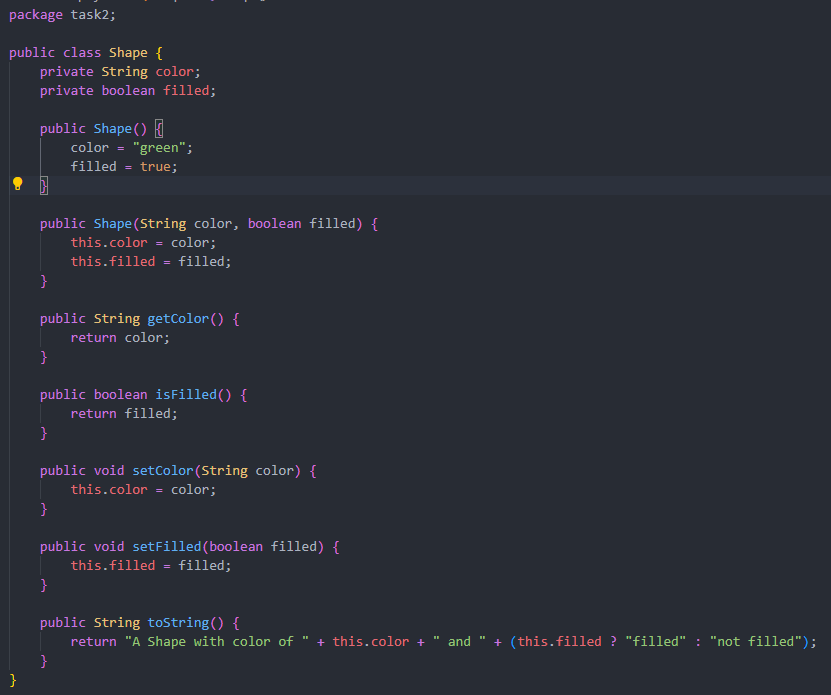
Pada task 1.3, kita diminta untuk mengubah implementasi method toString yang ada pada superclass Circle di dalam subclass Cylinder. Dalam hal ini, implementasi di subclass Cylinder harus memanggil method toString dari superclass Circle, dengan hasil akhir berupa teks seperti: "Cylinder: subclass of " + // menggunakan toString() dari Circle + " height=" + height. Untuk melakukan hal ini, kita dapat menggunakan syntax super.toString() untuk memanggil method toString dari superclass. Dengan demikian, method toString di subclass Cylinder akan menggabungkan informasi dari superclass Circle dan menambahkan detail tinggi (height) yang spesifik untuk tabung.

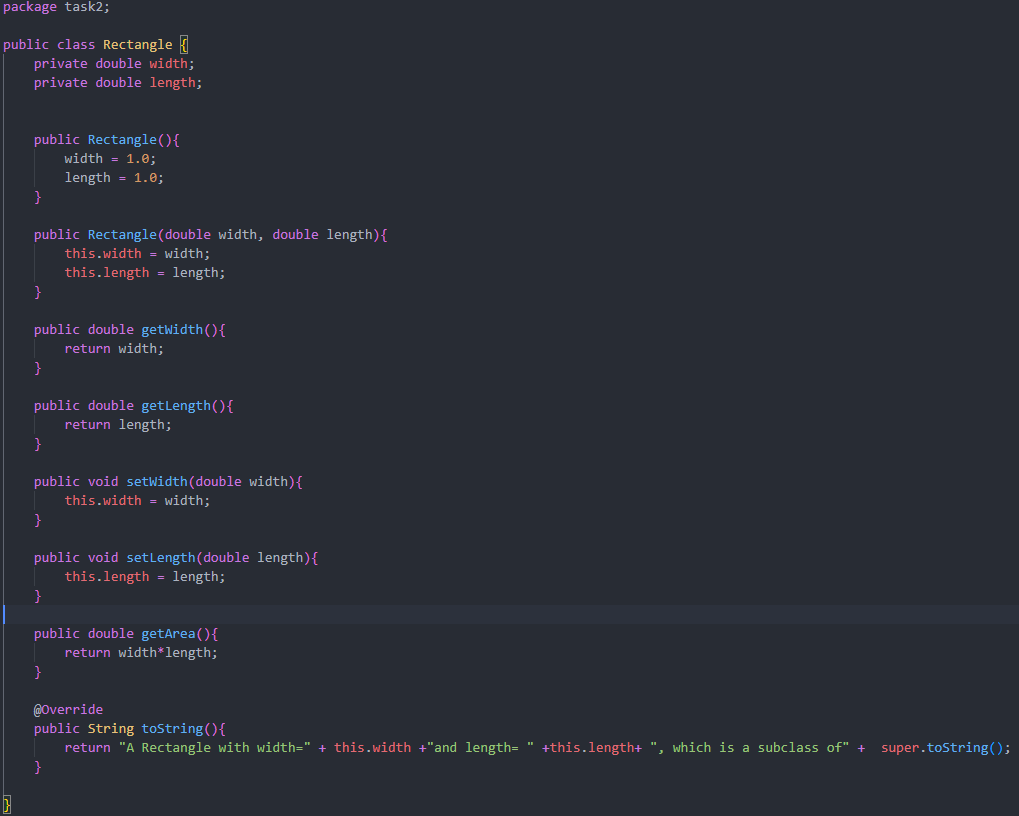
Masalah:-

Solusi:-

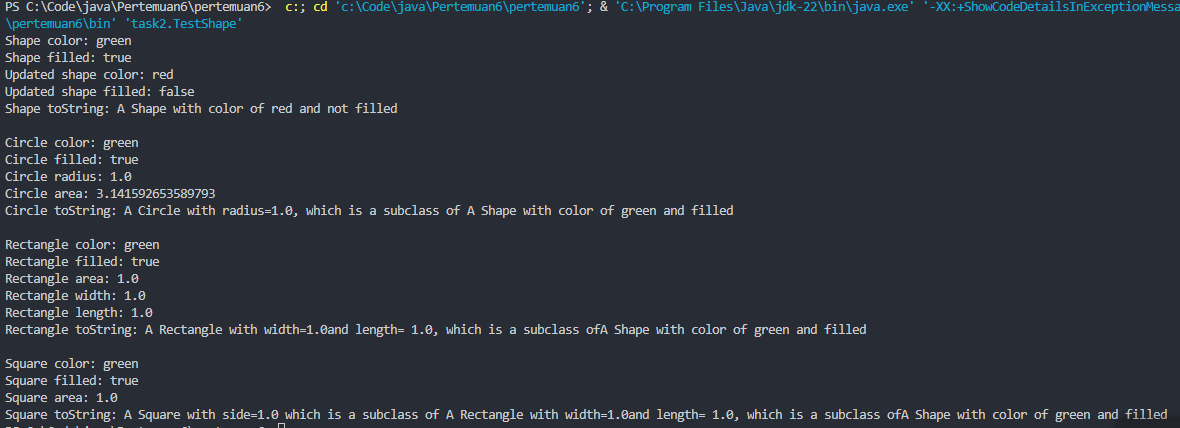
Teman yang membantu: -

1. Task 2.1





Hasil akhir program



Penjelasan:

Pada task 2.1, kita diminta untuk membuat empat kelas dengan ketentuan sebagai berikut:

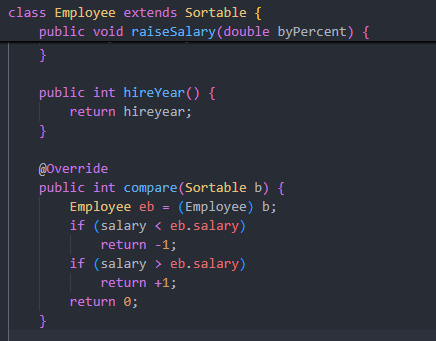
1. Class Shape berfungsi sebagai superclass.
2. Class Rectangle dan Circle merupakan subclass atau spesialisasi dari kelas Shape.
3. Class Square merupakan subclass dari kelas Rectangle.

Selain itu, pada masing-masing kelas, kita diminta untuk membuat constructor dengan nilai awal (initial value) yang berbeda-beda, tergantung pada jumlah parameter dalam constructor. Jika constructor tidak memiliki parameter, maka atribut class tersebut akan diisi dengan nilai default saat constructor dijalankan. Selain itu, kita juga perlu membuat method getter dan setter di setiap kelas untuk mengakses dan memodifikasi nilai atribut.

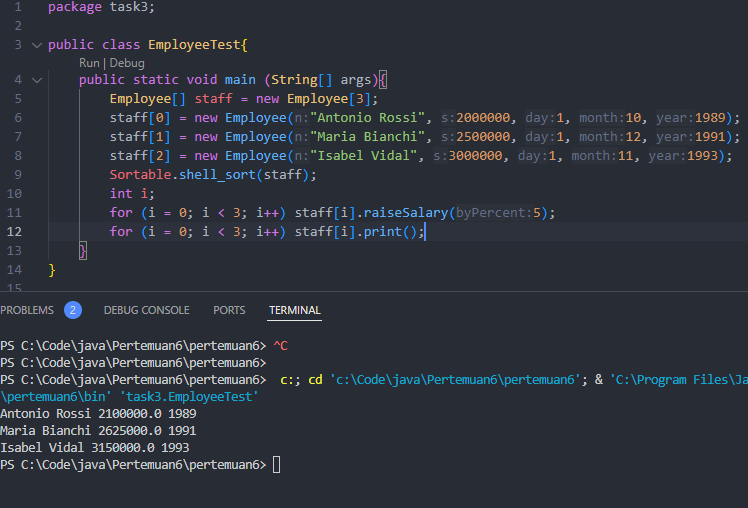
Dalam daftar class ini, Shape merupakan class yang paling umum (general), yang mewariskan atribut dan method ke setiap class di bawahnya. Class Shape berfungsi sebagai superclass yang mendefinisikan fitur umum yang dimiliki oleh semua bentuk (shape), seperti warna dan apakah bentuk tersebut terisi atau tidak. Setiap subclass di level bawah, seperti Rectangle, Circle, dan Square, akan mewarisi atribut dan method dari class Shape, serta memiliki atribut dan method spesifik yang unik bagi masing-masing bentuk. Hal ini juga berlaku untuk class Rectangle yang mewariskan atribut dan methodnya pada class Square.

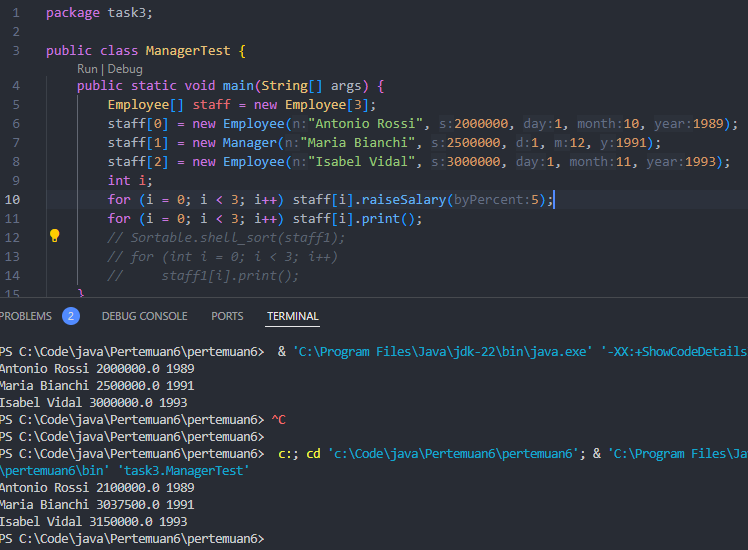
Selain itu, struktur ini menggunakan **multi-level single inheritance**, di mana subclass seperti Square mewarisi atribut dan method dari Rectangle, yang dimana Rectangle juga mewarisi atribut dan method dari Shape. Dengan begini Square dapat menggunakan method dan atribut yang berasal dari class Rectangle dan Shape.

1. Task 3.1



Hasil akhir program





Penjelasan:

Pada task 3.1, terdapat dua kelas, yaitu Employee dan Manager, di mana Employee berfungsi sebagai superclass bagi Manager dan mewariskan atribut serta method-nya kepada kelas Manager. Dalam kelas Manager, terdapat method raiseSalary yang sama namanya dengan yang ada di kelas Employee, namun dengan implementasi yang berbeda.

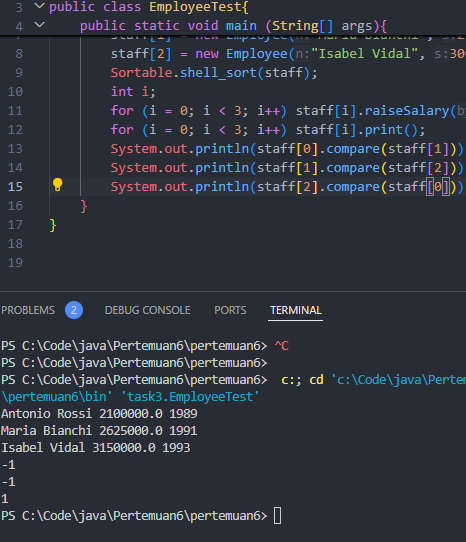
Meskipun method raiseSalary di kelas Manager tidak menggunakan overriding, saat kita memanggil method tersebut pada objek Manager, implementasi yang baru tetap dijalankan. Hal ini terjadi karena ketika objek yang dibuat adalah tipe Manager, maka Java akan menggunakan method raiseSalary yang ada pada kelas Manager meskipun tidak ada syntax @overriding, karena objek tersebut adalah instance dari subclass yang memiliki implementasi sendiri.

Pada kelas ManagerTest, kita membuat tiga objek Employee yang disimpan dalam bentuk array. Pada salah satu elemen array, kita mengisi dengan objek Manager (lebih tepatnya pada elemen kedua). Ini menunjukkan bahwa objek Manager dapat diperlakukan sebagai objek Employee karena Manager adalah subclass dari Employee. Dengan cara ini, kita dapat memanfaatkan polymorphism di mana metode yang dipanggil tergantung pada tipe objek yang sebenarnya, bukan pada tipe referensinya.

Maka dari itu pada saat object dengan nama Maria Bianchi dilakukan raiseSalary dan di print pada file class EmployeeTest dan ManagerTest memiliki hasil yang berbeda. Karena object dengan nama Maria Bianchi pada file class EmployeeTest merupakan object instansi dari class Employee maka dari itu dia menjalankan method raiseSalary() dari class Employee. Sedangkan object dengan nama Maria Bianchi pada file class ManagerTest meerupakan instalnsi dari class Manger maka dari itu dia menjalankan method raiseSalary() dari class Manager.

Pertanyaan:

1. When Sortable extended to Employee class, the method compare will be implemented. Please try the codes above. Call the method compare, in EmployeeTest class

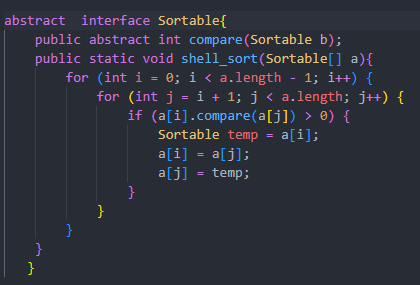


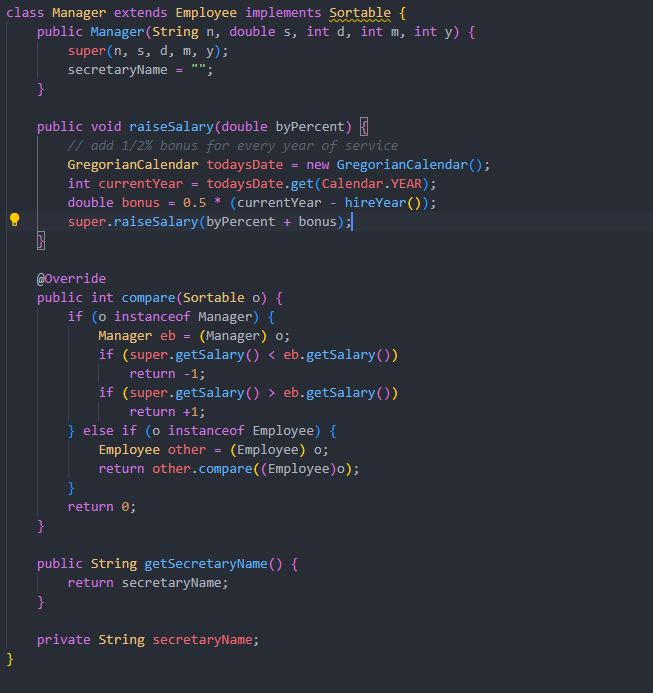
1. Imagine that we want to order the Managers in a similar way : class Managers extends Employee extends Sortable It will be work? What is your solution?

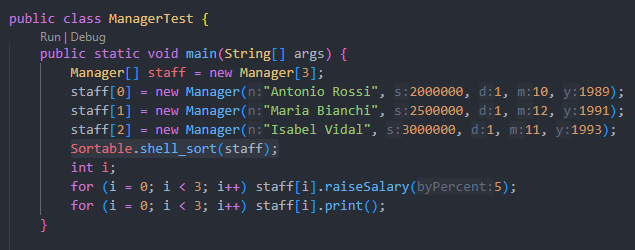
Untuk mencoba membuat sebuah class manager yang merupakan subclass dari sortable dan juga Employee dengan syntax class Managers extends Employee extends Sortable tidak akan bekerja, mengapa demikian? Hal ini dikarenakan java tidak memperbolehkan penulisan pewarisan 2 supperclass pada satu subclass dengan cara seperti ini, java akan memberikan pesan error pada saat menuliskan kedua syntax tersebut.

Untuk Solusi dari saya adalah kita tidak perlu membuat subclass Manager yang merupakan subclass dari superclass Employee dan Sortable, mengapa begitu? Hal ini karena class employee sendiri telah menjadi subclass dari class Sortable maka dari itu kita hanya perlu membuat class manager mejadi subclass dari class Employee. Dengan demikian class Manager tetap bisa menagakses dan menerima warisan atirbut dan method dari kedua class tersebut.

Namun, misal terdapat case dimana class Employee bukan merupakan subclass dari Sortable, kita dapat menggunakan konsep interface. Dalam hal ini, baik Employee maupun Sortable dapat memiliki metode dengan nama yang sama, seperti compare. Dengan mengimplementasikan interface Sortable di kelas Employee, Untuk contoh implmentasinya adalah sebagai berikut







Masalah:-

Solusi:-

Teman yang membantu: -

Link github source code week-6 [Tugas-PBO/Week-6 at main · RaditZX/Tugas-PBO (github.com)](https://github.com/RaditZX/Tugas-PBO/tree/main/Week-6)