DOKUMENTASI PROJEK KALKULATOR BANGUN DATAR DENGAN FACTORY METHOD

OLEH:
KELOMPOK 11

1. Abiyyu Cakra - 0110221100
2. Ahmad Fauzi Ariyanto - 0110221280
3. Ahmad Ihsanullah Rabbani - 0110221327
4. Isep Irmansyah - 0110221322
5. Salma Rihhadatul 'Aisy - 0110221102
6. Zulfikar Hilman Prayogo - 0110221279

PROGRAM STUDI TEKNIK INFORMATIKA
STT TERPADU NURUL FIKRI
DEPOK
JANUARI 2023

DAFTAR ISI

BAB I
Pendahuluan

A. Latar Belakang

Dalam penyusunan sebuah aplikasi banyak cara untuk melakukan penyusunan baris kode bukan hanya untuk terlihat rapi, namun dapat membuat program berjalan lebih efektif dalam arti kode yang tidak berulang, lebih cepat dll. Hal itu dapat kita lakukan dengan menyusun kode dengan menggunakan *design pattern*.

Design pattern atau yang dapat diartikan sebagai pola desain adalah metode yang dibuat untuk membantu tim pengembang dalam menemukan solusi dari masalah-masalah umum yang muncul saat pengembangan perangkat lunak sedang berlangsung.

Pola desain ini dapat digunakan kembali dalam pengembangan perangkat lunak selanjutnya. Ia bukanlah suatu metode yang dapat diimplementasikan langsung menjadi sebuah kode program, tetapi ia merupakan sebuah pola untuk menyelesaikan masalah dalam situasi yang bermacam-macam.

Salah satu dari pola design pattern adalah Factory Method. Factory Method Design Pattern adalah Creational Design Pattern yang bertugas untuk menyediakan interface untuk pembuatan implementasi objek dari sebuah superclass.

Pada kasus kali ini kami mencoba membuat suatu program “aplikasi penghitung bangun datar” dengan mengimplementasikan pola design pattern Factory Method. Yang akan berfungsi dalam pencarian luas dan keliling bangun datar tersebut.

B. Tujuan

Tujuan dari pengembangan aplikasi ini adalah:

1) Memahami apa itu design pattern.
2) Memahami implementasi dari pola Factory Method dalam pembuatan aplikasi rumus matematika bangun datar.

C. Manfaat

Manfaat dari pengembangan aplikasi ini adalah:

1) Memberikan kemudahan dalam menghitung luas dan keliling bangun datar
2) Membantu mempercepat dalam perhitungan luas dan keliling bangun datar
3) Membantu dalam pemecahan masalah baik bilangan sederhana maupun bilangan rumit

D. Teknologi dan Tools yang Digunakan

Teknologi dan tools yang digunakan adalah:

1) PHP 8.1 (Object Oriented)
2) Text Editor (VSCode)

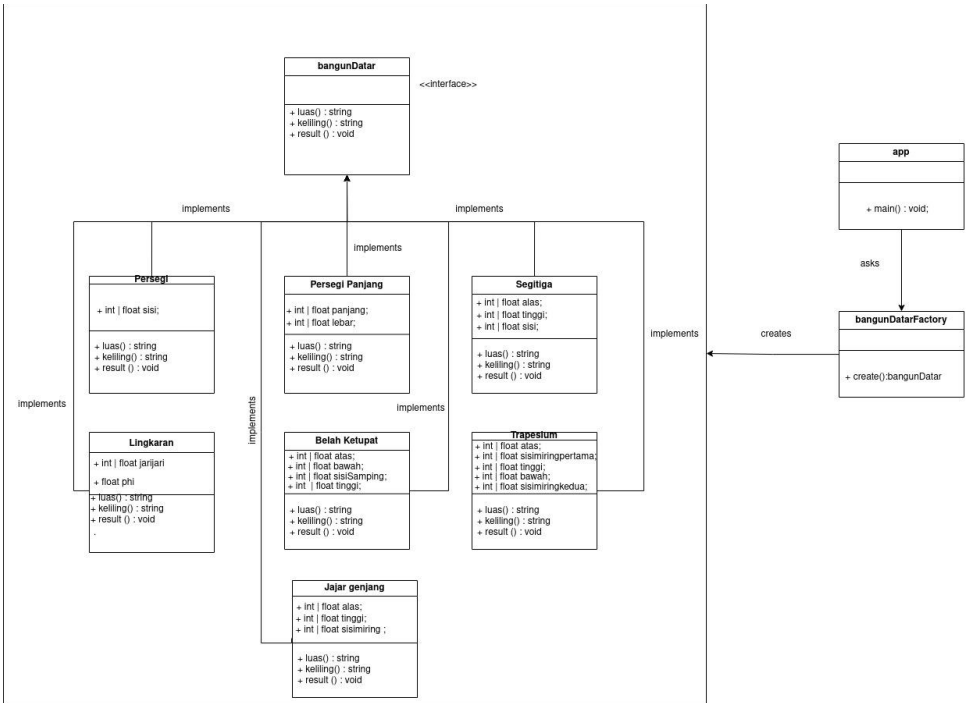
BAB II
HASIL

A. Pattern

Factory Pattern adalah pattern yang digunakan untuk memisahkan (*decouple*) proses pembuatan/instansiasi sebuah objek (produk) dari objek lain (klien) yang menggunakannya. Tujuannya supaya perubahan pada *product class* nggak menyebabkan kita harus mengubah kode pada *client*. Paling nggak akibat dari perubahan itu bisa diminimalisir. Dan juga supaya si *factory* bisa digunakan oleh banyak class.

Contoh penggunaan factory method di dunia nyata sama seperti pabrik mainan plastik. Bahan material yang digunakan untuk membuat mainan selalu sama, tapi untuk mainan yang berbeda bisa dibuat dengan cetakan yang berbeda. sama seperti halnya factory method yang dimana input nya adalah nama dari mainan yang kita minta dan outputnya adalah mainan plastik yang kita inginkan.

B. Class Diagram



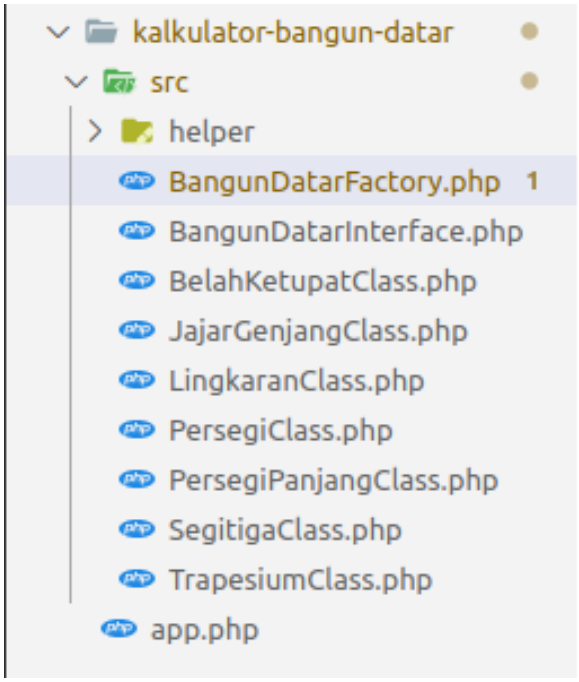
BangunDatarInterface diimplementasi oleh 7 class bangun datar. Diluar kotak, BangunDatarFactory akan memanggil 7 class bangun datar untuk digunakan di file app.php

C. Hasil Running Program

Hasil eksekusi program kalkulator bangun datar dengan menjalankan file app.php. tampilan program ini berupa tampilan CLI (*Command Line Interface*).

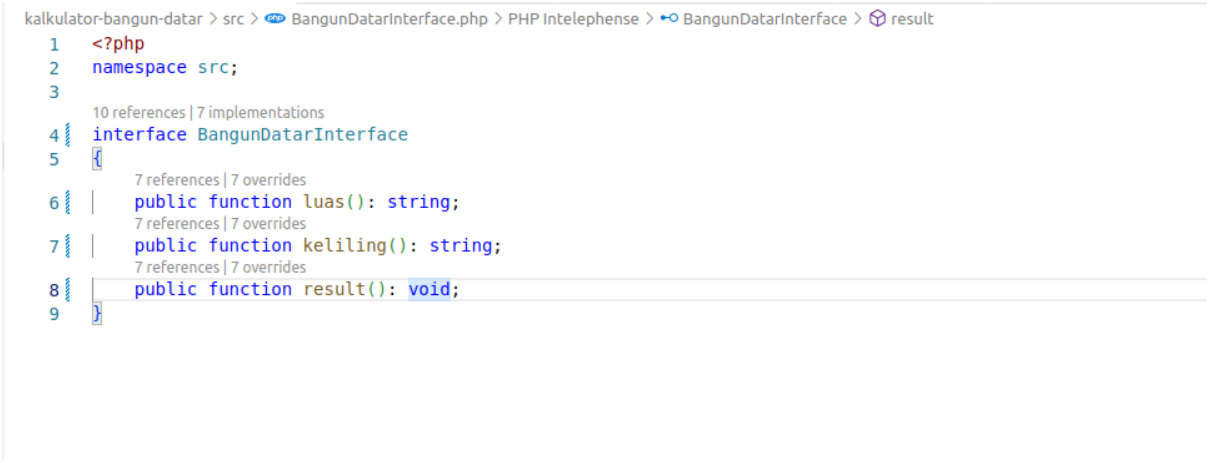
D. Source Code

Struktur direktori projek :



1. **BangunDatarInterface.php**

Interface ini diimplementasi oleh class-class bangun datar. ada 3 method yaitu luas, keliling dan result.



2. **helper.php**

Helper ini mempunyai adalah function input yang akan menerima inputan dari user (Stdin), yang nantinya akan dipanggil di banyak class untuk mengambil inputan yang akan dihitung.

```php
<?php

function input(string $info):string
{
    echo "$info: ";
    $result = fgets(STDIN);
    return trim($result);
}
```

3. LingkaranClass.php

LingkaranClass merupakan contoh class yang mengimplentasi Interface BangunDatarInterface. Terdapat tiga method yang diturunkan dari BangunDatarInterface, yaitu method luas, keliling dan result. Kemudian satu method construct, yang akan dijalankan otomatis ketika class dipanggil oleh BangunDatarFactory.

LingkaranClass.php 1

kalkulator-bangun-datar > src > LingkaranClass.php > PHP > src\LingkaranClass

```php
<?php
namespace src;

require_once "helper/helper.php";
require_once 'BangunDatarInterface.php';

2 references | 0 implementations
class LingkaranClass implements BangunDatarInterface
{
    2 references
    private float $phi = 3.14;
    6 references
    private float $jariJari;

    1 reference | 0 overrides
    public function __construct()
    {
        $input = input("masukan jari-jari ");
        $replace = str_replace(",", ".", $input);
        $this->jariJari = $replace;
    }

    7 references | 0 overrides
    public function luas()
    {
        $total = $this->phi * (float) $this->jariJari * (float) $this->jariJari;
        return "Luas Lingkaran dengan jari-jari $this->jariJari cm = $total cm";
    }

    7 references | 0 overrides
    public function keliling()
    {
        $total = 2 * (float) $this->phi * (float) $this->jariJari;
        return "Keliling lingkaran dengan jari-jari $this->jariJari cm = $total cm";
    }

    7 references | 0 overrides
    public function result()
    {
        echo "------------" . PHP_EOL;
        echo $this->luas() . PHP_EOL;
        echo $this->keliling() . PHP_EOL;
        echo "------------" . PHP_EOL;
    }
}
```

4. BangunDatarFactory.php

BangunDatarFactory memanggil semua class yang mengimplementasi BangunDatarInterface.

kalkulator-bangun-datar > src > BangunDatarFactory.php > ...

```php
<?php
namespace src;

require_once "LingkaranClass.php";
require_once "BelahKetupatClass.php";
require_once "SegitigaClass.php";
require_once "JajarGenjangClass.php";
require_once "PersegiClass.php";
require_once "PersegiPanjangClass.php";
require_once "TrapesiumClass.php";


8 references | 0 implementations
class BangunDataFactory
{
    7 references | 0 overrides
    static function create($type): BangunDatarInterface
    {
        if ($type == "lingkaran") {
            return new LingkaranClass();
        } else if ($type == "persegi") {
            return new PersegiClass();
        } else if ($type == "persegi panjang") {
            return new PersegiPanjangClass();
        } else if ($type == "segitiga") {
            return new SegitigaClass();
        } else if ($type == "trapesium") {
            return new TrapesiumClass();
        } else if ($type == "belah ketupat") {
            return new BelahKetupatClass();
        } else if ($type == "jajar genjang") {
            return new JajarGenjangClass();
        }

    }
}
```

5. app.php

File app.php menampilkan interface menu dari program ini. file ini memanggil beberapa file seperti BangunDatarFactory dan helper. BangunDatarFactory dipanggil ketika user menginputkan tipe bangun datar yang ingin dihitung.

kalkulator-bangun-datar > app.php > ...

```php
<?php

use src\BangunDataFactory;

require_once "./src/helper/helper.php";
require_once "./src/BangunDatarFactory.php";

$isContinue = true;

while ($isContinue) {

    clearScreen();

    echo " ===== Selamat Datang di Kalkulator Bangun Datar ===== " . PHP_EOL;
    echo "Pilihan Bangun Datar" . PHP_EOL;
    echo "1. lingkaran" . PHP_EOL;
    echo "2. Belah Ketupat" . PHP_EOL;
    echo "3. Segitiga" . PHP_EOL;
    echo "4. Trapesium" . PHP_EOL;
    echo "5. Jajar genjang" . PHP_EOL;
    echo "6. Persegi" . PHP_EOL;
    echo "7. Persegi Panjang" . PHP_EOL;
    $data = input("Masukan Pilihan Anda");
    echo "---------------------" . PHP_EOL;

    switch (strtolower($data)) {
        case '1':
        case 'lingkaran':
            BangunDataFactory::create("lingkaran")->result();
            break;
        case '2':
        case 'belah ketupat':
            BangunDataFactory::create("belah ketupat")->result();
            break;
        case '3':
        case 'segitiga':
            BangunDataFactory::create("segitiga")->result();
            break;
        case '4':
        case 'trapesium':
            BangunDataFactory::create("trapesium")->result();
            break;
        case '5':
        case 'jajar genjang':
            BangunDataFactory::create("jajar genjang")->result();
            break;
        case '6':
        case 'persegi':
            BangunDataFactory::create("persegi")->result();
```

BAB III
Penutup

A. Kesimpulan

Factory method merupakan pola desain yang sering digunakan, jika kita berpikir membutuhkan factory hanya untuk membuat objek, lebih baik tidak digunakan, karena hanya akan membawa kompleksitas yang tidak diperlukan. Namun, jika kita berpikir factory dibutuhkan karena ada banyak objek dengan tipe dasar yang sama dan kita manipulatif sebagai abstrak objek, maka sebaiknya kita gunakan factory.

B. Saran

Walaupun kami menginginkan kesempurnaan dalam penyusunan dokumentasi ini, akan tetapi pada kenyataannya masih banyak kekurangan yang perlu kami perbaiki. Hal ini dikarenakan minimnya pengetahuan kami dan manusia tidak luput dari kesalahan.

Oleh karena itu kritik dan saran yang membangun dari pembaca sangat diharapkan sebagai bahan evaluasi untuk kedepannya. Sehingga kami bisa menghindari atau memperbaiki kesalahan-kesalahan yang ada.

Referensi

- Setiawan, Rony. “Design Pattern untuk Pembuatan Perangkat Lunak.” *Dicoding*, 10 November 2021, https://www.dicoding.com/blog/design-pattern/. Diakses pada 13 Januari 2023.
- Bratadinata, Anggie. 2009 "Factory Pattern", https://masputih.com/2009/05/factory-pattern, diakses pada 13 Januari 2023
- AlifianAdexe. 2019 “Ragam Design Pattern : Factory”, https://www.anbidev.com/design-pattern-factory/, diakses pada 13 Januari 2023