PERTEMUAN 12

TAHAPAN PERANCANGAN BERORIENTASI OBJEK

POKOK BAHASAN

1. Identifikasi kelas, atribut, responsibility dan operasi
2. Menentukan hubungan antar objek dan kelas
3. Perancangan algoritma untuk operasi menggunakan desain struktur
4. Mengembangkan tes algoritma atau menjalankan algoritma

IDENTIFIKASI KELAS, ATRIBUT, RESPONSIBILITY DAN OPERASI

- Sama seperti pendekatan top-down, untuk mendefinisikan garisbawahi kata benda dan sifat yang relevan dengan masalah.
- Contoh

Dibutuhkan program untuk membaca file karyawan yang berisikan berisi nomor karyawan, tingkat upah, dan jumlah jam kerja per minggu. Program kemudian memvalidasi upah dan jam kerja. Jika benar, hitung upah mingguan pegawai dan cetak payslip

IDENTIFIKASI KELAS, ATRIBUT, RESPONSIBILITY DAN OPERASI (lanjutan)

Validasi : menurut aturan perusahaan, maksimal seorang pegawai bekerja 60 jam seminggu dan maksimum 250,000 per jam. Jika jam bekerja ata u tingkat upah melebihi aturan, input data dan pesan yang sesuai akan dicetak dan upah mingguan karyawan tidak dihitung.

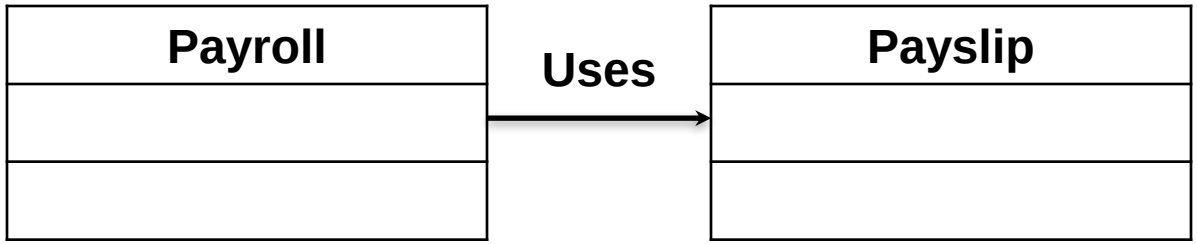
Hitung upah mingguan : upah mingguan dihitung jam kerja dikali tingkat upah. Jika lebih dari 35 jam kerja, pembayaran untuk lembur jam kerja dihitung setengahnya.

IDENTIFIKASI KELAS, ATRIBUT, RESPONSIBILITY DAN OPERASI (lanjutan)

Kelas	Atribut	Responsibility	operasi
Payslip	empNum	Validasi data	+setEmpNum() +setPayRate() +setHoursWorked()
	payRate	Laporan validasi data	+getValidInput(ValidInput)
	hoursWorked	Hitung upah mingguan	-setPay() (mutator)
	weeklyPay	Cetak slip gaji untuk setiap absen yang valid	+printPaySlip()
Payroll	File data absen	Baca file data absen	+runTimeSheet() +run()

MENENTUKAN HUBUNGAN ANTAR OBJEK DAN KELAS

- Berdasarkan pernyataan masalah diperoleh dua kelas yaitu penggajian dan payslip. Objek dari kelas penggajian menggunakan objek dari kelas payslip. Hubungan antar kelas digambarkan dengan notasi UML.



PERANCANGAN ALGORITMA UNTUK OPERASI MENGGUNAKAN DESAIN STRUKTUR

- Setiap operasi dalam kelas memerlukan penjelasan langkah-demi langkah instruksi yang menghasilkan perilaku yang diperlukan. Masing-masing algoritma dimulai dengan nama operasi dan diakhiri dengan pernyataan END.
- Data yang dibutuhkan objek payslip dikirim dari penggajian segera setelah diciptakan. Nilai atribut tersedia untuk semua operasi di dalam kelas
- dan empNumber, weeklyPay, validInput, hoursWorked dan payRate dapat dilihat untuk setiap operasi.

PERANCANGAN ALGORITMA UNTUK OPERASI MENGGUNAKAN DESAIN STRUKTUR (lanjutan)

```
setEmpNumber(inEmpNumber)
        empNumber = inEmpNumber
END
setPayRate(inPayRate)
        set errorMessage to blank
        IF inPayRate>MAX_PAY THEN
        errorMessage = ”Rate upah melebihi  250,000”
        Print empNumber, inPayRate, errorMessage
                validInput = false
        ELSE  payRate = inPayRate
        END IF
END
```

PERANCANGAN ALGORITMA UNTUK OPERASI MENGGUNAKAN DESAIN STRUKTUR (lanjutan)

setHoursWorked(inHoursWorked)

```
Set errorMessage to blank
IF inHoursWorked > MAX_HOURS THEN
        errorMessage = ”Jam kerja melebihi 60”
        Print empNumber, inHoursWorked, errorMessage
        validInput = false
ELSE
        hoursWorked = inHoursWorked
ENDIF
END
```

PERANCANGAN ALGORITMA UNTUK OPERASI MENGGUNAKAN DESAIN STRUKTUR (lanjutan)

setPay()

```
IF hoursWorked <= NORM_HOURS THEN
        weeklyPay = payRate * hoursWorked
ELSE
        overtimeHours = hoursWorked – NORM_HOURS
        overtimePay = overtimeHours * payRate * 1.5
        weeklyPay = (payRate * NORM_HOURS) + overtimePay
ENDIF
END
```

PERANCANGAN ALGORITMA UNTUK OPERASI MENGGUNAKAN DESAIN STRUKTUR (lanjutan)

Konstruktor paySlip dapat menggunakan mutator untuk diperbaharui nilai-nilai atribut. Namun, jika data tidak valid, nilai tidak akan diperbaharui, mungkin diperbaharui dengan data yang tidak valid. Oleh karena itu, standar konstruktor harus memperbaharui seluruh atribut.

Payslip()

Set empNumber to ”Uknown”

Set payRate to 0.0

Set hoursWorked to 0

Set validInput to true

Set weeklyPay to 0.0

END

PERANCANGAN ALGORITMA UNTUK OPERASI MENGGUNAKAN DESAIN STRUKTUR (lanjutan)

Konstruktor menerima input yang dapat memanggil standar konstruktor sebelum pengaturan nilai atribut

Payslip(inEmpNumber, inPayRate, inHoursWorked)

Payslip()

setEmpNumber(inEmpNumber)

setPayRate(inPayRate)

SetHoursWorked(inHoursWorked)

IF validInput THEN

    setPay()

ENDIF

END

PERANCANGAN ALGORITMA UNTUK OPERASI MENGGUNAKAN DESAIN STRUKTUR (lanjutan)

Objek eksternal perlu mengetahui tentang validitas payslip dan upah mingguan yang dihitung. Aksesor menginformasikan objek lain tentang validitas payslip

getValidInput(validInput)

END

printPayslip()

IF validInput THEN

    Print empNumber, payRate, hoursWorked, weeklypay

ENDIF

END

MENGEMBANGKAN TES ATAU MENJALANKAN ALGORITMA

- Langkah terakhir adalah mengembangkan kelas yang memiliki operasi dengan algoritma seperti baris utama pada program prosedural.
- Menjalankan algoritma dengan membuat kelas yang berisikan algoritma utama.
- Tes atau menjalankan kelas menyediakan interface antara objek payslip dan lingkungan luar termasuk pengguna.
- File data absen merupakan bagian dari interface antara lingkungan luar dan objek payslip. Data absen akan menjadi atribut pada driver kelas dan driver konstruktor akan membuka file.

```
Payroll()
    timeSheetFile = ”timesheet.dat”
END
```

MENGEMBANGKAN TES ATAU MENJALANKAN ALGORITMA (lanjutan)

Input	Proses	Output
Data Absen empNumber hoursWorked payRate	Baca data absen Buat payslip dari data absen Cetak paySlip yang valid	Cetak paySlip Pesan kesalahan untuk paySlip yang tidak valid

Algoritma untuk driver class Payslip

run()

Baca empNumber, hoursWorked, payRate

DoWhile more records

Create paySlip as New Payslip(empNumber, payRate, hoursWorked)

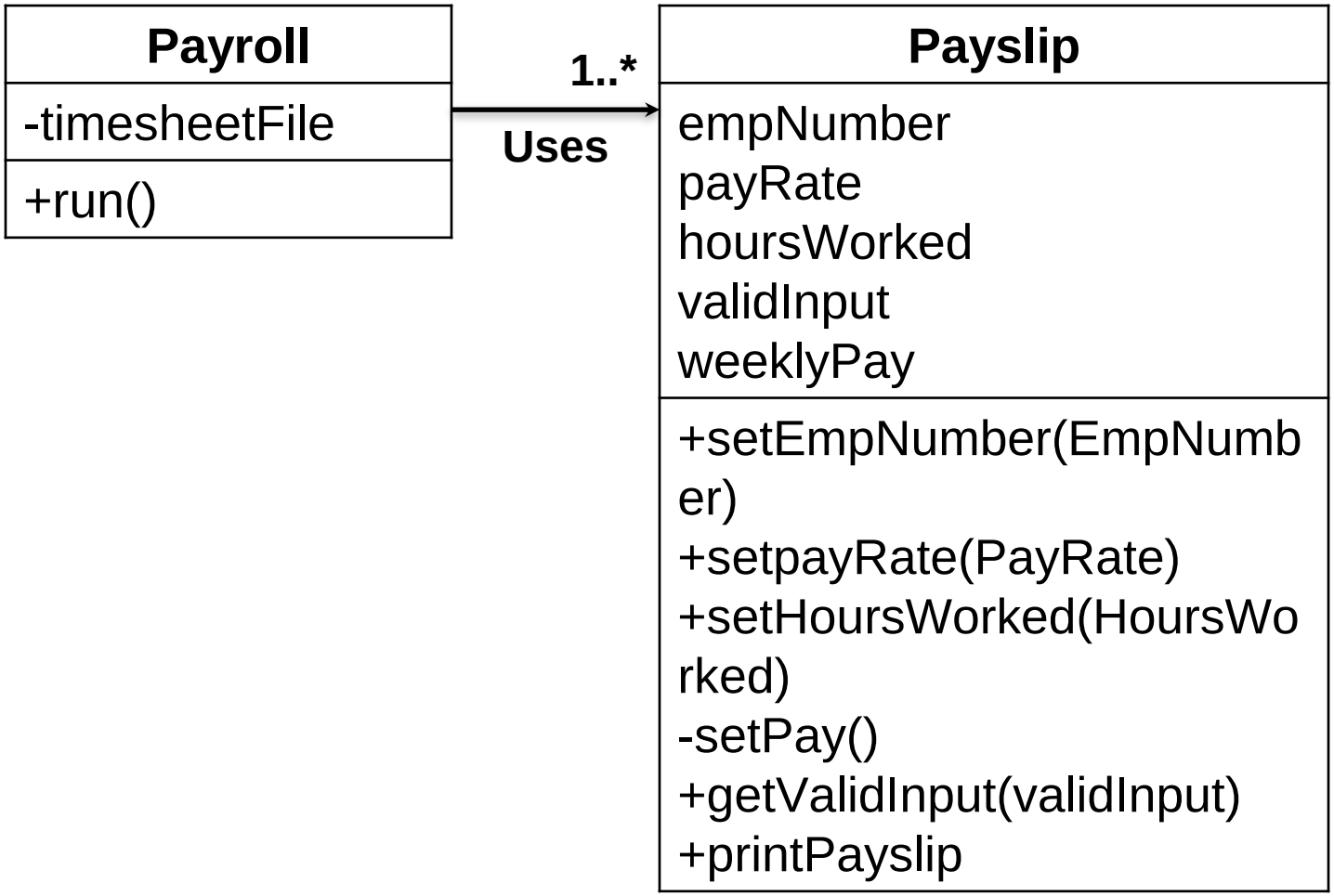
payslip.printPayslip()

Baca Payslip(empNumber, payRate, hoursWorked

Enddo

END

MENGEMBANGKAN TES ATAU DRIVER ALGORITMA (lanjutan)



Analisa kasus

Sebuah sawah berbentuk segitiga kalimat ini dapat dimaknai kita dapat membentuk Class SAWAH_SEGITIGA

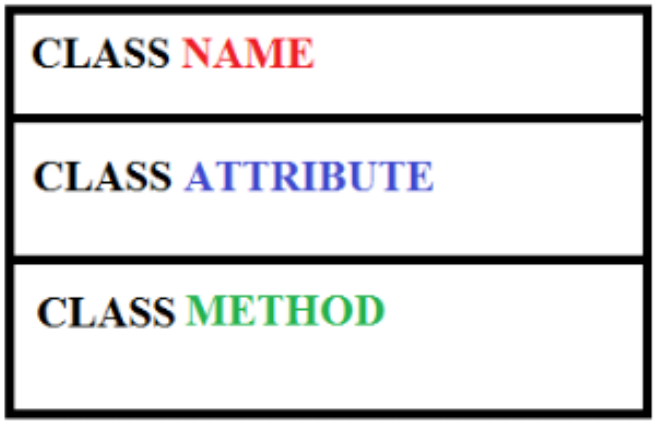
Tetapi bagaimana jika kita dihadapkan dengan bentuk sawah yang beda-beda (SEGITIGA, BUJUR_SANGKAR, PERSEGI_PANJANG, POLYGON)?

Sawah dapat menjadi ABSTRACT dan bentuk sawah(SEGITIGA, BUJUR_SANGKAR, PERSEGI_PANJANG, POLYGON) merupakan turunan dari sawah.

Abstrak dapat diumpamakan sebagai kepala tanpa badan, dimana informasi berupa konsep.

Abstrak tidak dapat digunakan langsung, hanya turunannya yang dapat menggunakan attribute/method tersebut

Review pengertian class



Struktur Class terdiri dari 3 bagian

1. Class Name
2. Class Attribut

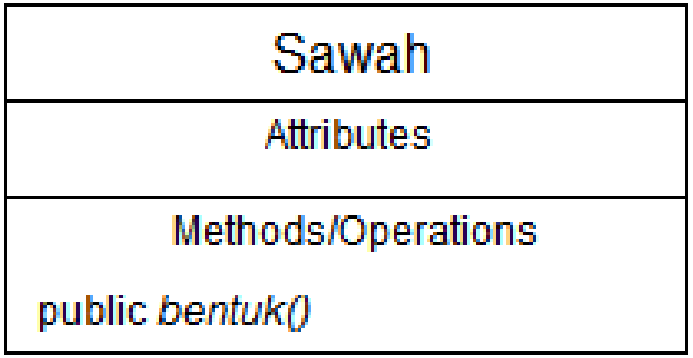
   Memiliki visibility : **(-)** Private; **(+)** Public; **(#)** Protected
3. Class Method

   Dapat dibentuk dengan operasi Aksesor (get) dan Mutator (set) dari attribut yang ada, dalam java disebut POJO (Plain Old Java Object)

Tetapi tidak semua class harus memiliki 3 bagian, bisa jadi hanya berisi **Class Name dan Attribut** saja atau **Class Name dan Method** saja

Abstract “Sawah”

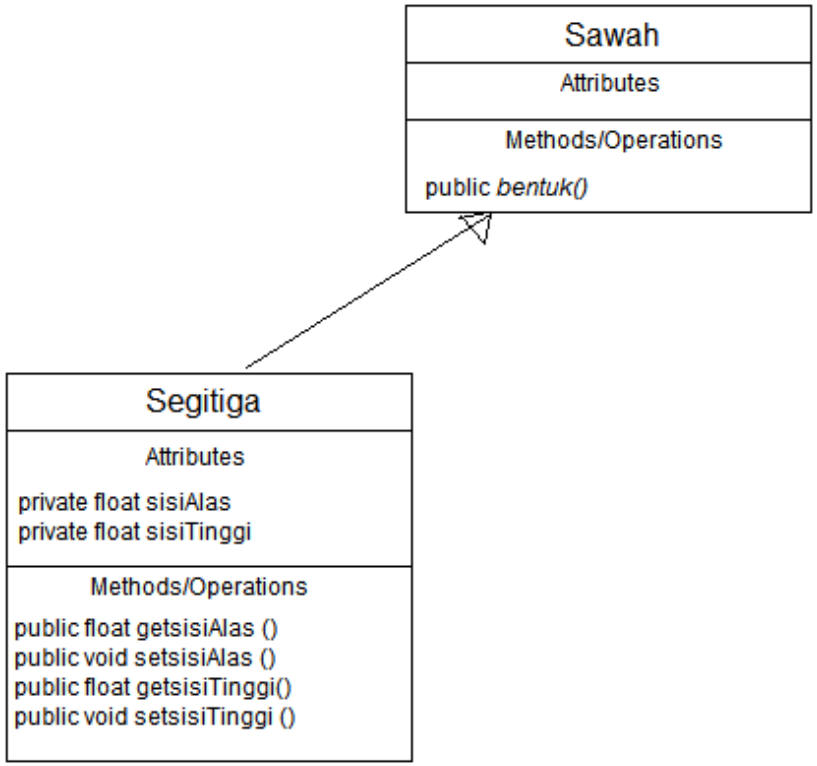
Sawah tidak memiliki atribut, tetapi memiliki methods bentuk () yang dapat mendeskripsikan bentuk sawah.



Sawah Segitiga

(sawah) Segitiga merupakan instant/class turunan dari sawah. Class ini memiliki 2 atribut dimensi (Alas dan Tinggi).

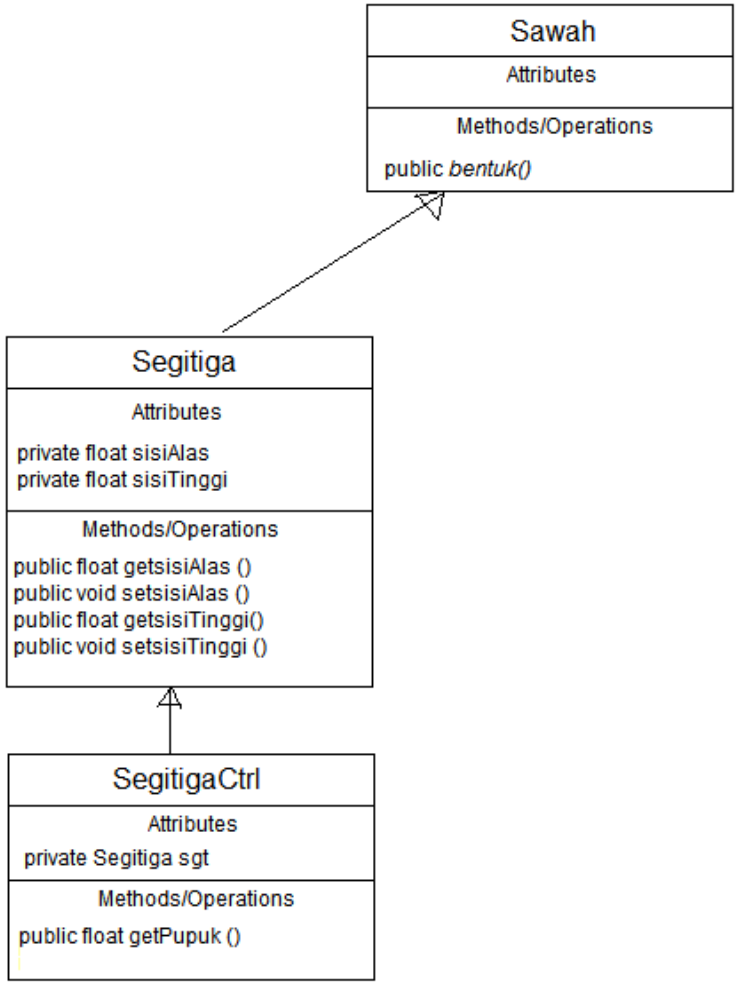
Methods (get dan set) digunakan untuk “mengisi” dan “mengambil” nilai dari attribute



Cara hitung Kebutuhan PUPUK?

Hitung kebutuhan pupuk kita bentuk class SegitigaCtrl (fungsi nya untuk mengontrol class Segitiga)

Methods getPupuk () digunakan untuk menghitung berapa jumlah pupuk yang diperlukan.

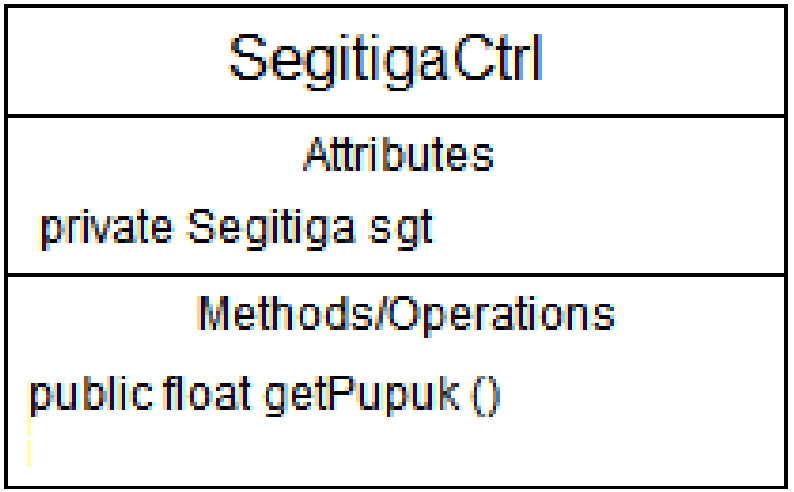


Kenapa SegitigaCtrl?

Class ini dibentuk untuk "mengendalikan" class segitiga tanpa harus merubah class Segitiga.

Kasus saat ini mencari berapa jumlah pupuk yang diperlukan, jika nanti kita perlu menghitung luas atau keliling segitiga bagaimana?

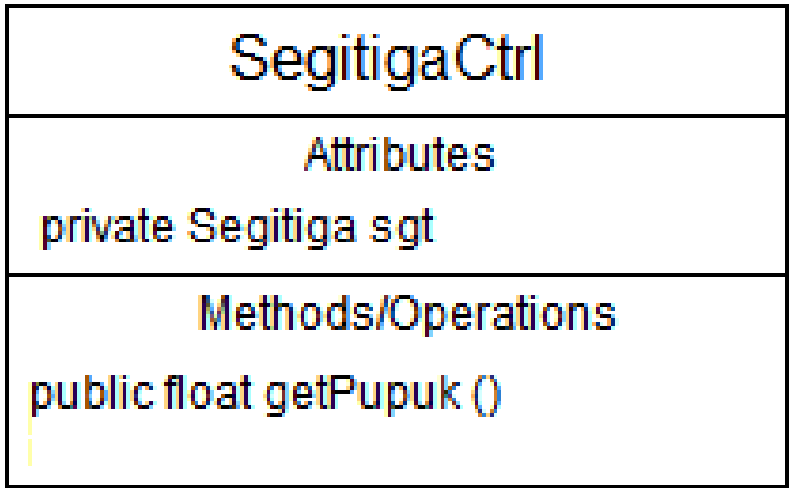
Kita cukup menambahkan fungsi getLuas () dan/atau getkeliling () disini



Private Segitiga sgt

Adalah instant yang menunjukkan bahwa class ini merupakan class turunan dari segitiga.

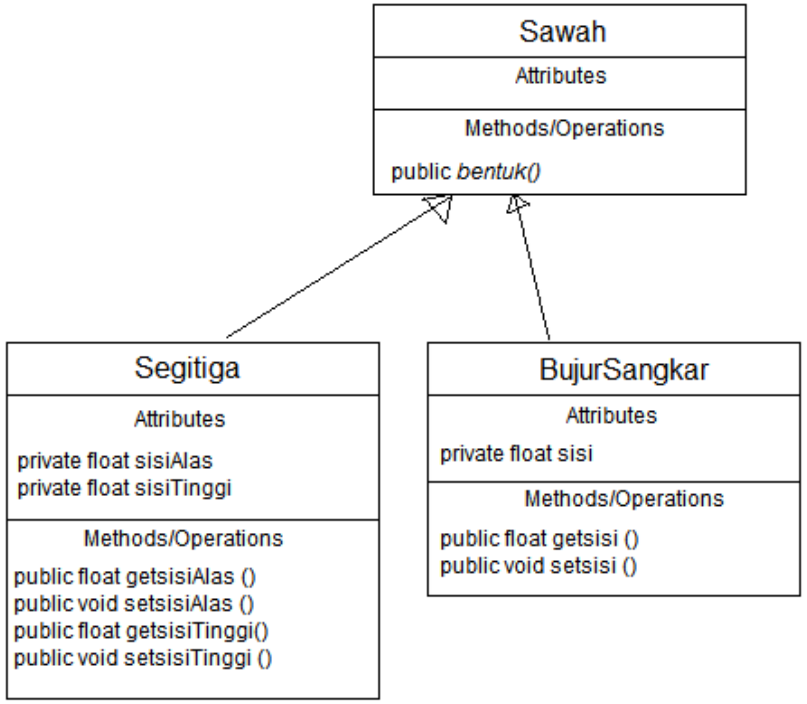
Artinya class SegitigaCtrl dapat menggunakan seluruh methods dari class (induk) Segitiga. Kita cukup fokus kan pada fungsi apa yang hendak kita terapkan di segitiga.



BagaimanaJika bentuk sawah berbeda? Bujursangkar misalnya

Kita cukup "extend" sawah dengan BujurSangkar.

Kenapa attributnya "private" karena attribute ini hanya bias diakses **HANYA** oleh class itu atau turunannya saja.



Kenapa

Class control untuk Segitiga dan BujurSangkar dibuat terpisah?

Karena fungsi nya berbeda, ketika kita menghitung luas segitiga, akan berbeda dengan cara menghitung luas bujur sangkar.