

Started on	Wednesday, 22 February 2023, 3:30 PM
State	Finished
Completed on	Wednesday, 22 February 2023, 3:40 PM
Time taken	10 mins 48 secs
Grade	600.00 out of 600.00 (100%)

Time limit	1 s
Memory limit	64 MB

Polymorphism merupakan konsep dimana sebuah subclass dapat mendefinisikan sifatnya sendiri dan dapat menggunakan fungsionalitas yang sama dari parent classnya. Pada soal ini, akan diuji pemahaman kalian terkait polymorphism

Pada nomor ini, terdapat sebuah parent class yang dapat dilihat pada [Motor.java](#), dimana:

- Merupakan parent class
- Memiliki konstruktor berupa numberOfWheels dan engineCapacity
- Memiliki getter untuk masing-masing atributnya
- Mengeluarkan suara "Ngenggg"
- Dapat mendeskripsikan dirinya dengan format "Motor ini memiliki a roda dengan kapasitas mesin b cc", dimana a adalah numberOfWheels dan b adalah engineCapacity

Praktikan akan diminta untuk melengkapi 2 buah file, yaitu FCX.java dan Fespa.java, dimana:

1. FCX.java

- Merupakan subclass dari Motor java
- Memiliki konstruktor yang sama dengan parentnya, dengan tambahan berupa luggageCapacity dan isIdleStoping
- Memiliki getter untuk masing-masing atributnya
- Mengeluarkan suara "Brmmm"
- Dapat mendeskripsikan dirinya sesuai dengan keadaan:
 1. Apabila isIdleStoping true, maka mengembalikan: "Motor ini memiliki a roda dengan kapasitas mesin b cc, memiliki kapasitas bagasi c liter, dan sedang dapat berhenti otomatis apabila didiamkan" dengan a adalah numberOfWheels, b adalah engineCapacity dan c adalah luggageCapacity
 2. Apabila isIdleStoping false, maka mengembalikan: "Motor ini memiliki a roda dengan kapasitas mesin b cc, memiliki kapasitas bagasi c liter, dan sedang **tidak** dapat berhenti otomatis apabila didiamkan" dengan a adalah numberOfWheels, b adalah engineCapacity dan c adalah luggageCapacity

2. Fespa.java

- Merupakan subclass dari Motor java
- Memiliki konstruktor yang sama dengan parentnya, dengan tambahan berupa color dan isRoundedFrontLamp
- Memiliki getter untuk masing-masing atributnya
- Mengeluarkan suara "Trototong"
- Dapat mendeskripsikan dirinya sesuai dengan keadaan:
 1. Apabila isRoundedFrontLamp true, maka mengembalikan: "Motor ini memiliki a roda dengan kapasitas mesin b cc, memiliki warna c, dan memiliki lampu depan berbentuk lingkaran" dengan a adalah numberOfWheels, b adalah engineCapacity dan c adalah color
 2. Apabila isRoundedFrontLamp false, maka mengembalikan: "Motor ini memiliki a roda dengan kapasitas mesin b cc, memiliki warna c, dan memiliki lampu depan berbentuk **persegi panjang**" dengan a adalah numberOfWheels, b adalah engineCapacity dan c adalah color

Submit zip file(nama dibebaskan) yang berisi [FCX.java](#) dan [Fespa.java](#)

Java 8

 [motor.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.07 sec, 28.90 MB
2	25	Accepted	0.07 sec, 28.17 MB

No	Score	Verdict	Description
3	25	Accepted	0.07 sec, 27.93 MB
4	25	Accepted	0.07 sec, 30.45 MB

Time limit	1 s
Memory limit	64 MB

Polymorphism - MotorPrinter

Buatlah kelas `MotorPrinter` yang berguna untuk mencetak informasi `Motor` dan kelas turunannya (`Fespa` dan `FCX`) dari beberapa motor. Menggunakan kelas yang telah dibuat sebelumnya, kelas `MotorPrinter` memiliki atribut `testMotor` bertipe `Motor` dan `motorList` yang berbentuk `ArrayList of Motor`. `MotorPrinter` juga memiliki metode-metode sebagai berikut:

1.

`addMotor`, yang menerima parameter `newMotor` dengan tipe `Motor`, menambahkannya ke dalam atribut `motorList`, dan tidak mengembalikan apapun (*Hint: Penambahan elemen ke dalam sebuah objek ArrayList dapat menggunakan method `add(E element)` yang sudah tersedia*)
2.

`getMotorList`, yang mengembalikan atribut `motorList`
3.

`getTestMotor`, yang mengembalikan atribut `testMotor`
4.

`setTestMotor`, yang menerima parameter `motorType` bertipe `String`, kemudian menjalankan aksi berikut berdasarkan isi parameter `motorType` dan tidak mengembalikan apapun:
 - Apabila `motorType` bernilai `Motor`, maka atribut `testMotor` akan menjadi sebuah objek baru dengan kelas `Motor` yang memiliki parameter:
 - `numberOfWheels = 2`
 - `engineCapacity = 3`
 - Apabila `motorType` bernilai `FCX`, maka atribut `testMotor` akan menjadi sebuah objek baru dengan kelas `FCX` yang memiliki parameter:
 - `numberOfWheels = 2`
 - `engineCapacity = 6`
 - `luggageCapacity = 10`
 - `isIdleStoping = true`
 - Apabila `motorType` bernilai `Fespa`, maka atribut `testMotor` akan menjadi sebuah objek baru dengan kelas `Fespa` yang memiliki parameter:
 - `numberOfWheels = 2`
 - `engineCapacity = 2`
 - `color = Red`
 - `isRoundedFrontLamp = false`
 - Hint: Pengecekan kesamaan antara dua string sebaiknya menggunakan method `equals()` yang dimiliki kedua Object tersebut. Contoh: `"test".equals("test")` bernilai `true`*
5.


`printMotorList`, yang akan melakukan iterasi terhadap isi `motorList`, kemudian, untuk setiap `motor`, dia akan secara berturut-turut melakukan hal berikut:

1.

Tergantung *instance* dari `Motor`:
 - Untuk kelas `Motor` saja dan bukan *instance* dari childnya, maka cetak `"Motor: "` lalu cetak output fungsi `sound()` dengan semua kata dipisah oleh spasi dan diakhiri *new line*
 - Untuk kelas `FCX`, cetak `FCX: "`, lalu cetak output fungsi `sound()` dengan semua kata dipisah oleh spasi dan diakhiri *new line*
 - Untuk kelas `Fespa`, cetak `"Fespa: "`, lalu cetak output fungsi `sound()` dengan semua kata dipisah oleh spasi diakhiri *new line*
 - Contoh untuk kelas `FCX`, maka output yang benar adalah : `FCX Brmmm`
 - Hint: Pengecekan instance sebuah kelas dilakukan dengan menggunakan operator `instanceof` (Contoh: `"string" instanceof String`)*

2.

Memanggil metode `printDescription`
- Lengkapi file [MotorPrinter.java](#)

Submit file `MotorPrinter.java`
- Java 8
-  [MotorPrinter.java](#)
- Score: 100
- Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.08 sec, 28.49 MB
2	12	Accepted	0.07 sec, 28.41 MB
3	12	Accepted	0.08 sec, 29.04 MB
4	12	Accepted	0.07 sec, 29.45 MB
5	12	Accepted	0.07 sec, 28.82 MB
6	12	Accepted	0.07 sec, 28.03 MB
7	12	Accepted	0.08 sec, 29.96 MB
8	16	Accepted	0.08 sec, 29.00 MB

Question **3**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Diberikan *interface* sebagai berikut:

- 1. **Engine** (**Engine.java**)
- 2. **Tyre** (**Tyre.java**)

Dari *interface* **Engine**, terdapat dua kelas implementasi sebagai berikut:

- 1. **TwoStrokeEngine** (**TwoStrokeEngine.java**) yang mengeluarkan suara 'taktaktak' dan bervariasi besaran kapasitas mesinnya.
- 2. **FourStrokeEngine** (**FourStrokeEngine.java**) yang mengeluarkan suara 'brumbrum' dan bervariasi besaran kapasitas mesinnya.

Dari *interface* **Tyre**, terdapat dua kelas implementasi sebagai berikut:

- 1. **DonlupTyre** (**DonlupTyre.java**) yang bervariasi besaran tekanan ban dan lebar dari ban-nya.
- 2. **MachelinTyre** (**MachelinTyre.java**) yang bervariasi besaran tekanan ban namun hanya memiliki ban dengan lebar 200mm.

Diberikan juga *class* **Motor** yang telah dimodifikasi sebagai berikut:

- 1. Konstruktor menerima 3 parameter yaitu **numberOfWheels** bertipe **integer**, **engine** bertipe **Engine**, dan **tyre** bertipe **Tyre**.
- 2. Terdapat tambahan method pada kelas:

- setEngine
- setTyre
- getNumberOfWheels
- getEngine
- getTyre
- sound
- printDescription

Si Cello ingin membangun sebuah bengkel motor untuk melakukan pengubahan ban dan mesin motor, oleh karena itu dibuatlah *class* **MotorWorkshop** (**MotorWorkshop.java**) untuk mewujudkan keinginannya.

Class **MotorWorkshop** memiliki dua method sebagai berikut:

- 1. changeTyre

Melakukan penggantian ban pada motor.

- 2. changeEngine

Melakukan penggantian mesin pada motor.

Lengkapilah metode-metode pada file **DonlupTyre.java**, **MachelinTyre.java**, **FourStrokeEngine.java**, **TwoStrokeEngine.java**, **Motor.java** dan **MotorWorkshop.java** lalu submit sebagai **MotorWorkshop.zip**.

Note: Perlu diingat bahwa kelas yang mengimplementasikan sebuah *interface* dapat dianggap sebagai *interface* tersebut atau apa yang disebut sebagai *polymorphism*.

Jangan lupa memberi *newline* pada setiap output **print**.

Attachments: [attachments.zip](#)

Java 8

 [MotorWorkshop.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	14	Accepted	0.08 sec, 28.55 MB
2	14	Accepted	0.07 sec, 28.14 MB
3	14	Accepted	0.08 sec, 28.15 MB
4	14	Accepted	0.08 sec, 26.65 MB
5	14	Accepted	0.07 sec, 28.59 MB
6	14	Accepted	0.08 sec, 27.82 MB
7	16	Accepted	0.08 sec, 30.36 MB

Question **4**

Correct

Mark 100.00 out of 100.00

Time limit	1 s
Memory limit	64 MB

Polymorphism merupakan konsep dimana sebuah subclass dapat mendefinisikan sifatnya sendiri dan dapat menggunakan fungsionalitas yang sama dari parent classnya. Pada soal ini, akan diuji pemahaman kalian terkait polymorphism

Pada nomor ini, terdapat sebuah parent class yang dapat dilihat pada [Car.java](#), dimana:

- Merupakan parent class
- Memiliki konstruktor berupa numberOfWheels dan engineCapacity
- Memiliki getter untuk masing-masing atributnya
- Mengeluarkan suara "Ngengggg"
- Mengeluarkan klakson "Din"
- Dapat mendeskripsikan dirinya dengan format "Mobil ini memiliki a roda dengan kapasitas mesin b cc", dimana a adalah numberOfWheels dan b adalah engineCapacity

Praktikan akan diminta untuk melengkapi 2 buah file, yaitu Bus.java dan Perari.java, dimana:

1. Bus.java

- Merupakan subclass dari Car
- Memiliki konstruktor yang sama dengan parentnya, dengan tambahan berupa passengerCapacity dan isUsedForLongTrip
- Memiliki getter untuk masing-masing atributnya
- Mengeluarkan suara "Cesss"
- Mengeluarkan klakson "Notnot"
- Dapat mendeskripsikan dirinya sesuai dengan keadaan:
 1. Apabila isUsedForLongTrip true, maka mengembalikan: "Mobil ini memiliki a roda dengan kapasitas mesin b cc, memiliki kapasitas penumpang c orang, dan digunakan untuk perjalanan jauh", dimana a adalah numberOfWheels, b adalah engineCapacity dan c adalah passengerCapacity
 2. Apabila isUsedForLongTrip false, maka mengembalikan: "Mobil ini memiliki a roda dengan kapasitas mesin b cc, memiliki kapasitas penumpang c orang, dan **tidak** digunakan untuk perjalanan jauh", dimana a adalah numberOfWheels, b adalah engineCapacity dan c adalah passengerCapacity

2. Perari.java

- Merupakan subclass dari Car
- Memiliki konstruktor yang sama dengan parentnya, dengan tambahan berupa averageSpeed dan isInsurance
- Memiliki getter untuk masing-masing atributnya
- Mengeluarkan suara "Brmmm"
- Mengeluarkan klakson "Siuuu"
- Dapat mendeskripsikan dirinya sesuai dengan keadaan:
 1. Apabila isInsurance true, maka mengembalikan: "Mobil ini memiliki a roda dengan kapasitas mesin b cc, memiliki kecepatan rata-rata c km/h, dan sedang dalam perlindungan asuransi", dimana a adalah numberOfWheels, b adalah engineCapacity dan c adalah averageSpeed
 2. Apabila isInsurance false, maka mengembalikan: "Mobil ini memiliki a roda dengan kapasitas mesin b cc, memiliki kecepatan rata-rata c km/h, dan sedang **tidak** dalam perlindungan asuransi", dimana a adalah numberOfWheels, b adalah engineCapacity dan c adalah averageSpeed

Submit zip file(nama dibebaskan) yang berisi [Bus.java](#) dan [Perari.java](#)

Java 8

 [car.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	25	Accepted	0.16 sec, 29.00 MB
2	25	Accepted	0.08 sec, 28.20 MB
3	25	Accepted	0.10 sec, 29.04 MB
4	25	Accepted	0.25 sec, 29.00 MB

Time limit	1 s
Memory limit	64 MB

Polymorphism - CarPrinter

Buatlah kelas `CarPrinter` yang berguna untuk mencetak informasi `Car` dan kelas turunannya (`Perari` dan `Bus`) dari beberapa mobil. Menggunakan kelas yang telah dibuat sebelumnya, kelas `CarPrinter` memiliki atribut `testCar` bertipe `Car` dan `carList` yang berbentuk `ArrayList of Car`. `CarPrinter` juga memiliki metode metode sebagai berikut:

1. `addCar`, yang menerima parameter `newCar` dengan tipe `Car`, menambahkannya ke dalam atribut `carList`, dan tidak mengembalikan apapun (*Hint: Penambahan elemen ke dalam sebuah objek ArrayList dapat menggunakan method `add(E element)` yang sudah tersedia*)
2. `getCarList`, yang mengembalikan atribut `carList`
3. `getTestCar`, yang mengembalikan atribut `testCar`
4. `setTestCar`, yang menerima parameter `carType` bertipe `String`, kemudian menjalankan aksi berikut berdasarkan isi parameter `carType` dan tidak mengembalikan apapun:
 - Apabila `carType` bernilai `Car`, maka atribut `testCar` akan menjadi sebuah objek baru dengan kelas `Car` yang memiliki parameter:
 - `numberOfWheels = 4`
 - `engineCapacity = 5`
 - Apabila `carType` bernilai `Bus`, maka atribut `testCar` akan menjadi sebuah objek baru dengan kelas `Bus` yang memiliki parameter:
 - `numberOfWheels = 6`
 - `engineCapacity = 10`
 - `passengerCapacity = 20`
 - `isUsedForLongTrip = true`
 - Apabila `carType` bernilai `Perari`, maka atribut `testCar` akan menjadi sebuah objek baru dengan kelas `Perari` yang memiliki parameter:
 - `numberOfWheels = 4`
 - `engineCapacity = 20`
 - `averageSpeed = 50`
 - `isInsuranced = false`
 - *Hint: Pengecekan kesamaan antara dua string sebaiknya menggunakan method `equals()` yang dimiliki kedua Object tersebut. Contoh: `"test".equals("test") == true`*
5. `printCarList`, yang akan melakukan iterasi terhadap isi `carList`, kemudian, untuk setiap elemennya, dia akan secara berturut-turut melakukan hal berikut:
 1. Tergantung *instance* dari elemennya:
 - Untuk kelas `Car` saja dan bukan *instance* dari childnya, maka cetak `"Car: "` lalu cetak output fungsi `sound()` dan `honk()` dengan semua kata dipisah oleh spasi diakhiri *new line*
 - Untuk kelas `Bus`, cetak `"Bus: "`, lalu cetak output fungsi `sound()` dan `honk()` dengan semua kata dipisah oleh spasi dan diakhiri *new line*
 - Untuk kelas `Perari`, cetak `"Perari: "`, lalu cetak output fungsi `sound()` dan `honk()` dengan semua kata dipisah oleh spasi diakhiri *new line*
 - Contoh untuk kelas `Bus`, maka output yang benar adalah : `Bus: Cesss Notnot`
 - *Hint: Pengecekan instance sebuah kelas dilakukan dengan menggunakan operator `instanceof` (Contoh: `"string" instanceof String == true`)*
 2. Memanggil metode `printDescription` diakhiri *new line*

Lengkapi file [CarPrinter.java](#)

Submit file `CarPrinter.java`

Java 8

 [CarPrinter.java](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	12	Accepted	0.07 sec, 28.07 MB
2	12	Accepted	0.07 sec, 28.03 MB
3	12	Accepted	0.08 sec, 28.66 MB
4	12	Accepted	0.08 sec, 27.98 MB
5	12	Accepted	0.08 sec, 29.03 MB
6	12	Accepted	0.08 sec, 28.02 MB
7	12	Accepted	0.08 sec, 28.97 MB
8	16	Accepted	0.08 sec, 28.47 MB

Time limit	1 s
Memory limit	64 MB

Diberikan *interface* sebagai berikut:

1. `Engine` (`Engine.java`)
2. `Tyre` (`Tyre.java`)

Dari *interface* `Engine`, terdapat dua kelas implementasi sebagai berikut:

1. `V6Engine` (`V6Engine.java`) yang mengeluarkan suara 'vroomvroom' yang bervariasi besaran kapasitas mesinnya dan dapat diberi turbo atau tidak. Apabila `isTurbo` true kembalikan nilai `engineCapacity` ditambah dengan 200. Apabila false kembalikan nilai `engineCapacity` saja.
2. `V8Engine` (`V8Engine.java`) yang mengeluarkan suara 'ngengngeng' yang bervariasi besaran kapasitas mesinnya dan dapat diberi supercharger atau tidak. Apabila `isSupercharged` true kembalikan nilai `engineCapacity` ditambah dengan 250. Apabila false kembalikan nilai `engineCapacity` saja.

Dari *interface* `Tyre`, terdapat dua kelas implementasi sebagai berikut:

1. `DonlupTyre` (`DonlupTyre.java`) yang bervariasi besaran tekanan ban dan lebar dari ban-nya. `DonlupTyre` dapat diberi nilai `isSlick` false atau true.
2. `ParelliTyre` (`ParelliTyre.java`) yang bervariasi besaran tekanan dan lebar dari ban-nya. `ParelliTyre` hanya memiliki nilai true dari method `isSlickTyre`.

Diberikan juga *class* `Car` yang telah dimodifikasi sebagai berikut:

1. Konstruktor menerima 3 parameter yaitu `numberOfWheels` bertipe `integer`, `engine` bertipe `Engine`, dan `tyre` bertipe `Tyre`.
2. Terdapat tambahan method pada kelas:

- `setEngine`
- `setTyre`
- `getNumberOfWheels`
- `getEngine`
- `getTyre`
- `sound`
- `isRacingCar`

Apabila sebuah mobil menggunakan ban yang memiliki nilai `isSlickTyre` true, maka akan dianggap sebagai mobil balap.

- `printDescription`

Apabila sebuah mobil memiliki nilai `isRacingCar` true, akan mengeluarkan output "Mobil balap ini memiliki a roda dengan kapasitas mesin b cc" sedangkan apabila `isRacingCar` false, akan mengeluarkan output "Mobil ini memiliki a roda dengan kapasitas mesin b cc". Dimana a adalah `numberOfWheels` dan b adalah nilai dari `engineCapacity` milik `Engine`.

Si Cello ingin membangun sebuah bengkel mobil untuk melakukan pengubahan ban dan mesin mobil, oleh karena itu dibuatlah *class* `CarWorkshop` (`CarWorkshop.java`) untuk mewujudkan keinginannya.

Class `CarWorkshop` memiliki dua method sebagai berikut:

1. `changeTyre`

Melakukan penggantian ban pada mobil.

2. `changeEngine`

Melakukan penggantian mesin pada mobil.

Lengkapilah metode-metode pada file `DonlupTyre.java`, `ParelliTyre.java`, `V8Engine.java`, `V6Engine.java`, `Car.java` dan `CarWorkshop.java` lalu submit sebagai zip file(nama dibebaskan).

Note: Perlu diingat bahwa kelas yang mengimplementasikan sebuah *interface* dapat dianggap sebagai *interface* tersebut atau apa yang disebut sebagai *polymorphism*.

Attachments: [attachments.zip](#)

Java 8

 [attach.zip](#)

Score: 100

Blackbox

Score: 100

Verdict: Accepted

Evaluator: Exact

No	Score	Verdict	Description
1	14	Accepted	0.07 sec, 27.80 MB
2	14	Accepted	0.08 sec, 28.07 MB
3	14	Accepted	0.07 sec, 28.39 MB
4	14	Accepted	0.08 sec, 27.88 MB
5	14	Accepted	0.08 sec, 28.30 MB
6	14	Accepted	0.08 sec, 30.44 MB
7	16	Accepted	0.08 sec, 28.80 MB