No. 1

NOTE

Buatlah seluruh atribut dengan *access modifier* `private` dan seluruh metode/konstruktor dengan *access modifier* `public`, **kecuali** jika diberi spesifikasi tambahan.

Buatlah sebuah *interface* bernama `HavePower` yang memiliki metode sebagai berikut:

1. `getRawPower`, tidak menerima argumen dan mengembalikan nilai bertipe `int`.
2. `isStrongerThan`, menerima argumen bertipe `HavePower` dan mengembalikan nilai bertipe `boolean`.

Buatlah sebuah *abstract class* bernama `Animal` yang mengimplementasi *interface* `HavePower` dan memiliki spesifikasi:

1. Atribut `rawPower` bertipe `int` dan `habitat` bertipe `String`.
2. Konstruktor yang menerima argumen `rawPower` bertipe `int` dan `habitat` bertipe `String`.
3. Metode `getHabitat` yang tidak menerima argumen dan mengembalikan `habitat` bertipe `String`.
4. Metode `getRawPower`, implementasi metode *interface* `HavePower` yang mengembailkan `rawPower` bertipe `int`.
5. Metode `isStrongerThan`, implementasi metode *interface* `HavePower` yang mengembalikan `true` apabila nilai atribut `rawPower` melebihi nilai metode `getRawPower` milik argumen, dan `false` jika sebaliknya.
6. Metode `fight` yang menerima argumen bertipe `Animal` dan akan menambahkan nilai atribut `rawPower` sebanyak atribut `rawPower` dari argumen bertipe `Animal` **apabila memenuhi kondisi** berikut:
   - Lebih kuat dari argumen bertipe `Animal` (nilai atribut `rawPower` melebihi nilai atribut `rawPower` milik argumen bertipe `Animal`)
   - Nilai atribut `habitat` sama dengan nilai atribut `habitat` milik argumen bertipe `Animal`.
7. Metode abstract `speak` yang tidak menerima argumen dan tidak mengembalikan keluaran.

Buatlah sebuah *class* bernama `Lion` yang meng-*extends class* `Animal` dan memiliki spesifikasi:

1. Konstruktor tanpa argumen, objek memiliki `rawPower` bernilai `150` dan `habitat` bernilai `"LAND"`.
2. Konstruktor dengan parameter `rawPower`, objek memiliki `rawPower` dari argumen dan `habitat` bernilai `"LAND"`.
3. Metode `speak` implementasi metode *abstract class* `Animal` yang akan menuliskan string `"GRAWRRRR!!! I'm a Lion!"` ke konsol diakhiri *new line*.

Buatlah sebuah *class* bernama `Tiger` yang meng-*extends class* `Animal` dan memiliki spesifikasi:

1. Konstruktor tanpa argumen, objek memiliki `rawPower` bernilai `100` dan `habitat` bernilai `"LAND"`.
2. Konstruktor dengan parameter `rawPower`, objek memiliki `rawPower` dari argumen dan `habitat` bernilai `"LAND"`.
3. Metode `speak` implementasi metode *abstract class* `Animal` yang akan menuliskan string `"RAWRRRR!!! I'm a Tiger!"` ke konsol diakhiri *new line*.

Buatlah `HavePower.java`, `Animal.java`, `Lion.java` dan `Tiger.java` tersebut, dan kumpulkan dalam satu ZIP *file* **(tanpa dibuat folder)** bernama `Animal.zip`.

Submit file `Animal.zip`

No.2

NOTE

Buatlah seluruh atribut dengan *access modifier* `private` dan seluruh metode/konstruktor dengan *access modifier* `public`, **kecuali** jika diberi spesifikasi tambahan.

Memanfaatkan *interface* `HavePower` dan kelas abstrak `Animal` yang didefinisikan pada soal sebelumnya, dan juga dengan memanfaatkan *interface* berikut

1. Carnivore.java
2. Herbivore.java

Buatlah kelas `Bear` yang meng-*extends* `Animal` dan mengimplementasikan *interface* `Carnivore` dan *interface* `Herbivore`.

Spesifikasi konstruktor dan metode dari kelas `Bear` adalah seperti berikut.

1. Terdapat 2 buah konstruktor:
   - Tidak menerima argumen apapun, objek memiliki `rawPower 125` dan `habitat "MOUNTAIN"`
   - Menerima sebuah argumen `rawPower`, objek memiliki nilai `rawPower` dari argumen dan `habitat "MOUNTAIN"`
2. `speak`, implementasi metode `speak` dari *abstract class* `Animal`, menuliskan string `"I'm a Good Bear."` ke konsol diakhiri *new line*.
3. `eatMeat`, implementasi metode `eatMeat` dari *interface* `Carnivore`, menuliskan string `"I'm eating a meat."` ke konsol diakhiri *new line*.
4. `hunt`, implementasi metode `hunt` dari *abstract class* `Carnivore`, melakukan `eatMeat` apabila `rawPower` lebih besar dari `animal` yang diburu.
5. `eatPlant`, implementasi metode `eatPlant` dari *interface* `Herbivore`, menuliskan string `"I'm eating a plant."` ke konsol diakhiri *new line*.

Submit file `Bear.java`

```
public interface Carnivore {
    public void eatMeat();

    public void hunt(Animal animal);
}

public interface Herbivore {
    public void eatPlant();
}
```

No.3

Buatlah sebuah *interface* bernama `Student` yang memiliki metode sebagai berikut:

1. `getStudentID`, tidak menerima argumen, mengembalikan nilai bertipe `long`
2. `getOverallGPA`, tidak menerima argumen, mengembalikan nilai bertipe `double`
3. `getGrade`, tidak menerima argumen, mengembalikan nilai bertipe `double`
4. `getMajor`, tidak menerima argumen, mengembalikan nilai bertipe `String`
5. `applyMajor`, menerima sebuah argumen bertipe `String`, mengembalikan nilai bertipe `boolean`
6. `applyScholarship`, menerima sebuah argumen bertipe `String`, tidak mengembalikan nilai apapun
7. `promoteGrade`, menerima sebuah argumen bertipe `double`, tidak mengembalikan nilai apapun
8. `payTuition`, menerima dua argumen bertipe `boolean` dan `String` berturut-turut, tidak mengembalikan nilai apapun

Submit file `Student.java` yang berisi *interface* Student dengan metode yang telah dijabarkan diatas!

Rifki K aida