# Firma przewozowa

Twoim zadaniem będzie utworzenie prostej aplikacji dla firmy transportowej, która odpowiada za przewóz zwierząt, osób oraz innego rodzaju przesyłek.

W pierwszej kolejności, należy utworzyć trzy klasy: `Animal` , `Person` oraz `Shipment` .

Klasa `Animal` powinna posiadać dwa pola prywatne:

- private String kind,
- private int weight.

Klasa `Person` powinna posiadać trzy pola prywatne:

- private String fristName,
- private String lastName,
- private boolean isRegularCustomer.

Natomiast klasa `Shipment` powinna posiadać trzy pola prywatne:

- private String prefix,

- private String serialNumber,

- private boolean isLarge.

Ponadto każdej z tych klas należy dodać publiczny konstruktor oraz gettery dla pól.

Następnym krokiem powinno być utworzenie interfejsu `TransportVisitor`. Będzie on miał deklarację trzech metod `visit` , które będą przyjmowały trzy konkretne typy: `Animal` , `Person` i `Shipment` .

Interfejs ten będą implementowały dwie klasy: `NameTransportVisitor` oraz `PriceTransportVisitor` .

Klasa `NameTransportVisitor` musi nadpisać trzy metody `visit` . Ich zadaniem będzie wypisanie imienia lub desygnacji przewożonej przesyłki, zwierzęcia lub osoby. Ciało metod niech stanowi metoda `System.out.println` , która ma wypisać:

- dla klasy `Animal` - typ zwierzęcia,

- dla klasy `Person` - imię oraz nazwisko,

- dla klasy `Shipment` - prefix oraz numer seryjny.

Klasa `PriceTransportVisitor` natomiast ma wyliczyć i wypisać na ekran cenę za kilometr, wyliczaną w zależności od tego, kto lub co będzie miało być transportowane:

- dla klasy `Animal` chcemy pobrać wagę zwierzęcia i pomnożyć tę wartość przez 0.2, i wypisać na ekran,

- dla klasy `Person` bazowa cena wynosić będzie 6. Należy sprawdzić czy dana osoba jest stałym klientem. Jeśli tak, to bazową cenę dzielimy przez 2 i wypisujemy na ekran:

```
    @Override
    public void visit(Person person) {
        int price = 6;
        if (person.isRegularCustomer()) {
            price = price / 2;
        }
        System.out.println("Price per kilometer for a person:
"
                + price + " PLN");
    }
```

- dla klasy `Shipment` cenę bazową ustawiamy na 2 i sprawdzamy czy przesyłka jest wielkogabarytowa. Jeśli tak, to cenę mnożymy przez 3 i wypisujemy w komunikacie.

Po implementacji klas visitorów, należy utworzyć interfejs `Transportable` . Będzie on miał deklarację jednej metody: `accept` , która będzie przyjmowała jako argument, klasę typu `TransportVisitor` .

Następnie interfejs ten należy zaimplementować w klasach `Animal` , `Person` oraz `Shipment` .

W demie naszej aplikacji tworzymy instancję powyższych klas "przewozowych", następnie tworzymy listę obiektów typu `Transportable` i dodajemy do niej obiekty tych klas:

```
        Animal animal = new Animal("dog", 30);
        Person person = new Person("Dawid", "Nowak", true);
        Shipment shipment = new Shipment("PL", "4325452", false);
        List<Transportable> transportableList = Arrays.asList(animal, person, shipment);
```

Następnie tworzymy obiekty typu `NameTransportVisitor` oraz `PriceTransportVisitor` .

Na koniec iterujemy po elementach listy i wywołujemy na nich metodę `accept` , która ma przyjmować obiekt typu `PriceTransportVisitor` , a następnie - w drugiej iteracji - ponownie wywołujemy metodę `accept` , która teraz ma przyjmować obiekt typu `NameTransportVisitor` .

W efekcie na ekranie powinniśmy otrzymać komunikat podobny do tego poniżej:

```
Price per kilometer for an animal: 6.0 PLN
Price per kilometer for a person: 3 PLN
Price per kilometer for a shipment: 2 PLN
----------------------
Animal kind: dog
Person name: Dawid Nowak
Shipment designation: PL-4325452
```