

# Java OOP. HW. Lesson 4

## Task 1 (Taxi)

Написать ядро приложения для учета поездок в такси с различными тарифами.

Поездка в такси (**Ride**) включает количество пассажиров (**passengers**), расстояние в километрах (**distance**), длительность в минутах (**duration**) и выбранный тариф (**tariff**).

Есть несколько типов тарифов:

1. Стандартный (**StandardTariff**):  $30 + 5 * \text{distance} + 2 * \text{duration}$ .

2. Семейный (**FamilyTariff**):  $50 + 20 * \text{distance} / \text{passengers}$ .

Система должна быть гибкой для добавления новых типов тарифов.

Реализовать **RidesHistory**, содержащий множество поездок и умеющий подсчитывать суммарную стоимость всех поездок в соответствии с заданными тарифами.

Написать **TaxiRunner** для тестирования функционала предыдущих классов.

### Шаблоны классов:

```
// file Ride.java
public class Ride {

    private final int passengers;
    private final int distance;
    private final int duration;
    private final Tariff tariff;

    public Ride(int passengers, int distance, int duration, Tariff tariff) { ... }

    public int getPassengers() { ... }
    public int getDistance() { ... }
    public int getDuration() { ... }

    public long getPrice() {
        return tariff.calculatePrice(this);
    }

    @Override
    public String toString() { ... }
}
```

```

// file Tariff.java
public interface Tariff {
    long calculatePrice(Ride ride);
}

// file StandardTariff.java
public class StandardTariff implements Tariff { ... }

// file FamilyTariff.java
public class FamilyTariff implements Tariff { ... }

// file RidesHistory.java
public class RidesHistory {

    private final List<Ride> rides = new ArrayList<>();

    public void addRide(Ride ride) { ... }
    public long getTotalPrice() { ... }

    @Override
    public String toString() { ... }
}

// file TaxiRunner.java
public class TaxiRunner {
    public static void main(String[] args) { ... }
}

```

---

## Task 2 (Messenger)

Написать ядро мессенджера с различными форматами текстового контента.

Сообщение ([Message](#)) содержит автора, дату публикации и текстовый контент.

Диалог ([Dialog](#)) содержит множество сообщений и предоставляет возможность печати всех сообщений в консоль.

Текстовый контент ([Text](#)) может быть нескольких видов:

1. Обычный текст ([PlainText](#)) - последовательность символов. Пример:

```

new PlainText("How are you?")      -> "How are you?"
new PlainText("I'm fine")         -> "I'm fine"

```

2. Эмодзи ([EmoticonText](#)). Пример:

```

new EmoticonText("winking-face") -> 😊
new EmoticonText("ghost")        -> 🕷️

```

3. ASCII-картишка (**PictureText**). Пример:

```
new PictureTest("like") ->
```

```
new PictureTest("heart") ->
```