

**Zadanie: klasa Maybe**

Zdefiniować klasę Maybe o następujących właściwościach.

Obiekty Maybe reprezentują kontenery, które mogą zawierać lub nie pojedynczą wartość. Motywacją do wprowadzenia takiej konstrukcji jest ułatwienie programowania w sytuacji, gdy zmienna może mieć wartość null, szczególnie kiedy wymagane jest jej dalsze bezpieczne przetwarzanie (na przykład za pomocą lambda-wyrażeń, oznaczających jakieś funkcje). Bezpieczne - to znaczy takie, które nie powoduje wyjątku NullPointerException.

Obiekty typu Maybe zawierają jakąś wartość lub są puste (nigdy nie powinny mieć wartości null). W klasie Maybe zdefiniować następujące metody:

- **Maybe.of(x)** - ta metoda statyczna zwraca obiekt Maybe, „opakowujący” wartość x, dowolnego typu referencyjnego.
- **void ifPresent(Consumer cons)** - jeżeli w obiekcie Maybe znajduje się wartość, wykonywana jest operacja cons z tą wartością jako argumentem, w przeciwnym razie - gdy obiekt Maybe jest pusty - nic się nie dzieje.
- **Maybe map(Function func)** - jeżeli w obiekcie jest wartość, wykonywana jest funkcja func z tą wartością jako argumentem i zwracany jest jej wynik „zapakowany” w nowy obiekt klasy Maybe (to opakowanie jest niezbędne, bo wynik mógłby być null, a tego chcemy uniknąć w ewentualnym dalszym przetwarzaniu; jeśli wynikiem funkcji jest null, zwracany jest pusty obiekt klasy Maybe).
- **T get()** zwraca zawartość obiektu Maybe, ale jeśli jest on pusty, powinna zgłosić wyjątek NoSuchElementException.
- **boolean isPresent()** - zwraca true jeśli w obiekcie Maybe zawarta jest wartość, a false - gdy jest on pusty
- **T orElse(T defVal)** - zwraca zawartość obiektu Maybe lub domyślną wartosć defVal, jeśli obiekt Maybe jest pusty.
- **Maybe filter(Predicate pred)** - zwraca to Maybe, jeśli spełniony jest warunek pred lub to Maybe jest puste; zwraca puste Maybe, jeśli warunek pred jest niespełniony.

Klasę Maybe przetestować na przykładzie następującej klasy Main::

```
public class Main {

    public static void test() {
        // Metoda of(...)
        String s = "aaa";
        Maybe<String> m1 = Maybe.of(s);
        System.out.println(m1);
        s = null;
        Maybe<String> m2 = Maybe.of(s);
        System.out.println(m2);

        // Metoda ifPresent(...)
        Integer num = null;
        Maybe<Integer> m4 = Maybe.of(num);
        // ZAMIAST
        if (num != null) System.out.println(num);
        // PISZEMY
        m4.ifPresent(n -> System.out.println(n));
        // A NAWET
        m4.ifPresent(System.out::println);

        Maybe<Integer> m5 = Maybe.of(10);
        m5.ifPresent(System.out::println);

        // Metoda map()
        Maybe<Integer> m6 = m5.map( n -> n +10 );
        System.out.println(m6);

        // Metoda get()
        System.out.println(m6.get());
        try {
            System.out.println(m4.get());
        } catch(Exception exc) {
            System.out.println(exc);
        }

        // Metoda orElse()
        // ZAMIAST
        String snum = null;
        if (num != null) snum = "Wartość wynosi: " + num;
        if (snum != null) System.out.println(snum);
        else System.out.println("Wartość niedostępna");

        //MOŻNA NAPISAC
        String res = Maybe.of(num).map(n -> "Wartość wynosi: "+n)
                                .orElse("Wartość niedostępna");
        System.out.println(res);

        // I filter(...)

        String txt = "Pies";
        String msg = "";

        //ZAMIAST
        if (txt != null && txt.length() > 0) {
            msg = txt;
        } else {
            msg = "Txt is null or empty";
        }

        //MOŻNA NAPISAC
        msg = Maybe.of(txt)
                    .filter(t -> t.length() > 0)
                    .orElse("Txt is null or empty");
        System.out.println(msg);
    }

    public static void main(String[] args) {
        test();
    }
}
// Wynik na konsoli:
/*
Maybe has value aaa
Maybe is empty
10
Maybe has value 20
20
java.util.NoSuchElementException: maybe is empty
Wartość niedostępna
Wartość niedostępna
Pies
*/
```