



# CONTROLLING REFERENCE ALIASES

## NUORODŲ PSEUDONIMŲ KONTROLIAVIMAS

ATLIKO: OLIVER MANTAS ALIŠAUSKAS

- Mano magistrinio tema - bendros kintamos būsenos valdymas
- Tikslai:
  - Surasti kalbos priemones padedančias programuotojui rašyti teisingas programas kai turime bendrą kintamą būseną
  - Surastas priemones įgyvendinti Starta kalboje
  - Priemonės turi užtikrinti teisingumą prieš kodo vykdymą

- Bendros kintamos būsenos pavyzdys

```
● ● ● Main.java  
var a = new Point(x: 1, y: 2);  
var b = a;  
b.x = 5; // a and b changes
```

- **@Immutable** objekto nuoroda ir laukai negali keistis
- **@Immutable** nuorodą galima priskirti tik **@Immutable** nuorodai

● ● ● Main.java

```
@Immutable var a = new Point(x: 1, y: 2);  
f(a);  
a.x = 5; // ERROR! 'a' is immutable
```

● ● ● Main.java

```
void f(@Immutable Point p) {  
    p = new Point(x: 3, y: 4); // ERROR! 'p' is immutable  
    p.x = 5; // ERROR! 'p' is immutable  
}
```

- **@ReadOnly** objekto nuorodos ir laukų negalima keisti, tačiau objektas gali keistis
- **@ReadOnly** nuorodą galima priskirti tik **@ReadOnly** nuorodai

● ● ● Main.java

```
var a = new Point(x: 1, y: 2);  
f(a); // 'a' guaranteed to not change  
a.x = 5; // OK! 'a' is mutable
```

● ● ● Main.java

```
void f(@ReadOnly Point p) {  
    p = new Point(x: 3, y: 4); // ERROR! 'p' is immutable  
    p.x = 5; // ERROR! 'p' is immutable  
}
```

- Keičiant **@CopyOnWrite** objekto laukus, automatiškai sukuriamas naujas objektas, jei egzistuoja 2 ar daugiau nuorodų į objektą

● ● ● Main.java

```
@CopyOnWrite var a = new Point(x: 1, y: 2);  
@CopyOnWrite var b = a;  
// 'b' = (1, 2); 'a' = (1, 2)  
b.x = 5; // OK! New object is created  
// 'b' = (5, 2); 'a' = (1, 2)  
b.x = 6; // OK! Existing object is modified  
// 'b' = (6, 2); 'a' = (1, 2)
```

# CONCURRENT READ EXCLUSIVE WRITE

- Turime 2 tipų nuorodas:
  - **@Write** – galime objektą rašyti ir skaityti
  - **@Read** – galime objektą tik skaityti
- Invariantas - vienu metu objektui teisinga tik 1 iš sąlygų:
  - 1 aktyvi **@Write** nuoroda
  - 1+ aktyvių **@Read** nuorodų
- Invariantui užtikrinti reikalingos 4 taisyklės
- Taisyklės užtikrina invariantą, jei nuorodų neišsaugom globaliai arba objektų laukuose

- Taisyklė 1 – Visos naujai sukurtos nuorodos yra aktyvios

● ● ● Main.java

```
@Write var a = new Point(x: 1, y: 2);  
// Active references: 'a'  
@Write var b = new Point(x: 1, y: 2);  
// Active references: 'a', 'b'  
...
```



- Taisyklė 2 – Jei sukurama nauja nuoroda iš senos **@Write** nuorodos, sena nuoroda laikinai tampa neaktyvi iki paskutinio nuorodos naudojimo, kuri rodo į senos nuorodos objektą

```
Main.java
{
    @Write var a = new Point(x: 1, y: 2); // 'a' is active
    @Read var b = a; // OK! 'a' becomes inactive
    @Read var c = b;
    int d = b.x + b.y;
    // 'c' still holds reference to 'a' object, so 'a' is not active yet
    int e = c.x + c.y;
    // 'a' becomes active again
}
```

- Taisyklė 3 - Nuoroda negali būti naudojama jei ji neaktyvi

● ● ● Main.java

```
{  
    @Write var a = new Point(x: 1, y: 2);  
    @Read var b = a; // OK! 'a' reference becomes inactive  
    a.x = 5; // ERROR! 'a' is inactive until last usage of 'b'  
    int c = b.x + b.y;  
    a.x = 5; // OK! 'a' is active and can be modified  
}
```

- Taisyklė 4 - **@Write** nuorodos negalima sukurti iš **@Read** nuorodos

● ● ● Main.java

```
@Read var a = new Point(x: 1, y: 2);  
@Write var b = a; // ERROR! 'a' is immutable
```

Ačiū už dėmesį