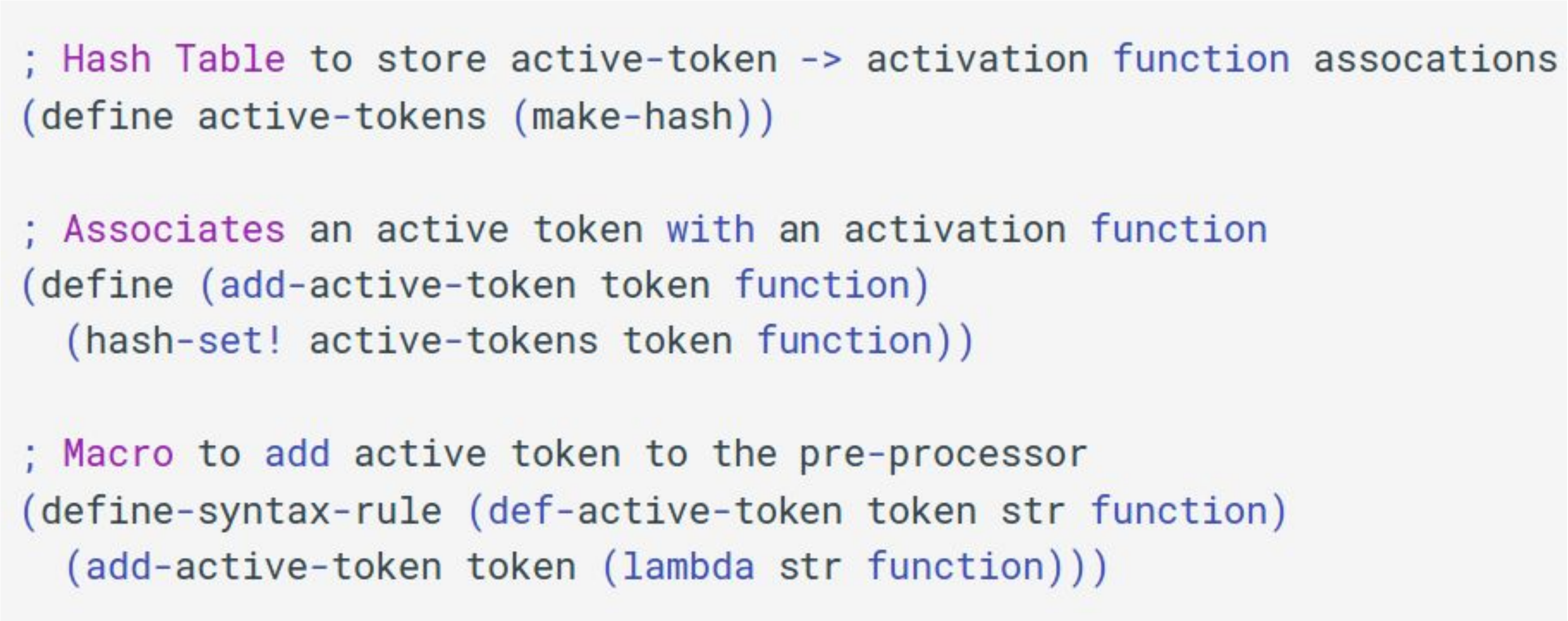
Pre-Processor in Racket

Daniel Correia, 80967 MEIC-A
Group 7

Developed Work

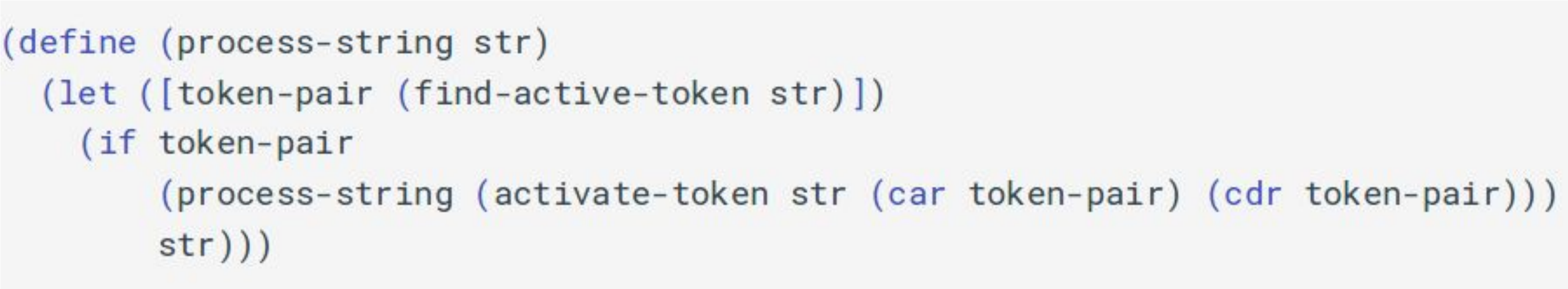
1. Basic Version requested (var, alias, #”..#{...}..”, macro)
2. Extensions
    a. Type Alias - Handle User Mistake
    b. @MetaToken
    c. Generate Getters+Setters
    d. Switch Expressions Syntax Sugar

Active Tokens Storage



```
; Hash Table to store active-token -> activation function assocations
(define active-tokens (make-hash))

; Associates an active token with an activation function
(define (add-active-token token function)
  (hash-set! active-tokens token function))

; Macro to add active token to the pre-processor
(define-syntax-rule (def-active-token token str function)
  (add-active-token token (lambda str function)))
```

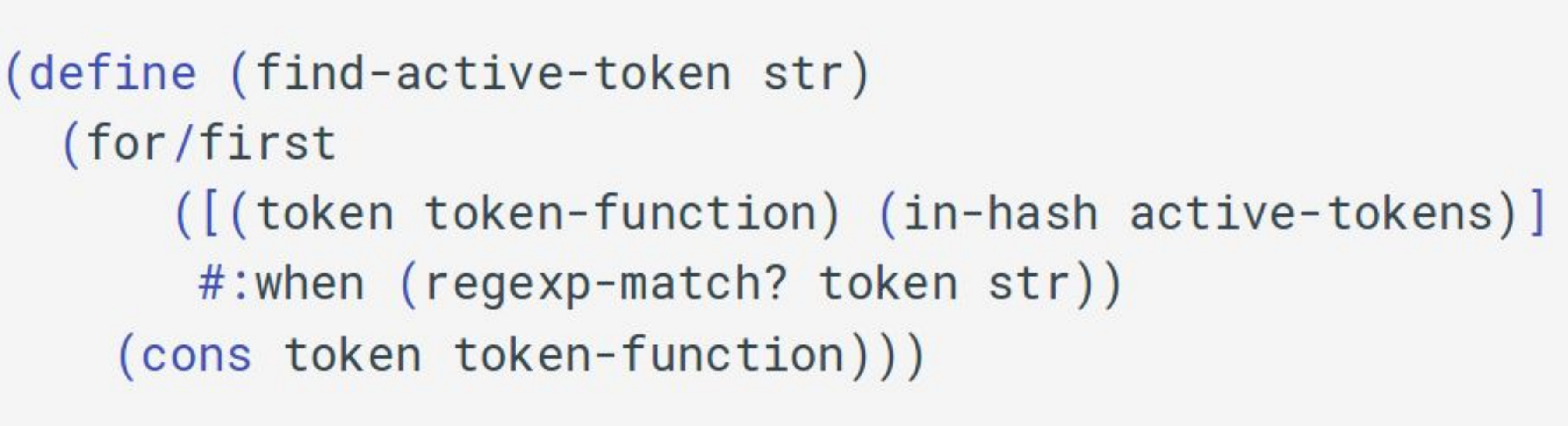
Pre-Processor Loop (Main)



```
(define (process-string str)
  (let ([token-pair (find-active-token str)])
    (if token-pair
        (process-string (activate-token str (car token-pair) (cdr token-pair)))
        str)))
```

Processes a string by recursively applying active token rules until no more active tokens are found in the string.

Find the first token match

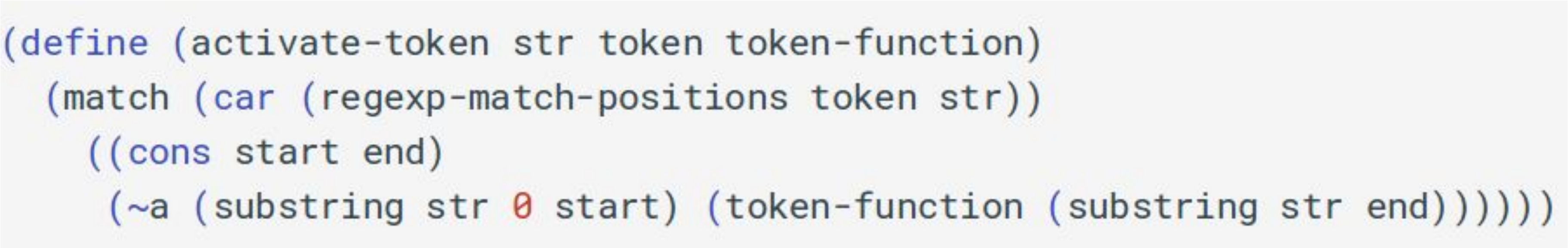


```
(define (find-active-token str)
  (for/first
      ([(token token-function) (in-hash active-tokens)]
       #:when (regexp-match? token str))
    (cons token token-function)))
```

Searches the string for an active token match.

On success, returns a pair with the matched token and activation function

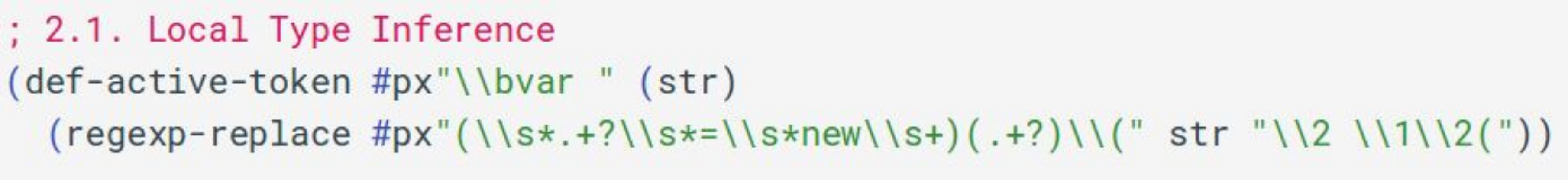
Activate token function



```
(define (activate-token str token token-function)
  (match (car (regexp-match-positions token str))
    ((cons start end)
     (~a (substring str 0 start) (token-function (substring str end))))))
```

Transform the substring after the matched token and join with non-transformed prefix string

Local Type Inference (var)



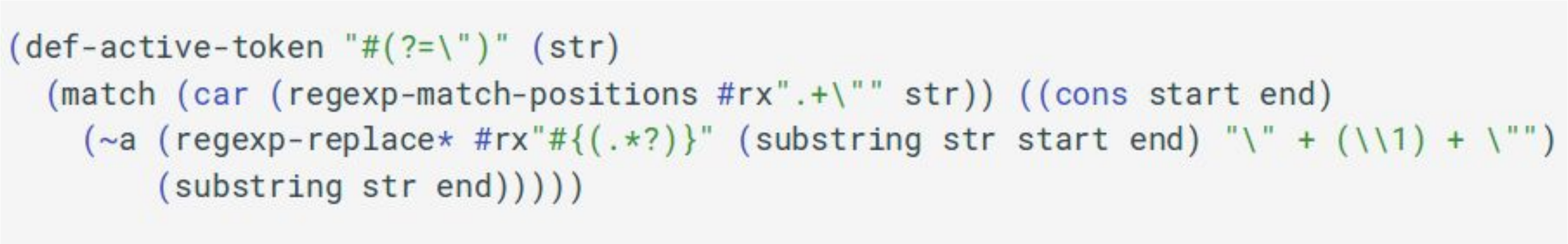
```
; 2.1. Local Type Inference
(def-active-token #px"\\bvar " (str)
  (regexp-replace #px"(\\s*.+?\\s*=\\s*new\\s+)(.+?)\\(" str "\\2 \\1\\2("))
```

Var Token
Need \\b prefix + space suffix due to partial matches like “xvary”, “varz”, etc.

Regex Capturing Groups

1. Everything between the token and the start of the “new” constructor type (\\1)
2. The constructor type (\\2)

String Interpolation (#”..#{...}..”)



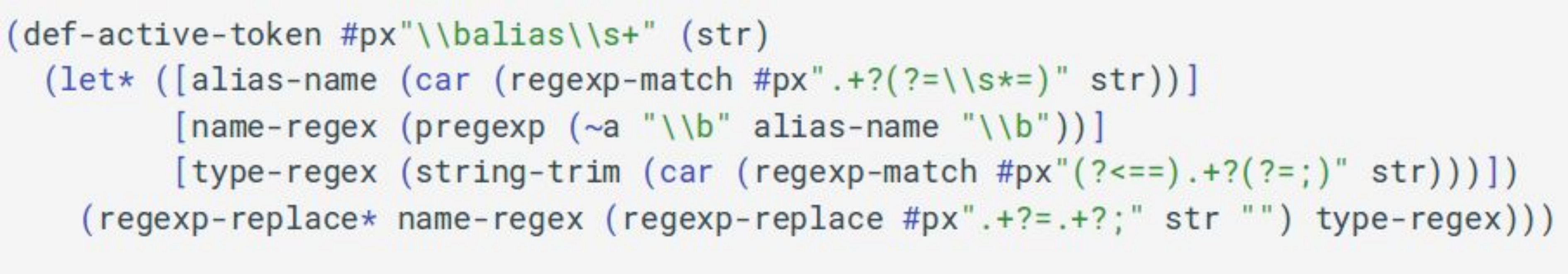
```
(def-active-token "#(?=\")" (str)
  (match (car (regexp-match-positions #rx".+\"" str)) ((cons start end)
    (~a (regexp-replace* #rx"#{(.*?)}" (substring str start end) "\" + (\\1) + \"")
        (substring str end)))))
```

Token Activation: #”a#{b}c” -> “a” + (b) + “c”

Make sure we only replace occurrences of #{...} blocks inside the string identified by the #” token.

Compose replacement result with rest of the text.

Type Aliases (alias)



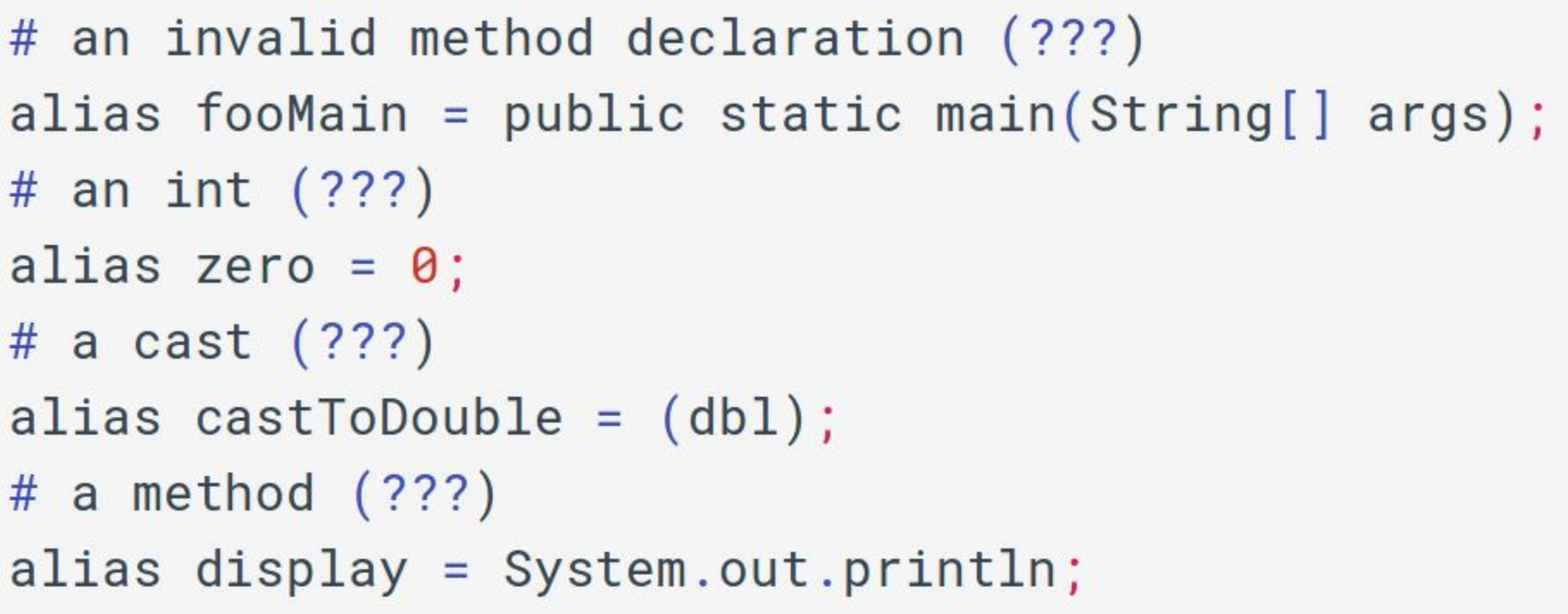
```
(def-active-token #px"\\balias\\s+" (str)
  (let* ([alias-name (car (regexp-match #px".+?(?=\\s*=)" str))]
         [name-regex (pregexp (~a "\\b" alias-name "\\b"))]
         [type-regex (string-trim (car (regexp-match #px"(?<==).+?(?=;)" str)))])
    (regexp-replace* name-regex (regexp-replace #px".+?=.+?;" str "") type-regex)))
```

Token Activation: alias name = type -> (empty) + side-effects

Clear alias definition from the string

Replace all occurrences of this alias in the string after the alias definition

“**Type**“ Aliases

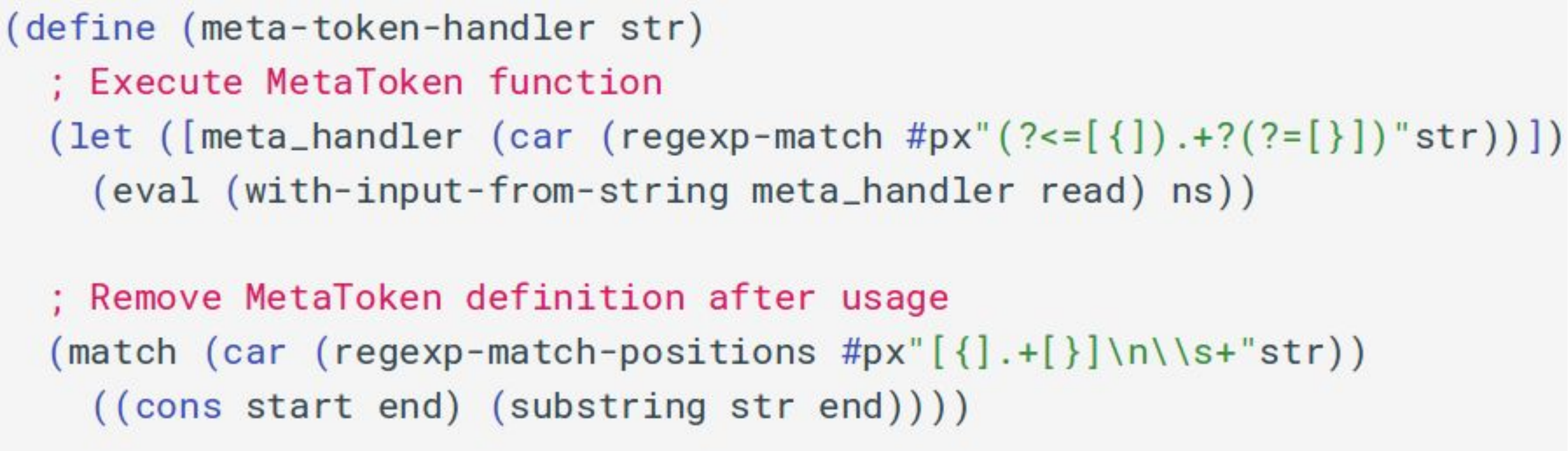


```
# an invalid method declaration (???)
alias fooMain = public static main(String[] args);
# an int (???)
alias zero = 0;
# a cast (???)
alias castToDouble = (dbl);
# a method (???)
alias display = System.out.println;
```

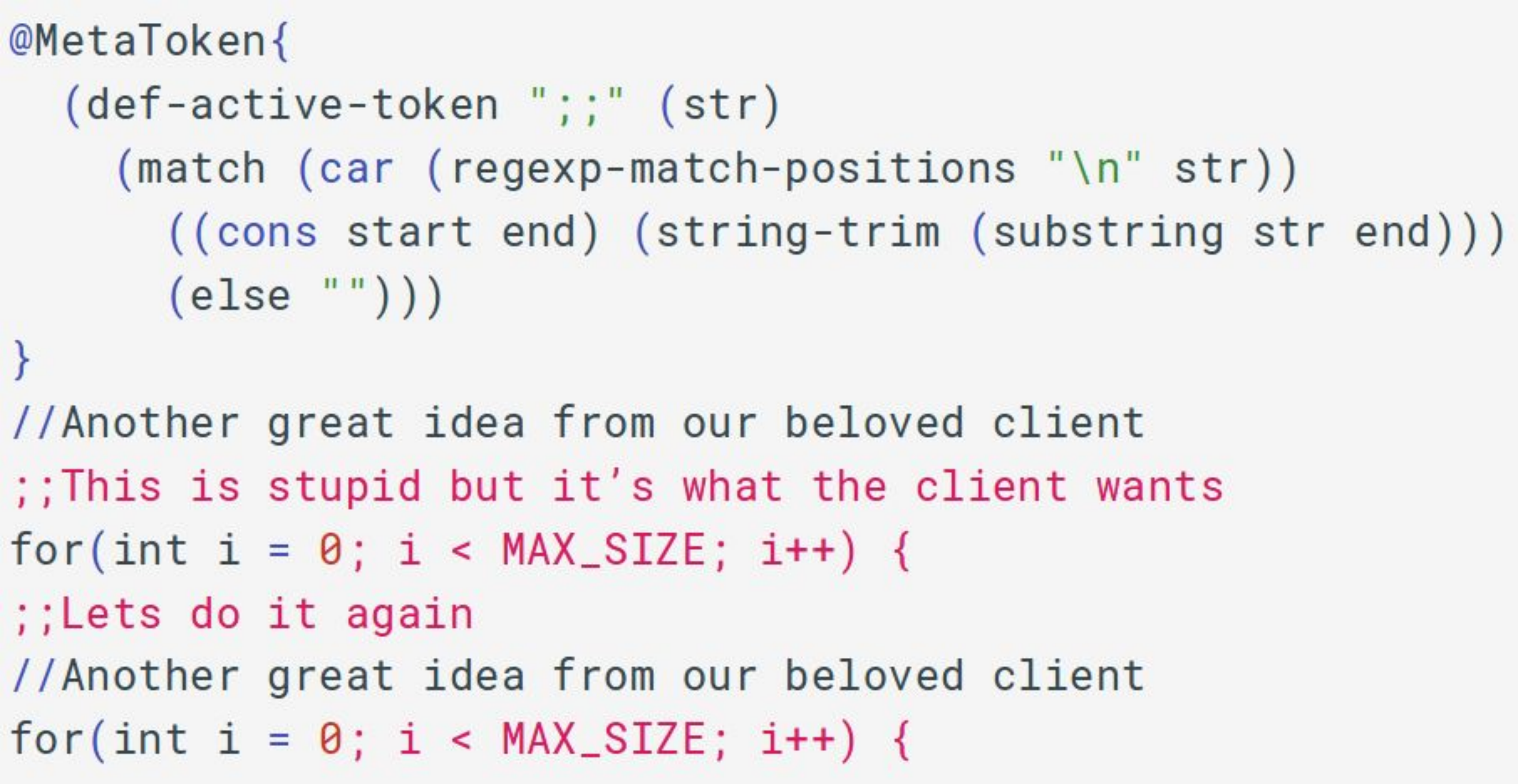
Relaxed/Strict Input Validation

Pre-Processor Extensions

@MetaToken



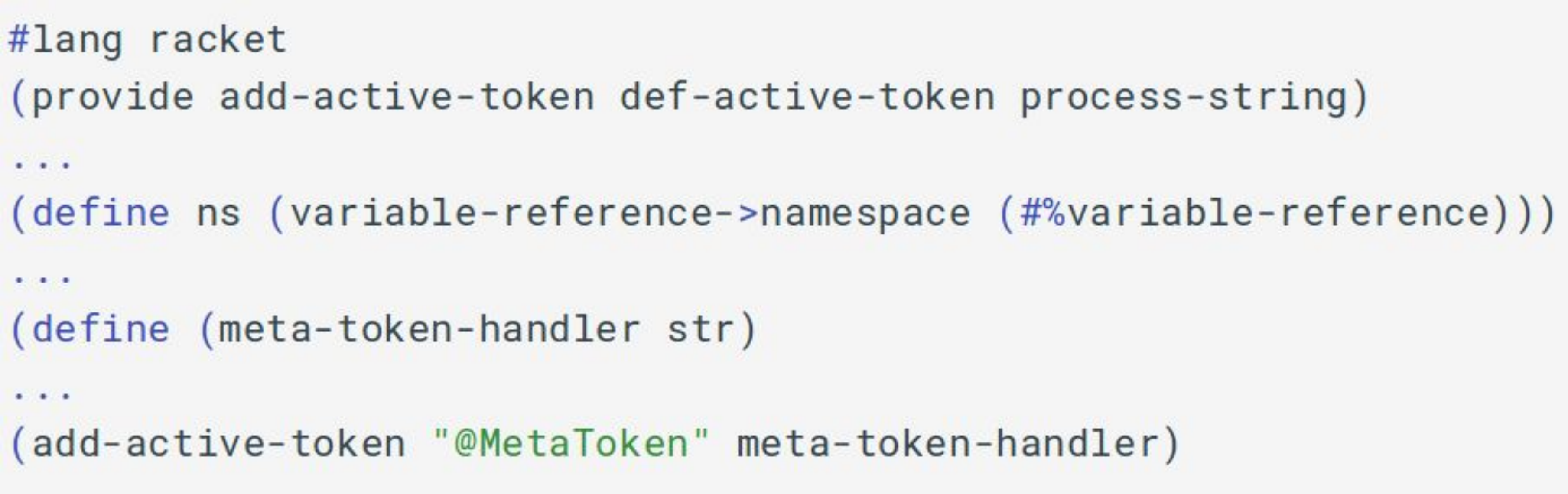
```
(define (meta-token-handler str)
  ; Execute MetaToken function
  (let ([meta_handler (car (regexp-match #px"(?<=[{]).+?(?=[}])"str))])
    (eval (with-input-from-string meta_handler read) ns))

  ; Remove MetaToken definition after usage
  (match (car (regexp-match-positions #px"[{].+[}]\n\\s+"str))
    ((cons start end) (substring str end))))
```



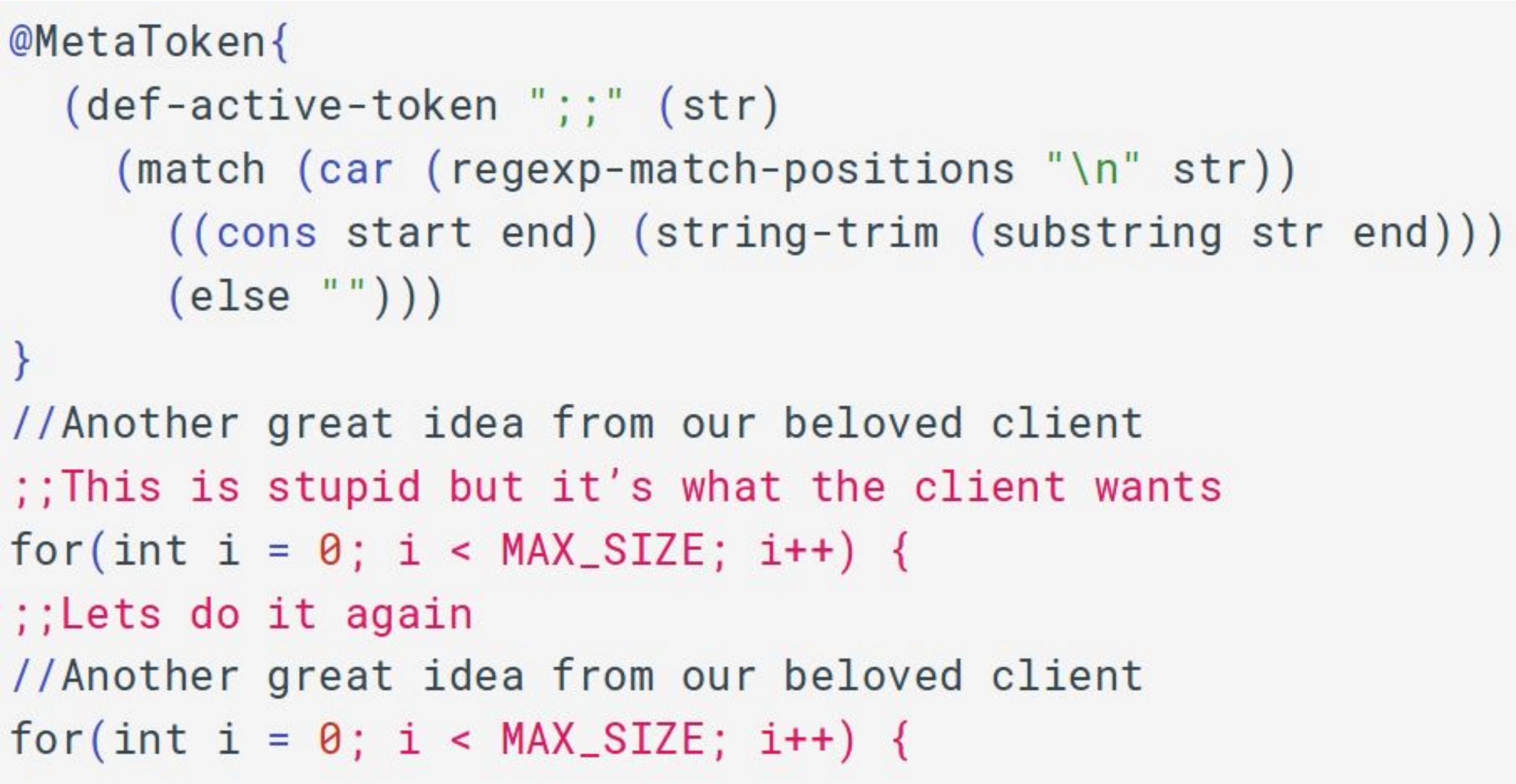
```
@MetaToken{
  (def-active-token ";;" (str)
    (match (car (regexp-match-positions "\n" str))
      ((cons start end) (string-trim (substring str end)))
      (else "")))
}
//Another great idea from our beloved client
;;This is stupid but it’s what the client wants
for(int i = 0; i < MAX_SIZE; i++) {
;;Lets do it again
//Another great idea from our beloved client
for(int i = 0; i < MAX_SIZE; i++) {
```

@MetaToken Example

@MetaToken Namespace Magic

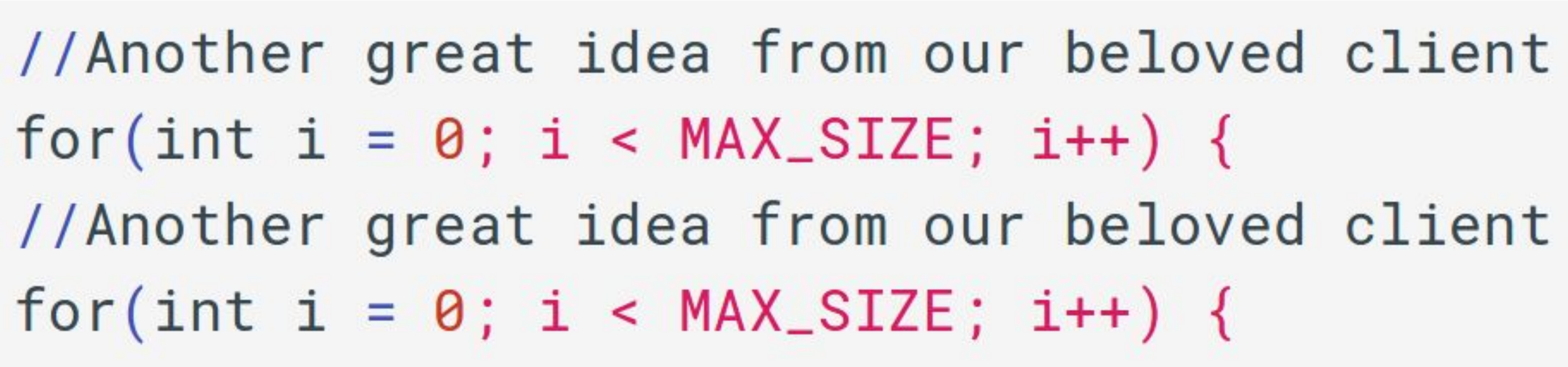


```
#lang racket
(provide add-active-token def-active-token process-string)
...
(define ns (variable-reference->namespace (#%variable-reference)))
...
(define (meta-token-handler str)
...
(add-active-token "@MetaToken" meta-token-handler)
```



```
@MetaToken{
  (def-active-token ";;" (str)
    (match (car (regexp-match-positions "\n" str))
      ((cons start end) (string-trim (substring str end)))
      (else "")))
}
//Another great idea from our beloved client
;;This is stupid but it’s what the client wants
for(int i = 0; i < MAX_SIZE; i++) {
;;Lets do it again
//Another great idea from our beloved client
for(int i = 0; i < MAX_SIZE; i++) {
```

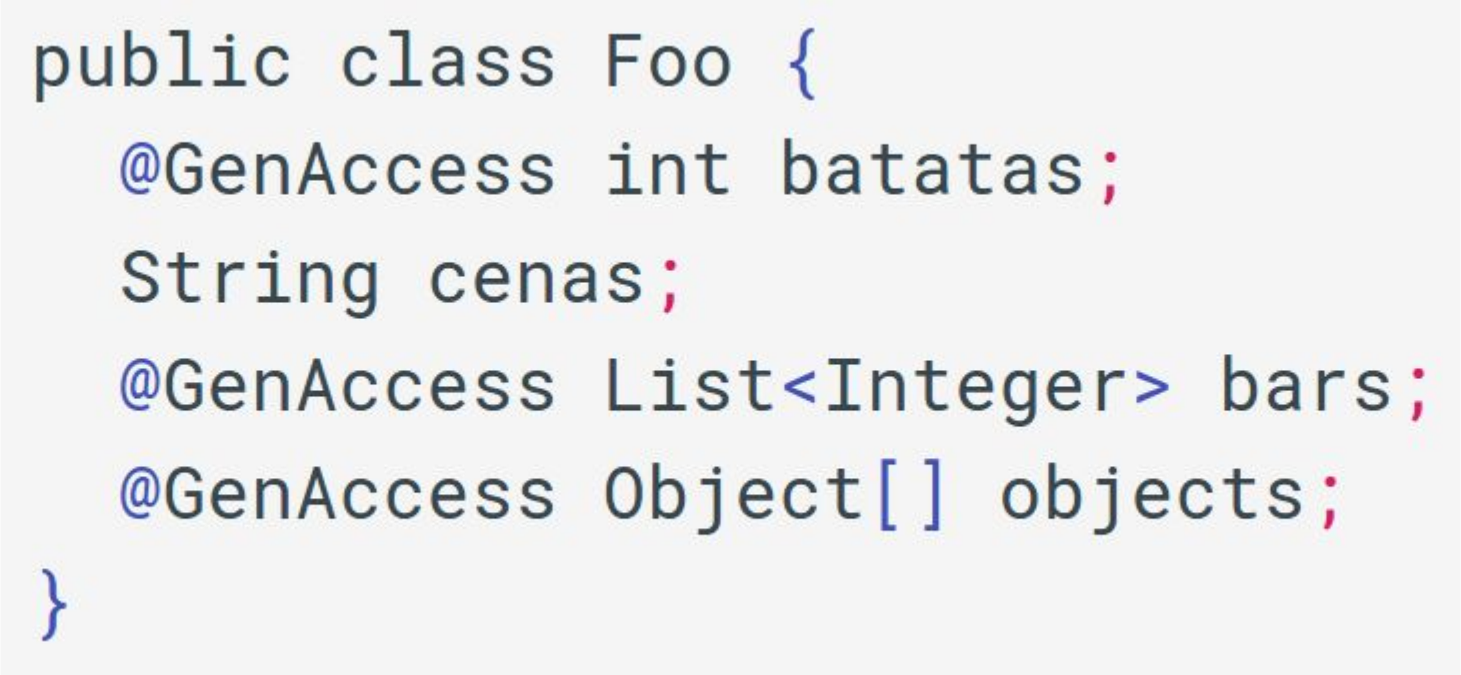
@MetaToken Example



```
//Another great idea from our beloved client
for(int i = 0; i < MAX_SIZE; i++) {
//Another great idea from our beloved client
for(int i = 0; i < MAX_SIZE; i++) {
```

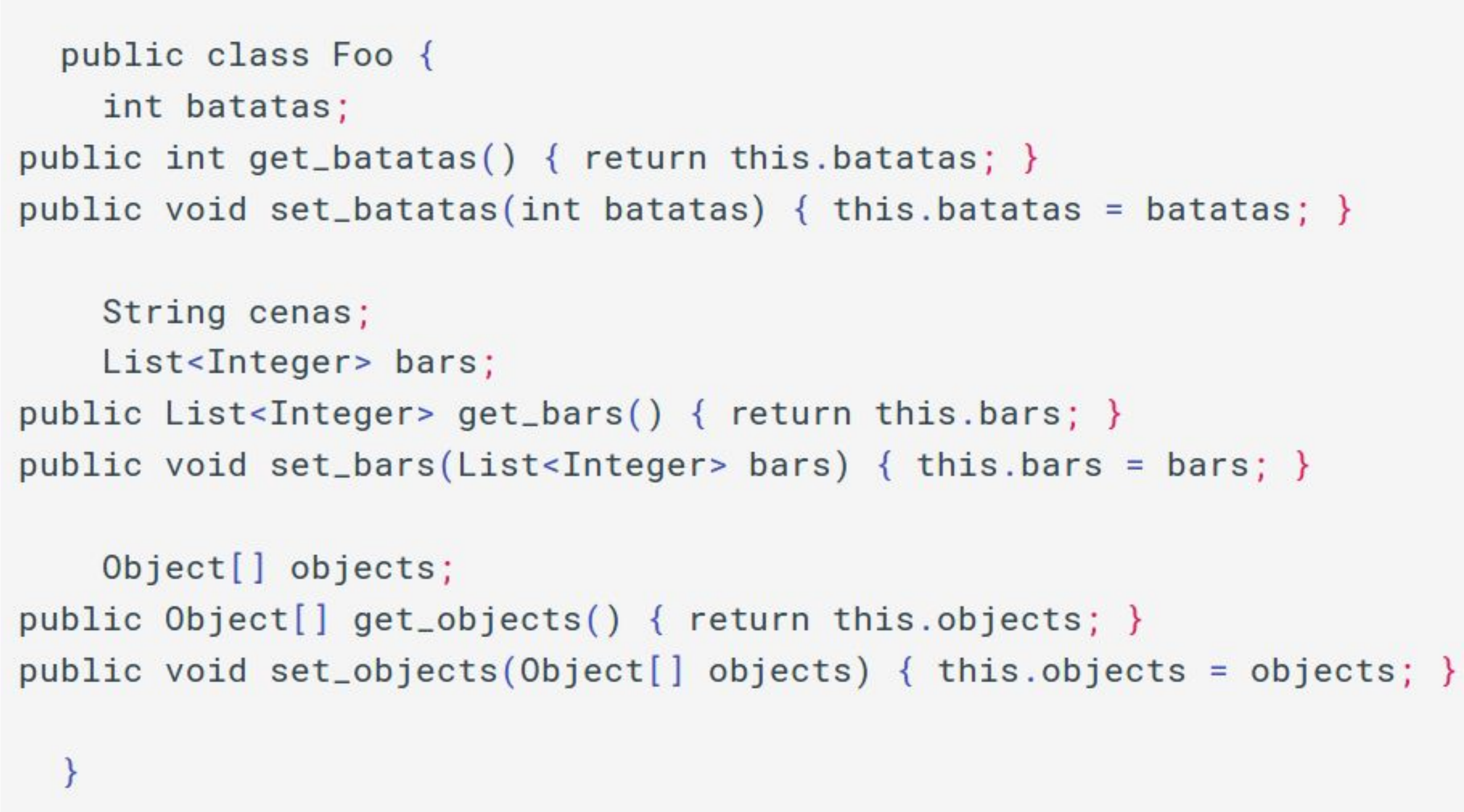
@MetaToken Example

@GenAccess



```
public class Foo {
  @GenAccess int batatas;
  String cenas;
  @GenAccess List<Integer> bars;
  @GenAccess Object[] objects;
}
```

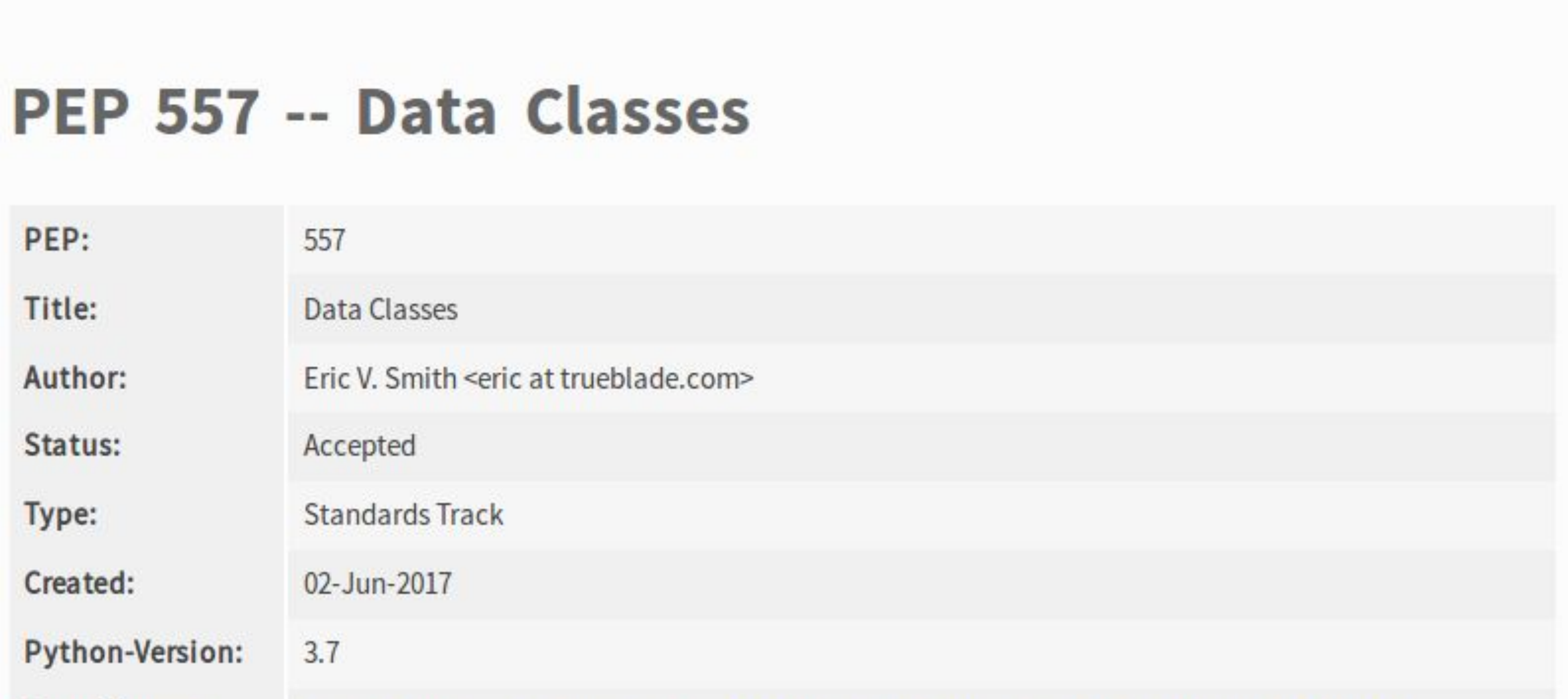
@GenAccess Example



```
  public class Foo {
    int batatas;
public int get_batatas() { return this.batatas; }
public void set_batatas(int batatas) { this.batatas = batatas; }

    String cenas;
    List<Integer> bars;
public List<Integer> get_bars() { return this.bars; }
public void set_bars(List<Integer> bars) { this.bars = bars; }

    Object[] objects;
public Object[] get_objects() { return this.objects; }
public void set_objects(Object[] objects) { this.objects = objects; }

  }
```

@GenAccess Example

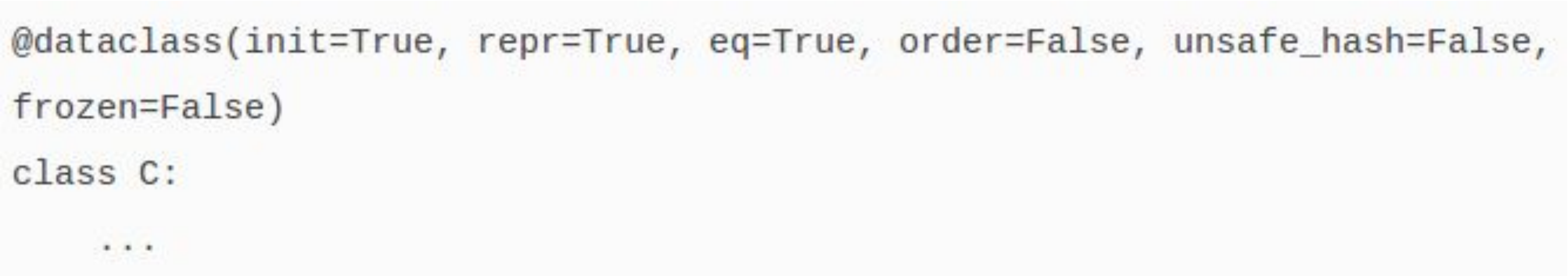
@DataClass



PEP 557 -- Data Classes

PEP:	557
Title:	Data Classes
Author:	Eric V. Smith <eric at trueblade.com>
Status:	Accepted
Type:	Standards Track
Created:	02-Jun-2017
Python-Version:	3.7

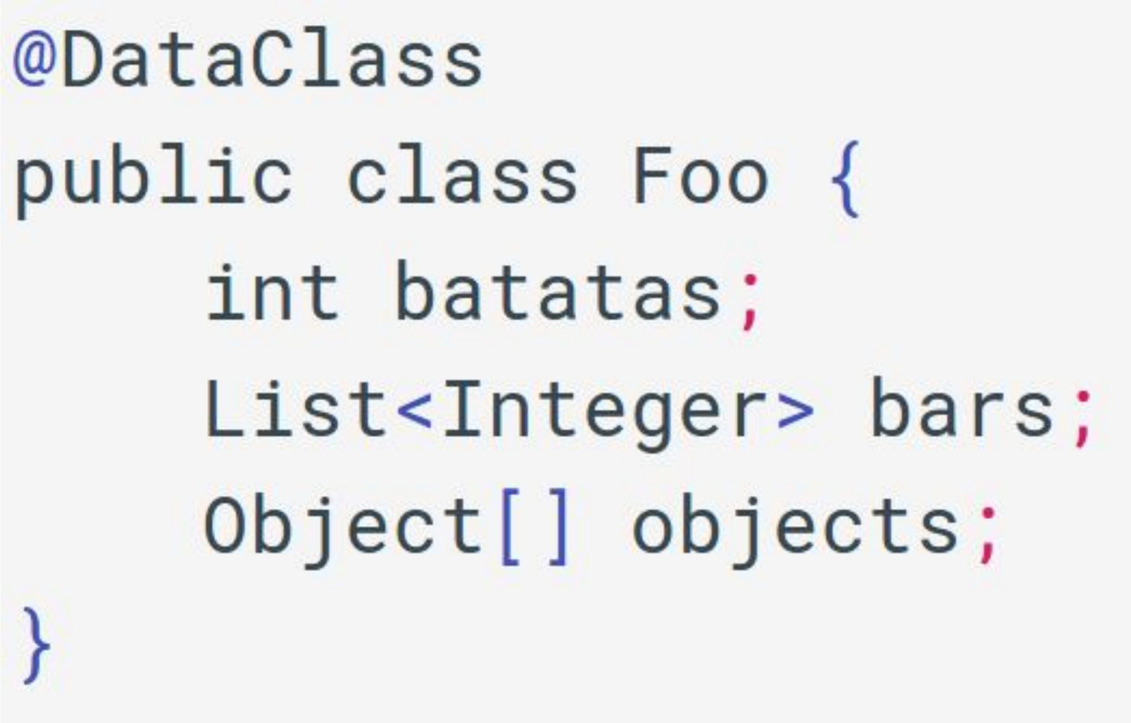
@DataClass



```
@dataclass(init=True, repr=True, eq=True, order=False, unsafe_hash=False,
frozen=False)
class C:
    ...
```

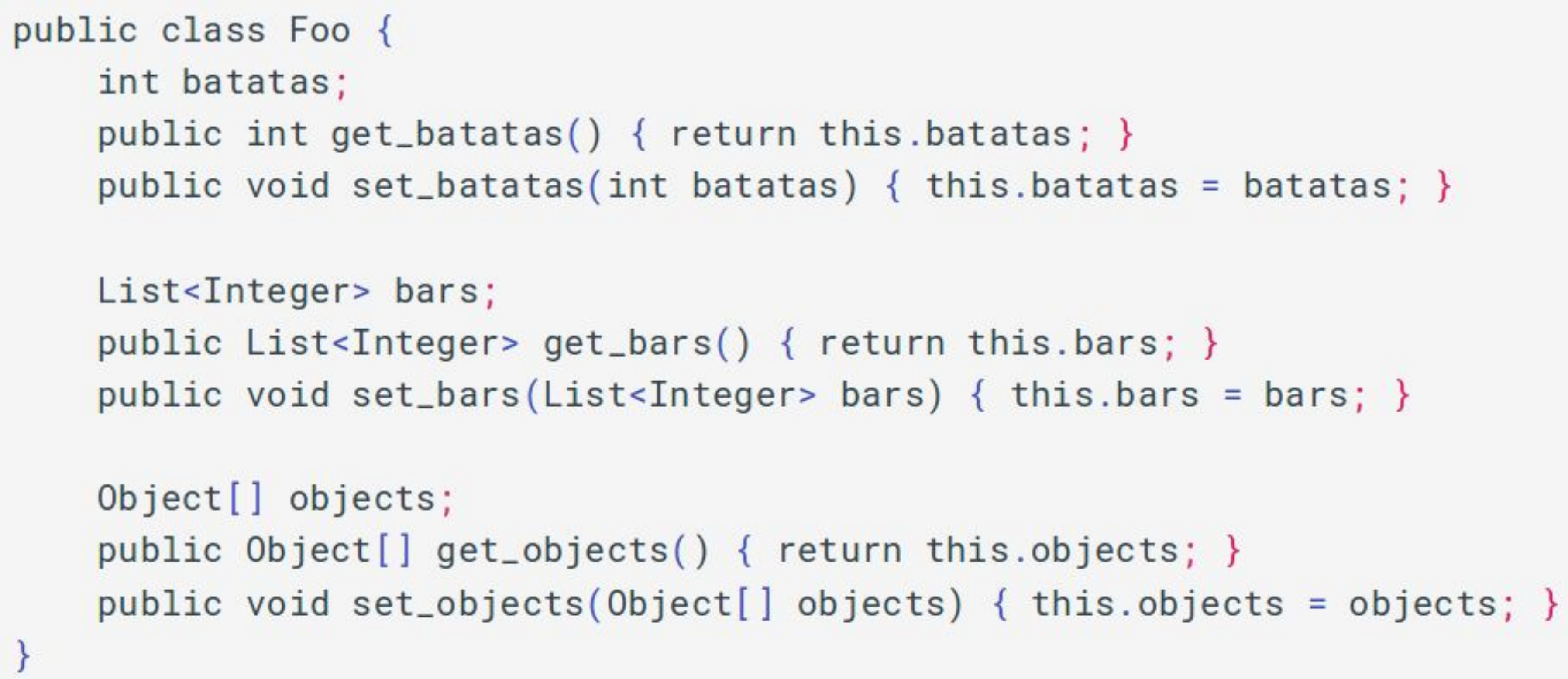
Python Data Classes are much more complex than my implementation (which only generates getters+setters)

Python allows the generation of comparators, class instance representation (similar to “toString”), stronger type checking, etc...



```java
@DataClass
public class Foo {
    int batatas;
    List<Integer> bars;
    Object[] objects;
}
```

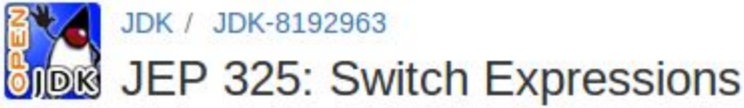
@DataClass Example



```
public class Foo {
    int batatas;
    public int get_batatas() { return this.batatas; }
    public void set_batatas(int batatas) { this.batatas = batatas; }

    List<Integer> bars;
    public List<Integer> get_bars() { return this.bars; }
    public void set_bars(List<Integer> bars) { this.bars = bars; }

    Object[] objects;
    public Object[] get_objects() { return this.objects; }
    public void set_objects(Object[] objects) { this.objects = objects; }
}
```

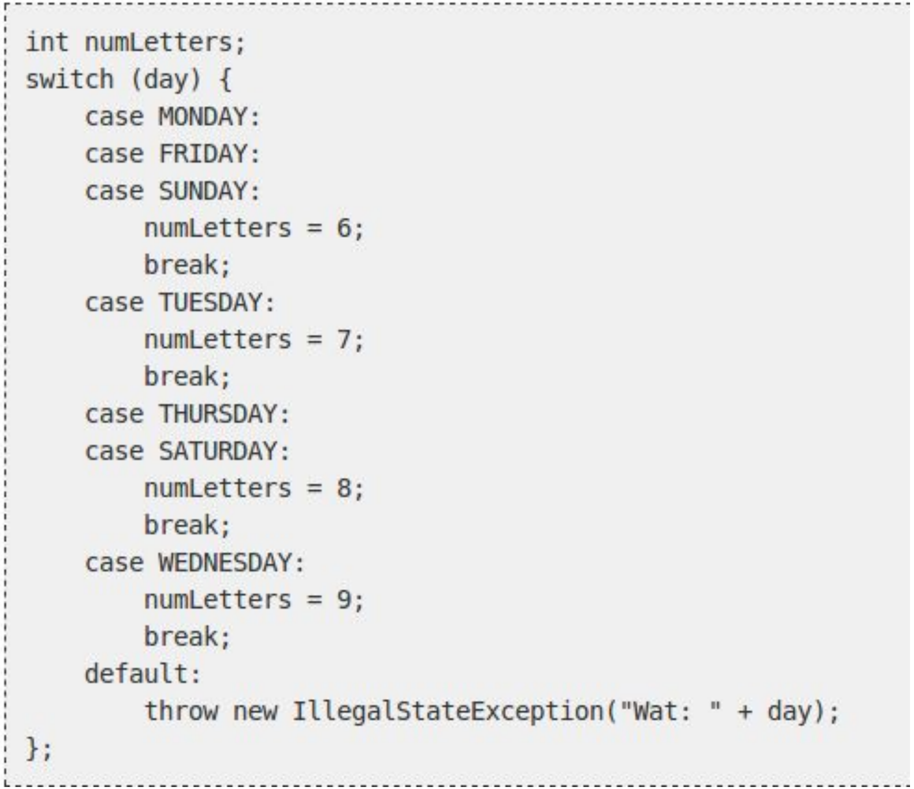
@DataClass Example

Switch Expressions

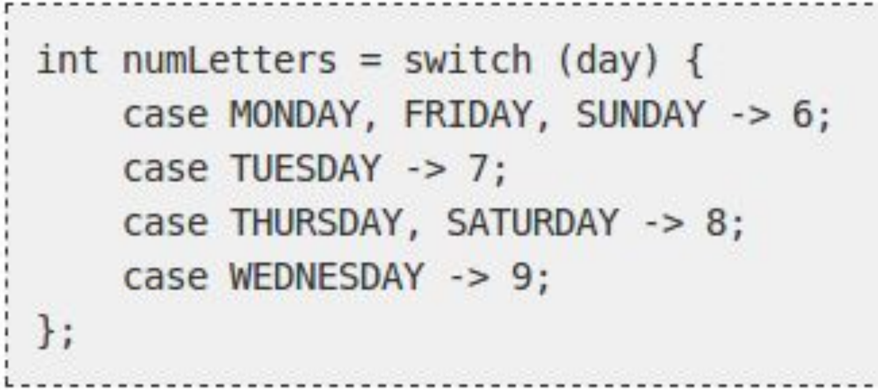


JDK / JDK-8192963

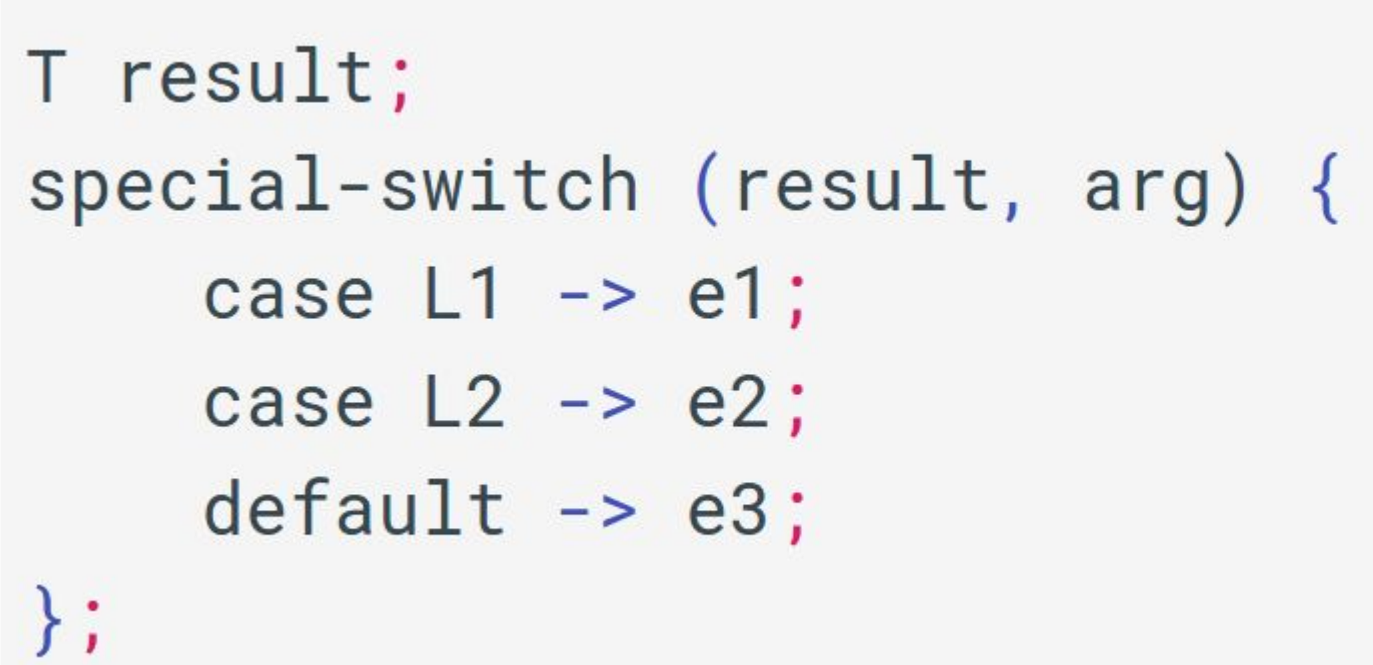
JEP 325: Switch Expressions



```
int numLetters;
switch (day) {
    case MONDAY:
    case FRIDAY:
    case SUNDAY:
        numLetters = 6;
        break;
    case TUESDAY:
        numLetters = 7;
        break;
    case THURSDAY:
    case SATURDAY:
        numLetters = 8;
        break;
    case WEDNESDAY:
        numLetters = 9;
        break;
    default:
        throw new IllegalStateException("Wat: " + day);
};
```



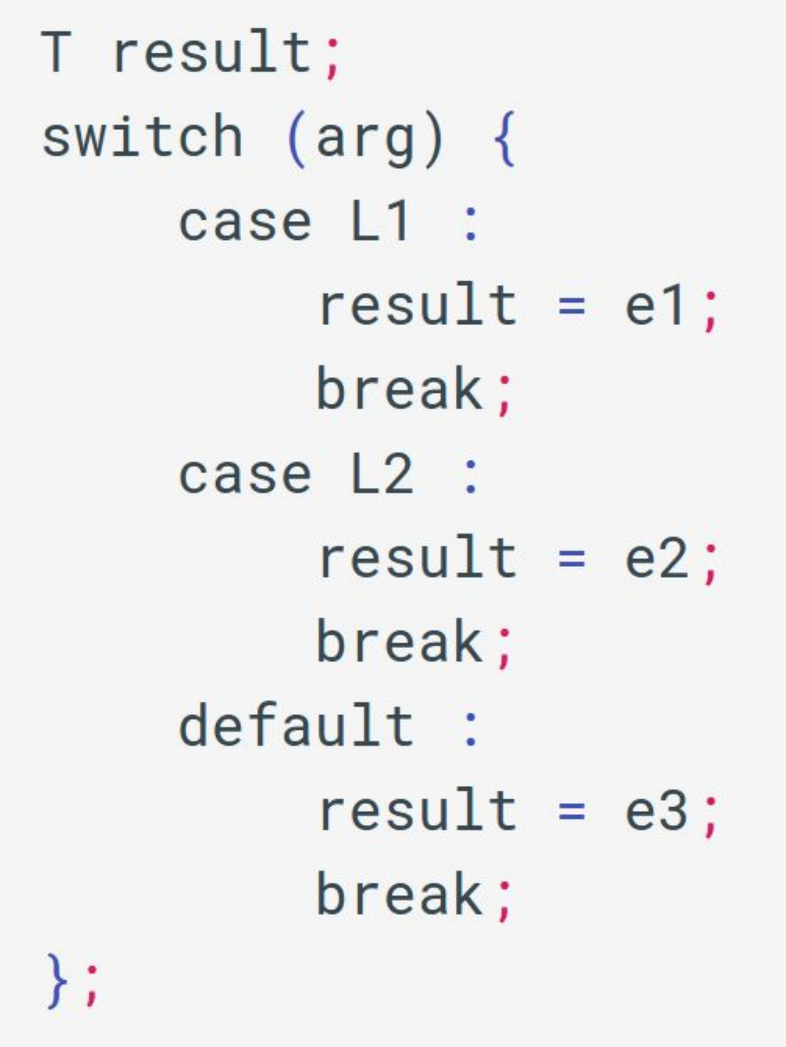
```
int numLetters = switch (day) {
    case MONDAY, FRIDAY, SUNDAY -> 6;
    case TUESDAY -> 7;
    case THURSDAY, SATURDAY -> 8;
    case WEDNESDAY -> 9;
};
```



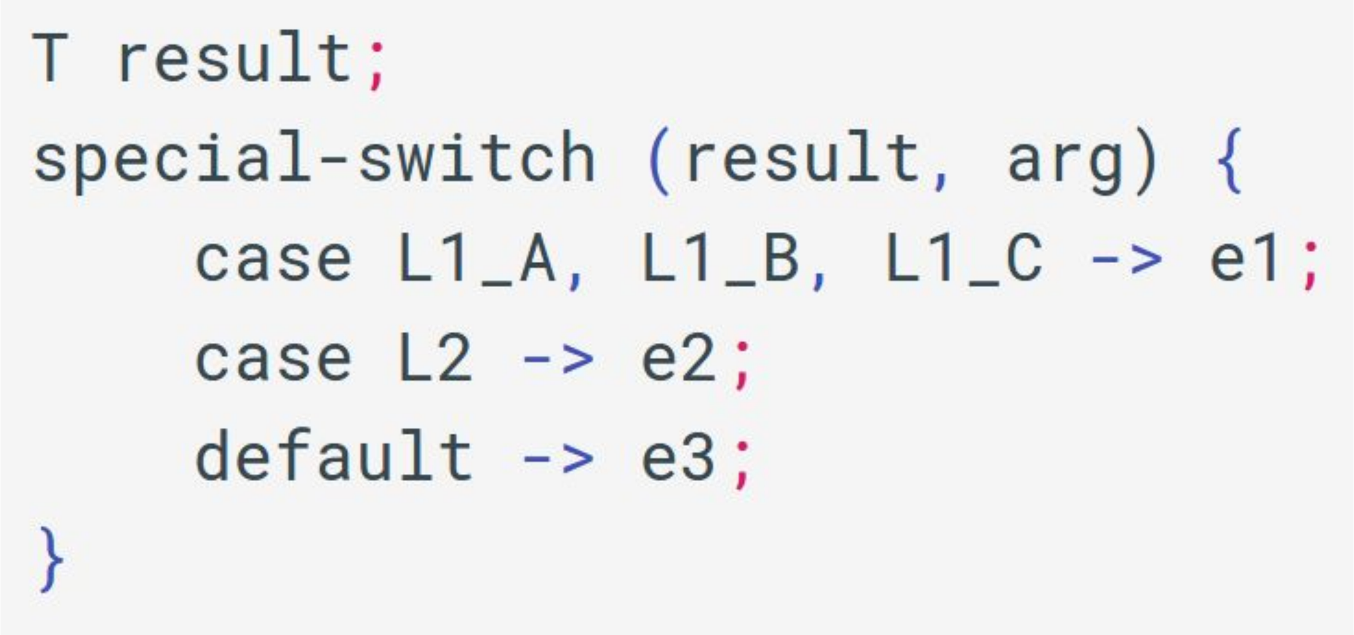
```
T result;
special-switch (result, arg) {
    case L1 -> e1;
    case L2 -> e2;
    default -> e3;
};
```

Basic Switch Example

Basic Switch Example

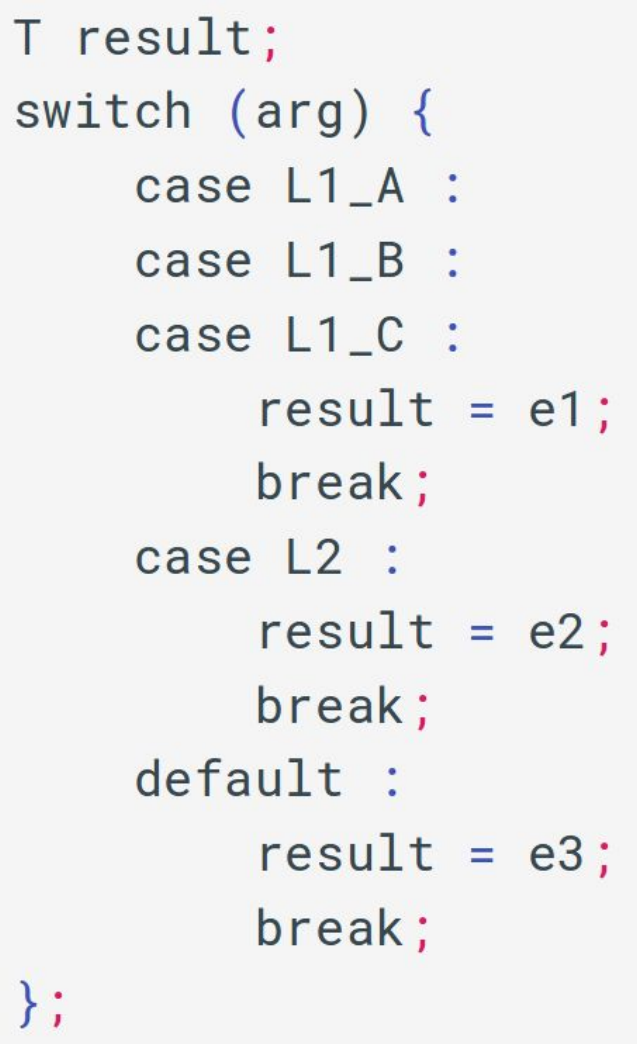


```
T result;
switch (arg) {
    case L1 :
        result = e1;
        break;
    case L2 :
        result = e2;
        break;
    default :
        result = e3;
        break;
};
```



```
T result;
special-switch (result, arg) {
    case L1_A, L1_B, L1_C -> e1;
    case L2 -> e2;
    default -> e3;
}
```

Cascade Switch Example



```
T result;
switch (arg) {
    case L1_A :
    case L1_B :
    case L1_C :
        result = e1;
        break;
    case L2 :
        result = e2;
        break;
    default :
        result = e3;
        break;
};
```

Cascade Switch Example

Questions?