Dataclasses

Tristan Crolard

Department of Computer Science
CEDRIC lab / SYS team

tristan.crolard@cnam.fr

cedric.cnam.fr/sys/crolard

Records – dataclasses

- Dataclasses are built upon object-oriented features: they are defined as a special kind of classes, using the decorator `@dataclass`
- However, they behave like named tuples (not like objects): they are similar to records and structs from other programming languages.
- Pattern matching is also available for dataclasses.
- They can be used to define variants or recursive datatypes

Dataclass basic example – Point

```
from dataclasses import dataclass

@dataclass
class Point:
    x: int
    y: int

def display(point: Point) -> None:
    match point:
        case Point(x, y):
            print("x␣=", x, ";", "y␣=", y)
```

Dataclass variant example – Value

```
from dataclasses import dataclass

type Value = IntValue | BoolValue | StrValue

@dataclass
class IntValue:
    i: int

@dataclass
class BoolValue:
    b: bool

@dataclass
class StrValue:
    s: str
```

Dataclass variant example – pattern matching

```
def to_string(v: Value) -> str:
    match v:
        case IntValue(i):
            return str(i)
        case BoolValue(b):
            return str(b)
        case StrValue(s):
            return s
```

Dataclass recursive example – Tree

```
from dataclasses import dataclass

type Tree[A] = Leaf[A] | Node[A]

@dataclass
class Leaf[A]:
    value: A

@dataclass
class Node[A]:
    left: Tree[A]
    right: Tree[A]
```

Dataclass recursive example – pattern matching

```
def size[A](t: Tree[A]) -> int:
    match (t):
        case Leaf(_):
            return 1
        case Node(l, r):
            return size(l) + size(r)
```