Java's Built-in Data Definitions

- **int**

  1   5999   -10

- **double**

  1.1   5999.33   -10.01

- **boolean**

  true   false

- **String**

  "hello"   "See you later!"

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  Str  g name;
  doub
  Stri
  Snake                     le weight, String food) {
    thi
    thi
    thi
  }
}
```

`class`

class declaration

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double wei
  String foo
  Snake(Stri                 eight, String food) {
    this.na
    this.wei
    this.foo
  }
}
```

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

{

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String nam                String food) {
    this.name = na
    this.weight = weight;
    this.food = food;
  }
}
```

type

name ;

field

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class S
  Stri
  doubl
  Stri     food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

constructor

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double
  String
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

(

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

,

constructor arguments

Compound Data in Java

```
  ; A snake is
  ;   (make-snake sym num sym)
  (define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

)

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

{

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double                     ood) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

```
            this
.           name
=           name
;
```

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

Compound Data in Java

```
; A snake is
;   (make-snake sym num sym)
(define-struct snake (name weight food))
```

```
class Snake {
  String name;
  double weight;
  String food;
  Snake(String name, double weight, String food) {
    this.name = name;
    this.weight = weight;
    this.food = food;
  }
}
```

}

Instances of Compound Data Types

```
(make-snake 'Slinky 12 'rats)
(make-snake 'Slimey 5 'grass)
```

```
new Snake("Slinky", 12, "rats")
new Snake("Slimey", 5, "grass")
```

Instances of Compound Data Types

```
(make-snake 'Slinky 12 'rats)
(make-snake 'Slimey 5 'grass)
```

```
new Snake("Slinky", 12, "rats")
new Snake("Slimey", 5, "grass")
```

new

Instances of Compound Data Types

```
(make-snake 'Slinky 12 'rats)
(make-snake 'Slimey 5 'grass)
```

```
new Snake("Slinky", 12, "rats")
new Snake("Slimey", 5, "grass")
```

Instances of Compound Data Types

```
(make-snake 'Slinky 12 'rats)
(make-snake 'Slimey 5 'grass)
```

```
new Snake("Slinky", 12, "rats")
new Snake("Slimey", 5, "grass")
```

(

Instances of Compound Data Types

```
(make-snake 'Slinky 12 'rats)
(make-snake 'Slimey 5 'grass)
```

```
new Snake("Slinky", 12, "rats")
new Snake("Slimey", 5, "grass")
```

,

Instances of Compound Data Types

```
(make-snake 'Slinky 12 'rats)
(make-snake 'Slimey 5 'grass)
```

```
new Snake("Slinky", 12, "rats")
new Snake("Slimey", 5, "grass")
```

)

Armadillos

```
class Dillo {
  double weight;
  boolean alive;
  Dillo(double weight, boolean alive) {
    this.weight = weight;
    this.alive = alive;
  }
}

new Dillo(2, true)
new Dillo(3, false)
```

Posns

```
class Posn {
  int x;
  int y;
  Posn(int x, int y) {
   this.x = x;
   this.y = y;
  }
}

new Posn(0, 0)
new Posn(1, -2)
```

Ants

```
class Ant {
  double weight;
  Posn loc;
  Ant(double weight, Posn loc) {
    this.weight = weight;
    this.loc = loc;
  }
}

new Ant(0.0001, new Posn(0, 0))
new Ant(0.0002, new Posn(1, -2))
```

Data with Variants

```
  ; An animal is either
  ;   - snake
  ;   - dillo
  ;   - ant

interface IAnimal {
}

class Snake implements IAnimal {
 ... as before ...
}
class Dillo implements IAnimal {
 ... as before ...
}
class Ant implements IAnimal {
 ... as before ...
}
```

Data with Variants

```
  ; An animal is either
  ;    - snake
  ;    - dillo
  ;    - ant

interface IAnimal {
}

  ass Snake implements IAnimal {
  .. as before ...
}
class Dillo implements IAnimal {
 ... as before ...
}
class Ant implements IAnimal {
 ... as before ...
}
```

interface

Data with Variants

```
   ; An animal is either
   ;    - snake
   ;    - dillo
   ;    - ant

interface IAnimal {
}

class                    IAnimal {
 ...
}
class                    IAnimal {
 ... as before ...
}
class Ant implements IAnimal {
 ... as before ...
}
```

Data with Variants

```
  ; An animal is either
  ;
  ;
  ;

inter
}
```

```
implements

     {
```

```
class Snake implements IAnimal {
 ... as before ...
}
class Dillo implements IAnimal {
 ... as before ...
}
class Ant implements IAnimal {
 ... as before ...
}
```

Data with Variants

```
   ; An animal is either
   ;    - snake
   ;    - dillo
   ;    - ant

interface IAnimal {
}

class Snake implements IAnimal {
 ... as before ...
}
cl                   ts IAnimal {
 .
}
class Ant implements IAnimal {
 ... as before ...
}
```

Variants in Java

•

```
; A grade is either      ⇒     ; A grade is either
;   - false                    ;   - no-grade
;   - num                      ;   - num-grade

                               ; A no-grade is
                               ;   (make-no-grade)
                               (define-struct no-grade ())

                               ; A num-grade is
                               ;   (make-num-grade num)
                               (define-struct num-grade (n))
```

•