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

```java
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
        this
        this
    }
}

class declaration
- , %&
#(0-, .123}
Compound Data in Java

```java
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 weight, String food) {
        this.name = name;
        this.weight = weight;
        this.food = food;
    }
}

> (6) 6: (7 6) .0::=* &1'/?(type 6", name 6", ;/ .." 7(" 6 : (6' - field
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;
  }
}

@/#()@6": (#/)?(-#)@6"*"+,"*
  , 0"2",8@6'
#(#)@6"
constructor
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;
    }
}

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;
   }
}

A6", )}
Compound Data in Java

```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;
    }
}

D 6)+"*",," 0.(" #0"#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-scope 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

```scheme
(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 '#(#+
-.'#, " G & ℐ H
.7 1'"
```
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")

A6", (}
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")
```

"A6",+)1)&1="": ( #* 93,
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")
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)
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 {
}

interface Z ()
* # " +, -, 
? -# & (-, #

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 IAnimal {
    ...
}
class IAnimal {
    ...
    # as before ...
}
class Ant implements IAnimal {
    ...
    as before ...
}
Data with Variants

!"#$
; An animal is either
;
;
; F 6,2")#6") 1" )7 ( 6 &(-, #93) **-, 2
; " 6 &(-, #93) **-, 2
; implements #6", )#6" 

&&$

inter 2(. =-, 2 1", 0 "+ 

} 9"7 (" 

class Snake implements IAnimal { 
... as before ...
}
class Dillo implements IAnimal { 
... as before ...
}
class Ant implements IAnimal { 
... as before ...
}
Data with Variants

```java
interface IAnimal {
    ...
}

class Snake implements IAnimal {
    ...
    ... as before ...
}

class Dillo implements IAnimal {
    ...
    ... as before ...
}

class Ant implements IAnimal {
    ...
    ... as before ...
}
```
Variants in Java

- `@* #)*"+,-#. ,)? -#) & (-, #)0 = #("7(. , B)# . #6("* #)*"+,-#. ,)'G 6- 6) ("), . #9="H, H

; A grade is either   ⇒   ; A grade is either
; - false
; - num

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

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

- `@* #)*"+,-#. ,) , )9") )& (-, #, ) #0 . ' #., "'). #6(" *
  # *"+,-#. ,