CSE 3302
Programming Languages

Data Types

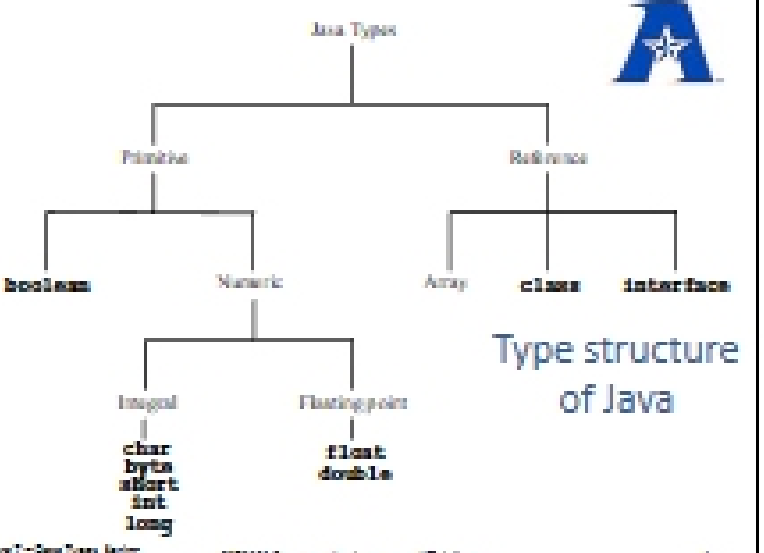
Chengkai Li, Weimin He
Spring 2008

Data Types

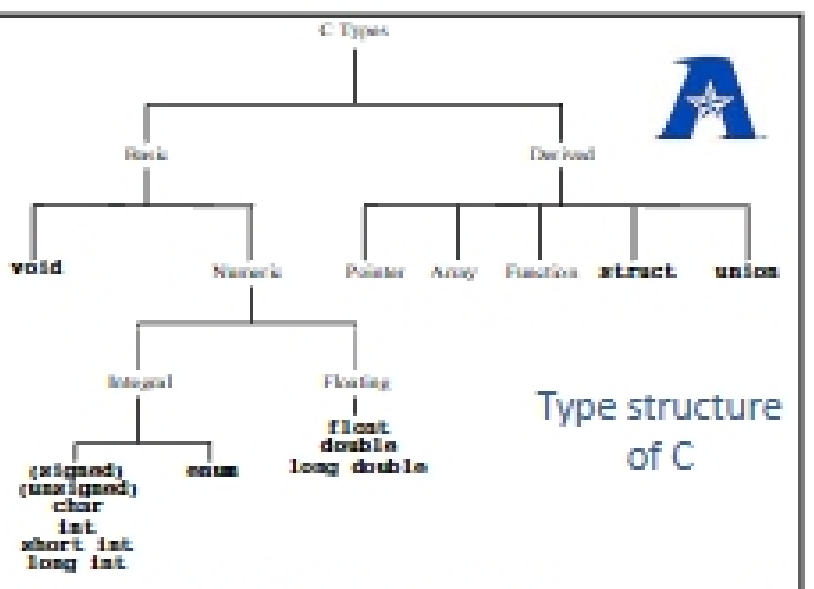
- **What is a data type?**
  A name with certain attributes:
  - The values that can be stored, the internal representation, the operations, ...
- **A data type is a set of values**
  - e.g., `int` in Java:
    ```
    int x;
    ```
    x: Integers= [-2147483648, 2147483647]
- **A data type is also a set of operations on the values**

Why are data types important?

- **Example: z = x / y; (Java)**
  - int x, y; x=5; y=2;
    - Integer division, x/y results in 2.
    - int z: z = 2;
    - double z: z=2.0;
  - double x, y; x=5; y=2;
    - floating-point division, x/y results in 2.5
    - int z: wrong!
    - double z: z=2.5;



Type structure of Java



Type structure of C

Simple Data Types

- **No internal structure:**
  e.g., `integer`, `double`, `character`, and `boolean`.
- **Often directly supported in hardware.**
  - machine dependency
- **Most predefined types are simple types.**
  - Exceptions: `String` in Java.
- **Some simple types are not predefined**
  - Enumerated types
  - Subrange types

Enumerated Types

Ordered set, whose elements are *named* and *listed* explicitly.

- Examples:
  ```
  enum Color_Type {Red, Green, Blue};        [C]
  type Color_Type is (Red, Green, Blue);     [Ada]
  datatype Color_Type = Red | Green | Blue;  [ML]
  ```
- Operations: ?
  Successor and predecessor

Ada Example

```
type Color_Type is (Red, Green, Blue);

x : Color_Type := Green;
x : Color_Type'Succ(x);
x : Color_Type'Pred(x);
put(x);          -- prints GREEN
```

- No assumptions about the internal representation of values
- Print the value name itself

Pascal Example

```
type
      cardsuit = (club, diamond, heart, spade);
      card = record
                  suit: cardsuit;
                  value: 1 .. 13;
             end;
var
      hand: array [ 1 .. 13 ] of card;
```

- Succ(diamond) = heart; Pred(spade) = heart;
- club < heart; is true.
- for acard := club to heart do

C Example

```
#include <stdio.h>
enum Color {Red, Green, Blue};
enum Courses {CSE1111=1, CSE1302=2, CSE2310=3, CSE6666=4};
main() {
    enum Color x = Green;
    enum Courses c = CSE2302;
    x++;
    printf("%d\n",x);
    printf("%d\n",Blue+1);
    printf("%d\n",c);
    return 0;
}
```

- **Enum** in C is simply int.
- Can customize the values

Java Example

```
public enum Planet { MERCURY (2.4397e6), EARTH (6.3781e6);
  private final double radius; // in meters
  Planet(double radius) {this.radius = radius; }
  private double radius() { return radius; }

  public static void main(String[] args) {
    for (Planet p : Planet.values())
      System.out.printf("the radius of %s is %f\n", p, p.radius());
  }
}
```

java.util.Enumeration has different meaning

```
for (Enumeration e = v.elements() ; e.hasMoreElements() ;)
    System.out.println(e.nextElement());
```


Evaluation of Enumeration Types

- **Efficiency** – e.g., compiler can select and use a compact efficient representation (e.g., small integers)
- **Readability** -- e.g. no need to code a color as a number
- **Maintainability** – e.g., adding a new color doesn't require updating hard-coded constants.
- **Reliability** -- e.g. compiler can check operations and ranges of value.

Courtesy of Charles Nicholas at UMBC

C Example for Maintainability

```
enum Color {White, Green, Blue, Black};
enum Color {White, Yellow, Green, Blue, Black};
main(){
    enum Color x = Black;
    int i = x;
    while (i >= White){
        if (i < Green)
            printf("this is a light color!\n");
        i--;
    }
}
```

What if no enumeration?

```
if (i < 3) printf("this is a light color!\n");
```

Has to be changed to:

```
if (i < 2) printf("this is a light color!\n");
```


Ada Example for Reliability

```
type DAY is (MON, TUE, WED, THU, FRI, SAT, SUN);
type DIRECTION is (NORTH, EAST, SOUTH, WEST);

GOAL : DIRECTION;
TODAY : DAY;
START : DAY;

TODAY := MON;
GOAL := WEST;
START := TODAY;

TODAY := WEST; -- Illegal. WEST is not a DAY value
TODAY := 5; -- Illegal. 5 is not a DAY value
TODAY := TODAY + START; -- Illegal. "+" is not defined for DAY
```


Subrange Types

Contiguous subsets of simple types, with a least and greatest element.

- Example:

```
type Digit_Type is range 0..9;                (Ada)
```

- Not available in C,C++,Java. Need to use something like:

```
byte digit;     //-128..127
...
if (digit>9 || digit <0) throw new DigitException();
```

- defined over **ordinal types:**
  - ordered, every value has a next/previous element
    - E.g., integer, enumerations, and subrange itself

Type constructors: Defining New Types

- **Type constructors as set operations:**
  - Cartesian product
  - Union
  - Subset
  - Functions (Arrays)
- **Some type constructors do not correspond to set operations (e.g., pointers)**
- **Some set operators don't have corresponding type constructors (e.g., intersection)**

Cartesian Product

- **Ordered Pairs of elements from U and V**

  $U \times V = \{(u, v) \mid u \in U \text{ and } v \in V\}$

- **Operations:**
  - **projection**

  $p_1: U \times V \rightarrow U$; $p_2: U \times V \rightarrow V$

  $p_1((u,v))=u$; $p_2((u,v))=v$

Examples

- **struct** in C

```
struct IntCharReal
{    int i;
     char c;
     double r;
}
int x char x double
```

- **record** in Ada

```
type IntCharReal is record
     i: integer;
     c: character;
     r: float;
end record;
```