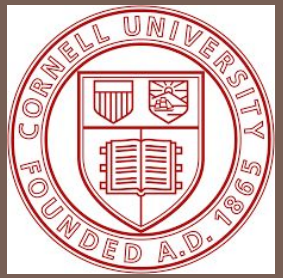
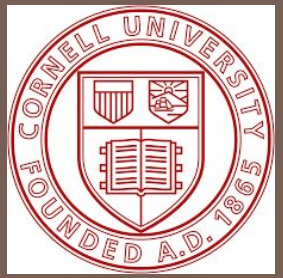
Object-oriented programming and data-structures



CS/ENGRD 2110 SUMMER 2018



Lecture 1: Types and Control Flow

http://courses.cs.cornell.edu/cs2110/2018su

Lecture 1 Outline

- Languages Overview
  - Imperative vs Declarative.
- Types
- Variable Assignment
- Control Flow and Procedures
  - scoping

Languages

- Declarative vs Imperative languages
  - Declarative:
    - Specify what should be done, not how
    - Ex: SQL `select * from people where name = "Natacha""`
  - Imperative
    - Specify both how and what should be done

- Java is **imperative** and **procedural**. What about Python?

Types

- Definition: **A type is a set of values together with operations on them**

- Java defines **primitive types** and **reference types**
  - Primitive types: Built-in types that act as building blocks for more complicated types that we'll look at next lecture
  - Reference types: Next lecture :)

- Example Type: integer:
  - values: …, –3, –2, –1, 0, 1, 2, 3, …
  - operations: +, –, *, /, unary –

- What about type boolean?

Most-used ‘primitive’ types

int: values: -2^{31} .. $2^{31}-1$
operations: +, −, *, /, %, unary −
size: 32 bits as signed integer

double: values like : −22.51E6, 24.9
operations: +, −, *, /, %, unary −
size: 64 bits as floating point number

char: values like : 'V' '$' '\n'
operations: none
size: 16 bits

boolean: values: true false
operations: ! (not), && (and), || (or)
size: 1 bit

Strong Typing

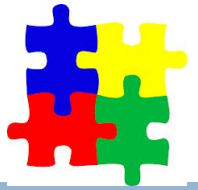
Matlab and Python are **weakly typed**:

One variable can contain at different times a number, a string, an array, etc.

One isn't so concerned with types.

Java **strongly typed**:
A variable must be declared before it is used and can contain only values of the type with which it is declared

Strong Typing



Matlab and Python are **weakly typed**:

One variable can contain at different times a number, a string, an array, etc.

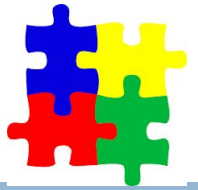
One isn't so concerned with types.

Java **strongly typed**:
A variable must be declared before it is used and can contain only values of the type with which it is declared

Valid Python sequence:

```
x= 100;
x= 'Hello World';
x= (1, 2, 3, 4, 5 );
```

Strong Typing



Matlab and Python are **weakly typed**:

One variable can contain at different times a number, a string, an array, etc.

One isn't so concerned with types.

Java **strongly typed**:
A variable must be declared before it is used and can contain only values of the type with which it is declared

Valid Python sequence:

```
x= 100;
x= 'Hello World';
x= (1, 2, 3, 4, 5 );
```

Corresponding Java

```
int x;
x= 100;

x= "Hello";
```

Illegal assignment:
"Hello" is not an **int**

Declaration of x:
x can contain only values of type **int**

Program Structure in Java

- The reason for all of this may not seem clear right now, will become clearer in next couple of lectures

```
package   packageName;
class myClass{
        void proc() {...}
        int fun() {...}
        public static void main(String[] args) { fun(); proc(); … }
}
```

- Must place myClass in file myClass.java

Basic variable declaration

Declaration: gives name of variable, type of value it can contain

`int x;`	Declaration of x, can contain an **int** value
`double area;`	Declaration of area, can contain a **double** value
`int[] a;`	Declaration of a, an **int** array.

Assignment statement

Assignment: assigns value to a variable.

Much like in other languages —need ';' at end:

<variable>= <expression> ;

Assignment statement

Assignment: assigns value to a variable.

Much like in other languages —need ';' at end:

<variable>= <expression> ;

```
int x;
x= 10;
… other code
x= x+1;
```

Have to declare x before assigning to it.

```
int x= 10;
… other code
x= x+1;
```

Can combine declaration with an initializing assignment. Shorthand for a declaration followed by an assignment.

Weakly typed versus strongly typed


```
x =  75 + "Hello";
int x = 75 + "Hello";
```

What happens in Python?

What happens in Java?

```
myVar = 100;
myVar = myvar + 1
print myVar
```

What happens in Python?

```
int myVar = 100;
myVar = myvar + 1;
System.out.println(myVar);
```

What happens in Java?

Weakly typed versus strongly typed

Weakly typed:

Shorter programs, generally.

Programmer has more freedom, language is more liberal in applying operations to values.

Strongly typed:

Programmer has to be more disciplined. Declarations provide a place for comments about variables.

More errors caught at compile-time (e.g. it's a syntax error to assign a string to an **int** variable).

Note: weak and strong typing not well defined; literature has several definitions

Functions & Procedures.

- Group linked actions into a single unit of execution
  - **Functions** take input **parameters** and return **something**
  - **Procedures** take input **parameters** and return **nothing**

Functions & Procedures.

- Group linked actions into a single unit of execution
  - **Functions** take input **parameters** and return **something**
  - **Procedures** take input **parameters** and return **nothing**

```
/** return sum of a and b */
public double sumFunction(double a, double b) {
    System.out.println("Sum of " a + " and " +  b);
      return a + b;
}

/** prints sum of a and b */
public void sumProcedure(double a, double b) {
    System.out.println("Sum is "+ (a + b));
}
```

Specification: in comment before function

Parameter declarations

Function Body

Return Type

No Return Type for Void procedures

Control Flow Recap

- Control flow syntax is similar to other languages
  - For (initialisation; termination; increment)
    - For (int i = 0 ; i < 10 ; i++) { … }
  - While(boolean_expresion)
    - While (i < 10) { … ; i++}
  - If (boolean_exp)
    - If { … } else { … }

- Branching statements
  - Break: Exit loop
  - Continue: Skip concurrent iteration of loop
  - Return: Exit function immediately

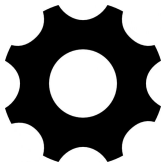
Local Variables

- **Definition**: defined inside a function/procedure or in any conditional block. They have **block-level scope** and are only accessible in the block where they are defined.

Local Variables

- **Definition**: defined inside a function/procedure or in any conditional block. They have **block-level scope** and are only accessible in the block where they are defined.

```
/** return sum of a and b */
public double sumFunction(double a, double b) {
    double sum = a + b;
    System.out.println("Sum of " a + " and " +  b);
    return sum;
}
```



- **Use local variables to write clean code and avoid repetition!**

Local Variables - Scoping

- **Definition**: defined inside a function/procedure or in any conditional block. They have **block-level scope** and are only accessible in the block where they are defined.

- A block is defined by a starting bracket **{** and a closing bracket **}**

- Local variables are destroyed once they go outside of scope

Local Variables - Shadowing

- **Definition**: A variable *shadows* another if they have the same name and are accessible in the same place.

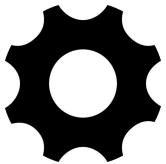
- To what declaration does a name refer?
  - Code in a block can reference names declared in that block, as well as names that appear in enclosing blocks.

  - **Use inside-out rule:** Look first in method body, starting from name and moving out; then look at parameters; then look outside method in the object.

Local Variables - Shadowing


```
double sum = 0.0;
/** return sum of a and b */
double sumFunction(double a, double b) {
    if (a>0.0) {
        double sum = a + b;
        System.out.println("Sum is " + sum);
    }
    System.out.println("Sum is " + sum);
    return sum;
}
```

What will the print statements output?



- Always give clear names to your variables
- Create variables with the smallest possible scopes. As close to their first use

References in JavaHyperText

type
primitive type
type, strong versus weak typing
function
function call
procedure
procedure call
variable
variable declaration
expression
assignment statement
local variables