```java
int foo() {
    bar(); // Java: scope of methods is entire enclosing class
}

int bar() {
    // scope of class names is entire package (may be spread across different files)
}

C/C++: names must be declared before use
```
Module: collection of mutually usable things, some may be exported to outside procedures/globally data operated on by them in only one instance

modules as types: create multiple instances of a single module (just like classes but no inheritance)
Dynamic scope

drawLine(x1,x2,y1,y2)
  setColor(color)
  for (every pixel on line)
    drawPixel(
  )

3

drawBlueTriangle(-----)
  saved = setColor(C)
  setColor(BLUE)
  drawLine(-----)
  drawLine(-----)
  drawLine(-----)
  setColor(saved)

3
drawCircle(-----)
  setColor(GREEN)
  drawCircle(-----)
  drawLine(-----)
  drawLine(-----)
  drawLine(-----)
1st class procedures: can use procedures like variables (pass as arguments, return assigning) [create at run-time ?]

2nd class procedures: passed as arguments only,

3rd class procedures: can only call

shallow binding: use bindings in place when passed procedures called

deep binding: use bindings in place at time procedure was passed in

(requires referencing environment to be passed along with procedure (procedure + referencing environment = closure))
aliases: different names for same thing

\[ x \rightarrow l \]
\[ 8 \rightarrow p \]
\[ x, y, z \text{ alias same thing} \]

overloading: 1 name for many things

Java: methods w/same name, differ by lists
C++: extends to operators

\[ x \text{foo} \Rightarrow x.\text{operatorfoo}(y) \]