Pascal Code Fragment

procedure C; begin end;

procedure A (P : procedure; i : integer);
procedure B;
begin B
  write(i);
end B;
begin A
  if i = 1 then A(B,2)
else P;
end A;

begin main
  A(C,1);
end main.

JavaScript Code Fragment

function sub1() {
  var x;
  function sub2() {
    alert(x);
  };
  function sub3() {
    var x;
    x = 3;
    sub4(sub2);
  };
  function sub4(subx) {
    var x;
    x = 4;
    subx();
  };
  x = 1;
  sub3();
}

function sub1() {
  var x;
  function sub2() {
    alert(x);
  };
  function sub3() {
    var x;
    x = 3;
    sub4(sub2);
  };
  function sub4(subx) {
    var x;
    x = 4;
    subx();
  };
  x = 1;
  sub3();
}

Java without Generics

import java.util.Hashtable;

class Test {
  public static void main(String[] args) {
    Hashtable h = new Hashtable();
    h.put(new Integer(0), "value");
    String s = (String) h.get(new Integer(0));
    System.out.println(s);
  }
}
Java with Generics

import java.util.Hashtable;
class Test {
    public static void main(String[] args) {
        Hashtable<Integer, String> h =
                new Hashtable<Integer, String>();
        h.put(new Integer(0), "value");
        String s = h.get(new Integer(0));
        System.out.println(s);
    }
}

Co-routines

Chapter 10 Topics

- The General Semantics of Calls and Returns
- Implementing “Simple” Subprograms
- Implementing Subprograms with Stack-Dynamic Local Variables
- Nested Subprograms
- Blocks
- Implementing Dynamic Scoping
The General Semantics of Calls and Returns

- Def: The subprogram call and return operations of a language are together called its subprogram linkage

Implementing “Simple” Subprograms

- Call Semantics:
  1. Save the execution status of the caller
  2. Carry out the parameter-passing process
  3. Pass the return address to the callee
  4. Transfer control to the callee

Implementing “Simple” Subprograms

- Return Semantics:
  1. If pass-by-value-result parameters are used, move the current values of those parameters to their corresponding actual parameters
  2. If it is a function, move the functional value to a place the caller can get it
  3. Restore the execution status of the caller
  4. Transfer control back to the caller

Implementing “Simple” Subprograms

- Required Storage: Status information of the caller, parameters, return address, and functional value (if it is a function)
- The format, or layout, of the noncode part of an executing subprogram is called an activation record
- An activation record instance is a concrete example of an activation record (the collection of data for a particular
An Activation Record for “Simple” Subprograms

- Local variables
- Parameters
- Return address

Implementing Subprograms with Stack-Dynamic Local Variables

- More complicated because:
  - The compiler must generate code to cause implicit allocation and deallocation of local variables
  - Recursion must be supported (adds the possibility of multiple simultaneous activations of a subprogram)

Typical Activation Record for a Language with Stack-Dynamic Local Variables

- Local variables
- Parameters
- Dynamic link
- Return address

Code and Activation Records of a Program with “Simple” Subprograms
Implementing Subprograms with Stack-Dynamic Local Variables

- The activation record format is static, but its size may be dynamic
- The dynamic link points to the top of an instance of the activation record of the caller
- An activation record instance is dynamically created when a subprogram is called

An Example C Function

```c
void sub(float total, int part)
{
    int list[4];
    float sum;
    ...
}
```