Systems Software: Tracing Code Examples

Andrew Harn

University of Central Florida

Brief Review

- **Subprogram linkage** refers to the call and return operations of the subprograms
  - Linkage in languages depends on several factors, including parameter passing and value returning
- The data structure representing the non-code portion of a subprogram is an **activation record**
- The structure of a an activation record depends on the linkage designed for the language

Language Comparisons between Java and PL/0

- Parameter-Passing: Java has it, PL/0 does not
- Function Value Returning: Java has it, PL/0 does not
- Nested Subprograms: Java does not have it, PL/0 does
- PL/0 AR Contents: Static Link, Dynamic Link, Return Address, Local Variables
- Java AR Contents*: Functional Value, Dynamic Link, Return Address, Parameters, Local Variables

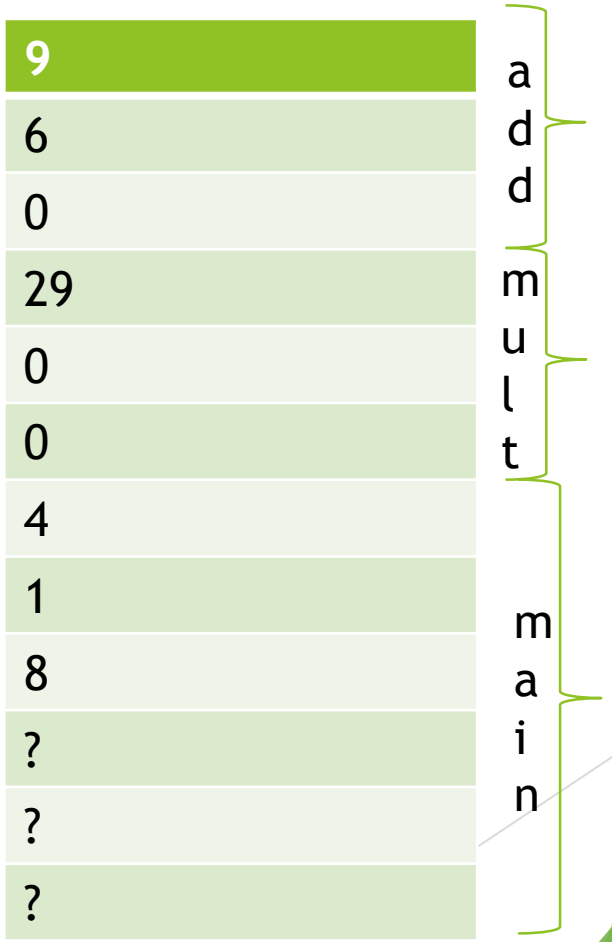
**Does not entirely reflect the actual Java activation record*

Tracing the execution of the PL/0 language

- Essentially the same as running through the P-Machine ISA instructions, except that each step taken is that of a line of the PL/0 language
  - The AR structure is the same as that for the P-machine described
- The P-Machine examples in the earlier slides have example PL/0 codes
  - Traces can be verified by comparing the machine states made to those reached by the ISA instructions
  - If for each state reached by the PL/0 language can be matched to a state reached by the ISA, then the trace should be valid

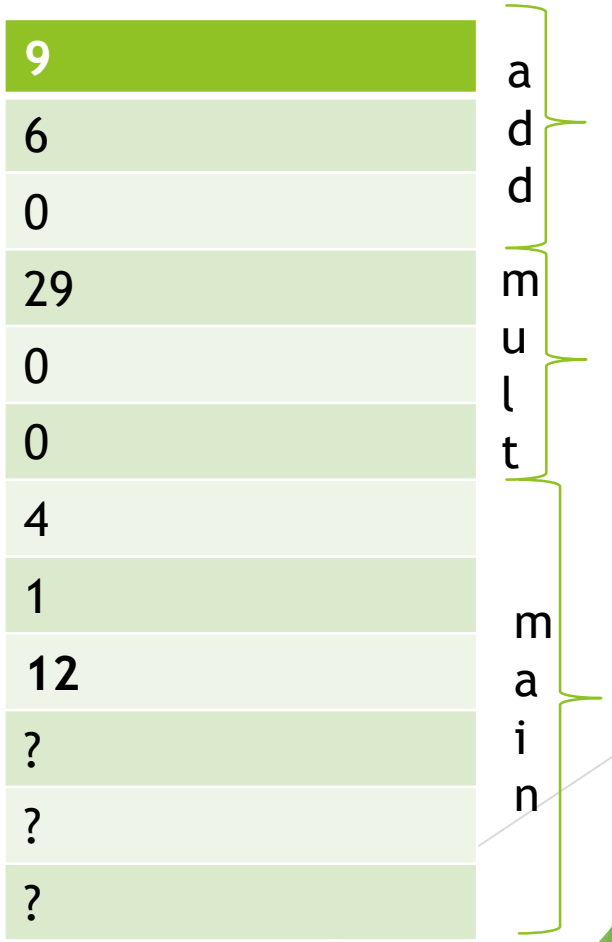
Brief PL/0 Line Execution Example

```
var product, a, b;
procedure mult;
  begin
    […]
    call add;
    […]
  end;
end;
procedure add;
  begin
    product := product + b;
  end;
end;
begin
  […]
  call mult;
  […]
end.
```



Brief PL/0 Line Execution Example

```
var product, a, b;
procedure mult;
  begin
    […]
    call add;
    […]
  end;
end;
procedure add;
  begin
    product := product + b;
  end;
end;
begin
  […]
  call mult;
  […]
end.
```



Java Example #1

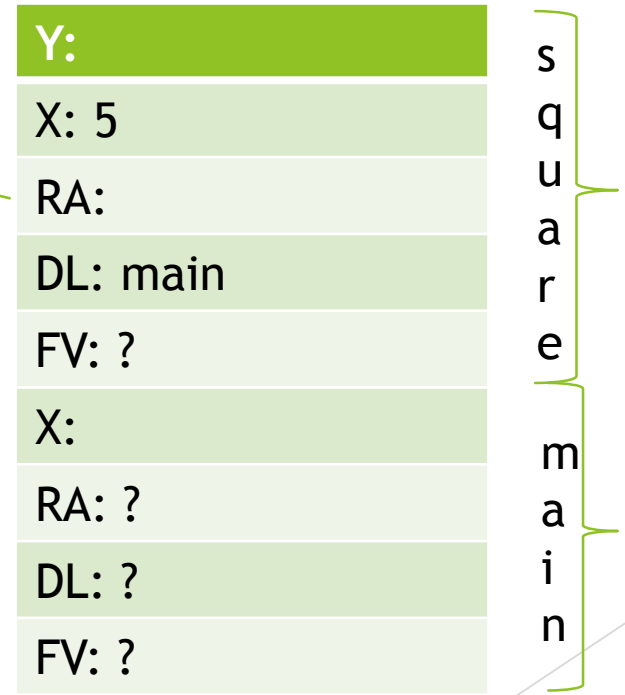
```
void main(String[]
args) {
    int x = square(5);
}
int square(int x) {
    int y = x;
    y *= x;
    return y;
}
```

Java Example #1

```
void main(String[]
args) {
    int x = square(5);
}
int square(int x) {
    int y = x;
    y *= x;
    return y;
}
```

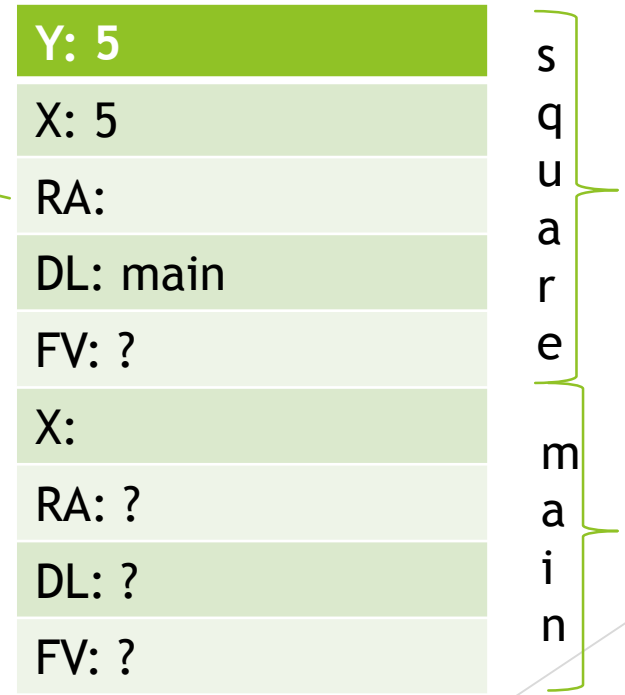
Java Example #1

```
void main(String[]
args) {
    int x = square(5);
}
int square(int x) {
    int y = x;
    y *= x;
    return y;
}
```



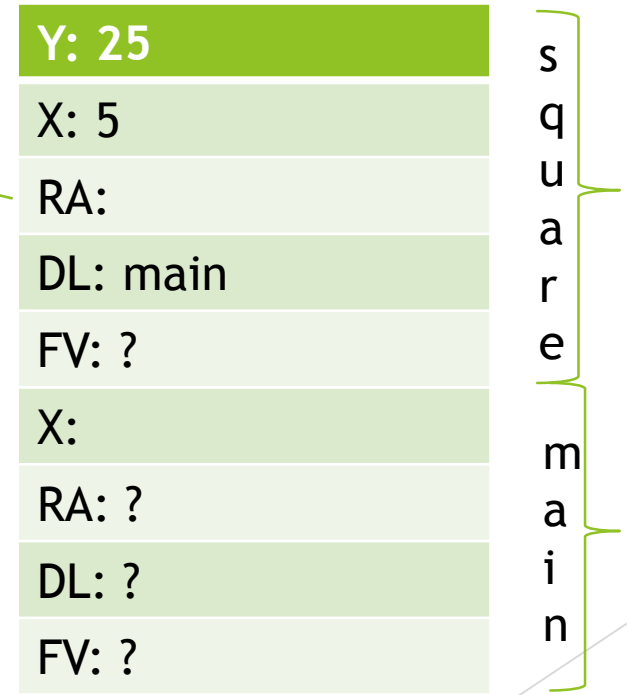
Java Example #1

```
void main(String[]
args) {
    int x = square(5);
}
int square(int x) {
    int y = x;
    y *= x;
    return y;
}
```



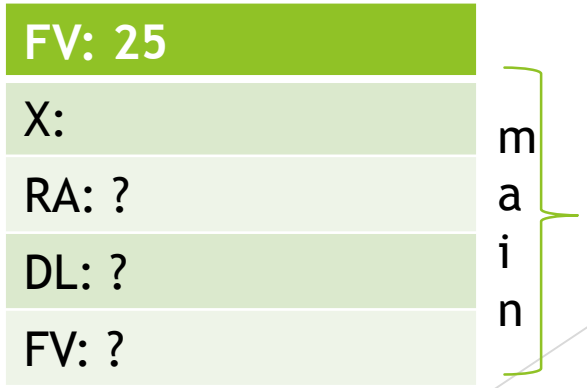
Java Example #1

```
void main(String[]
args) {
    int x = square(5);
}
int square(int x) {
    int y = x;
    y *= x;
    return y;
}
```



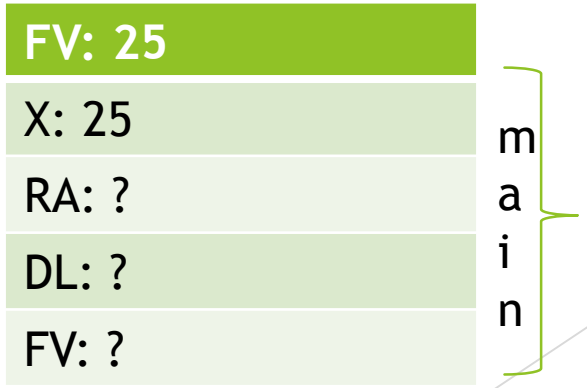
Java Example #1

```
void main(String[]
args) {
    int x = square(5);
}
int square(int x) {
    int y = x;
    y *= x;
    return y;
}
```



Java Example #1

```
void main(String[]
args) {
    int x = square(5);
}
int square(int x) {
    int y = x;
    y *= x;
    return y;
}
```



Java Example #2

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```

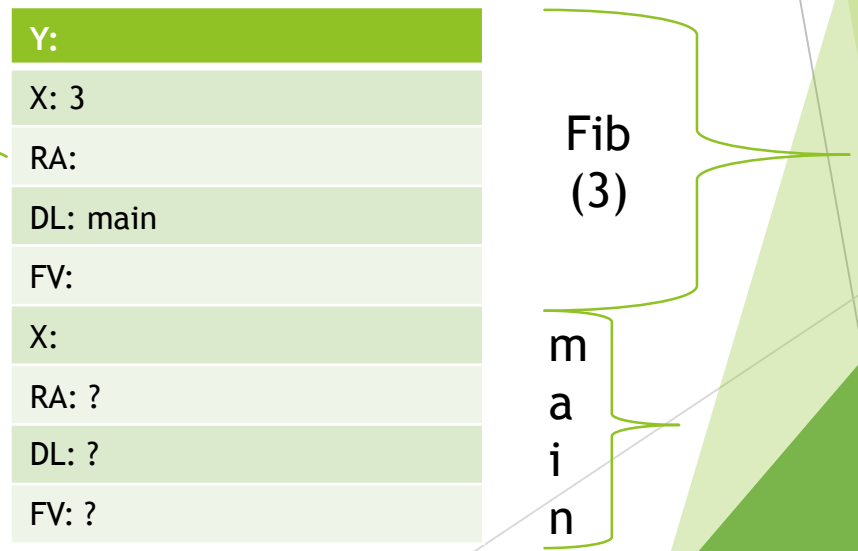
We wish to know what the activation record stack looks like at each Fibonacci call.

Java Example #2

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x – 1);
    x = fib(x – 2);
    return x + y;
}
```

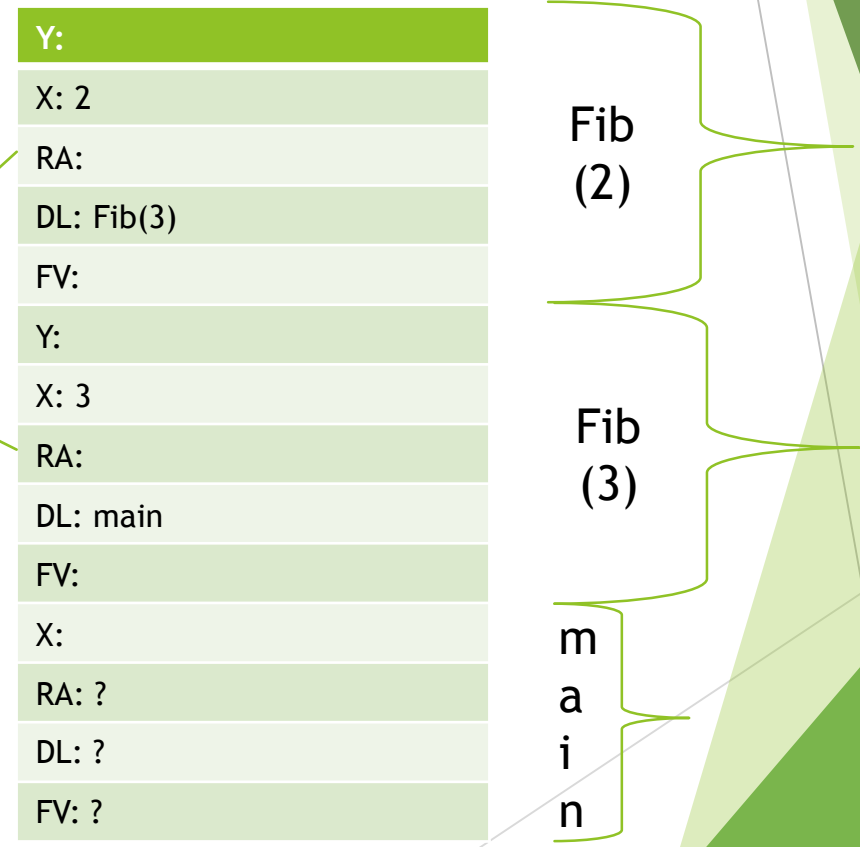
Java Example #2 Checkpoint 1

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```



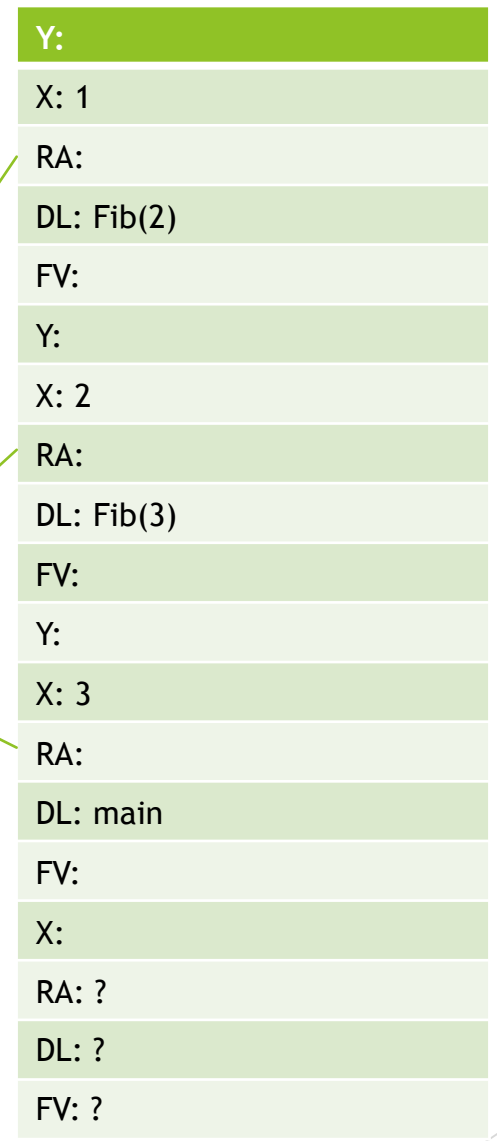
Java Example #2 Checkpoint 2

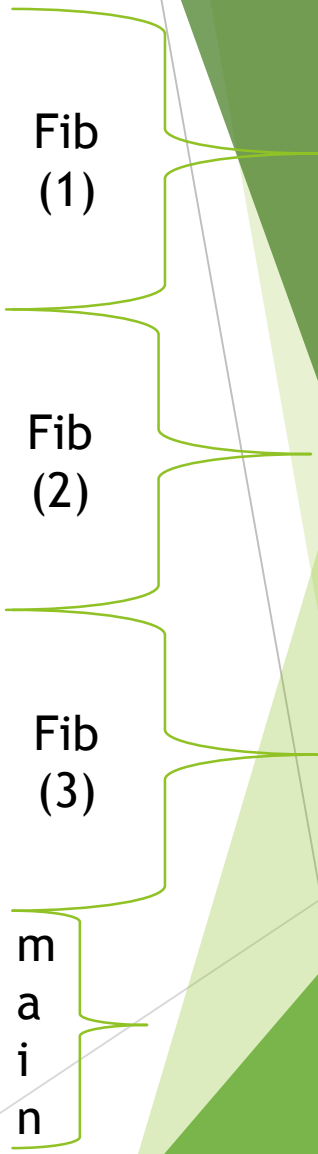
```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```



Java Example #2 Checkpoint 3

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```

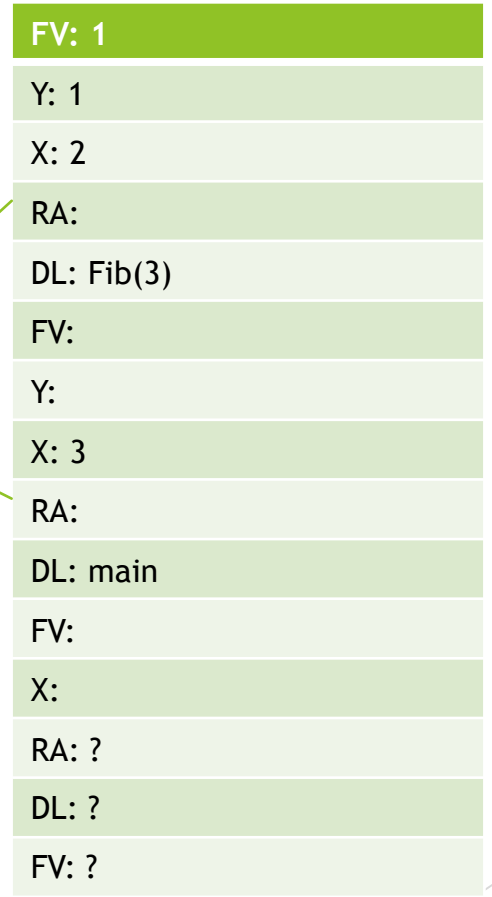




Java Example #2

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```

Executed Fib(2) and assigned returned value to y



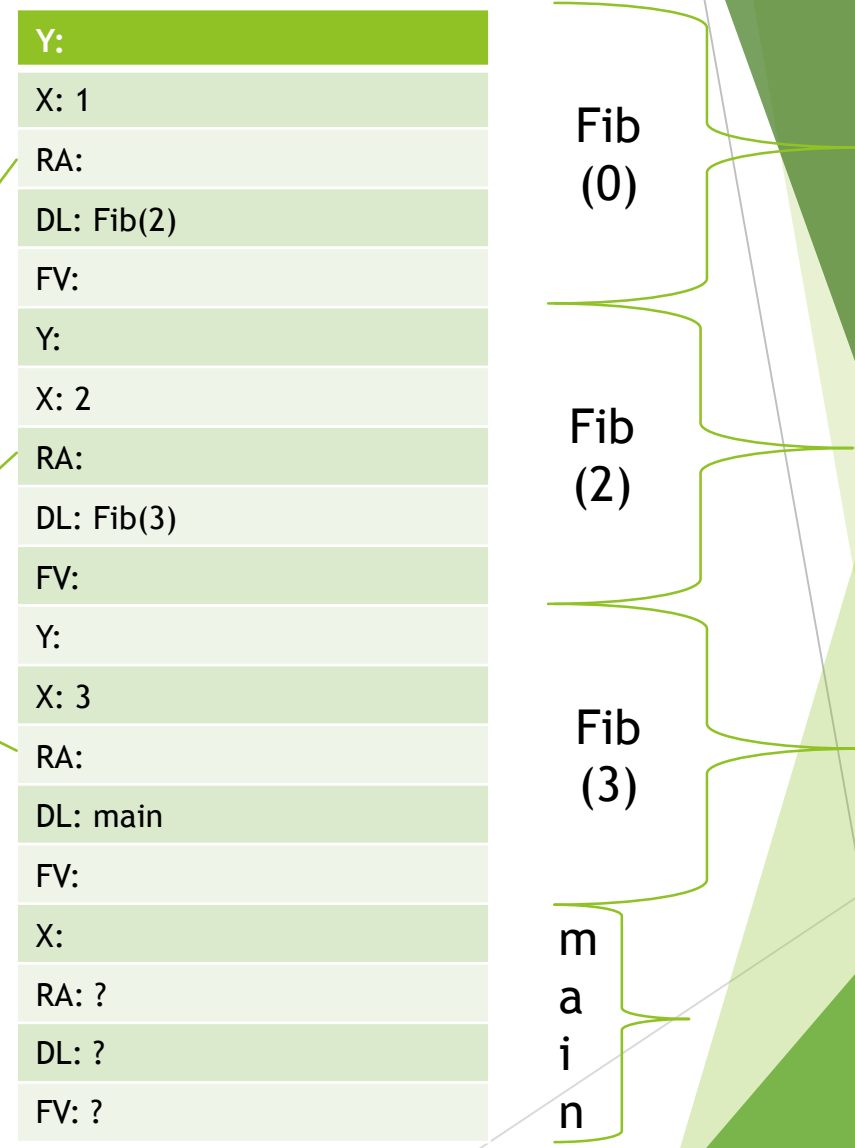
Fib (2)

Fib (3)

main

Java Example #2 Checkpoint 4

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```



Java Example #2

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```

Going to return from Fib(2)

FV: 1
Y: 1
X: 1
RA:
DL: Fib(3)
FV:
Y:
X: 3
RA:
DL: main
FV:
X:
RA: ?
DL: ?
FV: ?

Fib (2)

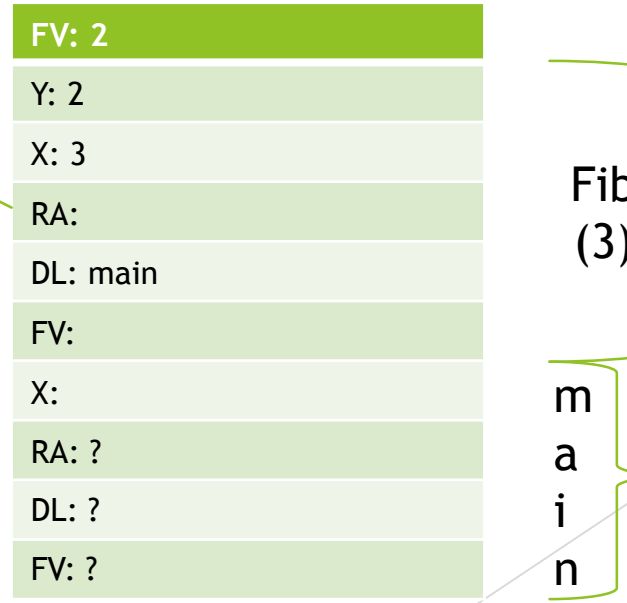
Fib (3)

main

Java Example #2

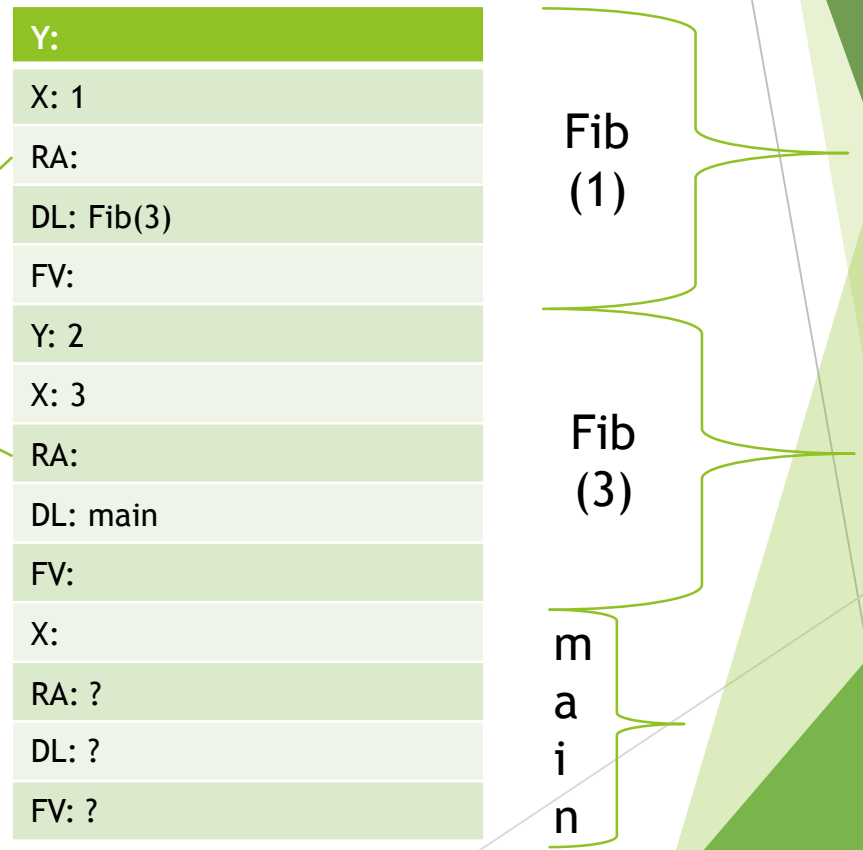
```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```

Returned from Fib(2)



Java Example #2
Checkpoint 5

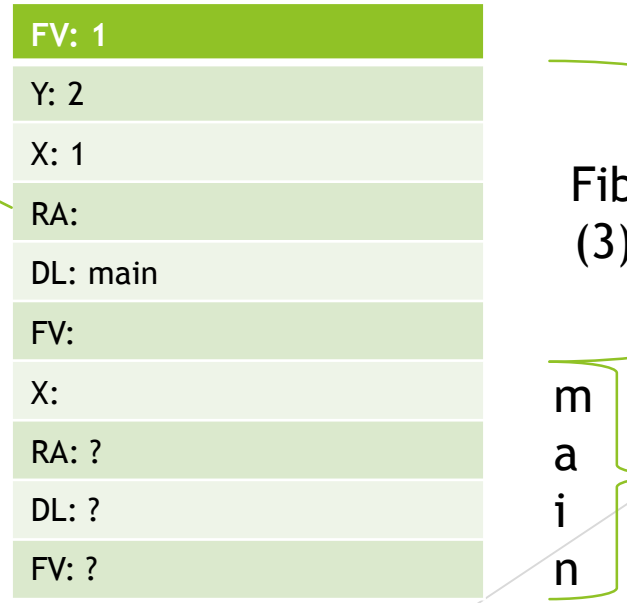
```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```



Java Example #2

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```

Returning from Fib(3)



Java Example #2

```
void main(String[] args) {
    int x = fib(3);
}
int fib(int x) {
    if (x < 2) {
        return 1;
    }
    int y = fib(x - 1);
    x = fib(x - 2);
    return x + y;
}
```