Homework 2 Concerns

- http://www.cse.buffalo.edu/~shapiro/Courses/CSE305/2010/Homeworks/hw2.pdf

Language Subsets

- Sebesta makes the point that we only learn a subset of any language.

Readability vs. Writability (vs. Optimality)

- Conditional Operator vs. Conditional Statement
  - C, C++, Java

```
System.out.println((x == y) ? x : y);
```

Vs.

```
If(x == y) System.out.println(x);
else System.out.println(y);
```

(Fun fact: This is a lot how functional languages express if's!)

Readability vs. Writability (vs. Optimality)

Shift Operator

int n = 10;

n = n << 2;

What's the value of n?

Readability vs. Writability (vs. Optimality)

Shift Operator

int n = 10;

n = n << 2;

What's the value of n?

1010 bit shifted 2 to the left.
Becomes: 101000 (=40).

Readability vs. Writability (vs. Optimality)

Shift Operator

int n = 10;

n = n << 2;

What's the value of n?

1010 bit shifted 2 to the left.
Becomes: 101000 (=40).

int n = 100;

n = n >> 1;

What's the value of n?

Readability vs. Writability (vs. Optimality)

Shift Operator

int n = 10;

n = n << 2;

What's the value of n?

1010 bit shifted 2 to the left.
Becomes: 101000 (=40).

int n = 100;

n = n >> 1;

What's the value of n?

1100100 bit shifted 1 to the right.
Becomes: 110010 (=50).

Readability vs. Writability (vs. Optimality)

Shift Operator

```
int n = 10;
n = n << 2;
```

What's the value of n?

1010 bit shifted 2 to the left.
Becomes: 101000 (=40).

```
int n = 100;
n = n >> 1;
```

What's the value of n?

1100100 bit shifted 1 to the right.
Becomes: 110010 (=50).

```
int n = 10;
n = n << 30;
```

What's the value of n?

Readability vs. Writability (vs. Optimality)

Shift Operator

int n = 10;

n = n << 2;

What's the value of n?

1010 bit shifted 2 to the left.
Becomes: 101000 (=40).

int n = 100;

n = n >> 1;

What's the value of n?

1100100 bit shifted 1 to the right.
Becomes: 110010 (=50).

int n = 10;

n = n << 30;

What's the value of n?

This is a 32 bit int. One bit is the sign.
This becomes invalid!
(Actual result: -2147483648)

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Static Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Static Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

X refers to Main's X

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Static Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

X refers to Main's X

-If Main calls sub1 calls sub2, what does Y refer to in sub2?

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Static Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

X refers to Main's X

-If Main calls sub1 calls sub2, what does Y refer to in sub2?

-**Y refers to sub2's Y**

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Dynamic Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Dynamic Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

X refers to sub1's X

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Dynamic Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

X refers to sub1's X

-If Main calls sub1 calls sub2, what does Y refer to in sub2?

Scope

```
procedure Main is
    X, Y : Integer;
    procedure sub1 is
        X: Integer;
        begin
        …
        end;
    procedure sub2 is
        Y: Integer;
        begin
        …X…
        …Y…
        end;
    begin
    …
    end;
```

Assume Dynamic Scoping:
-If Main calls sub1 calls sub2, what does X refer to in sub2?

X refers to sub1's X

-If Main calls sub1 calls sub2, what does Y refer to in sub2?

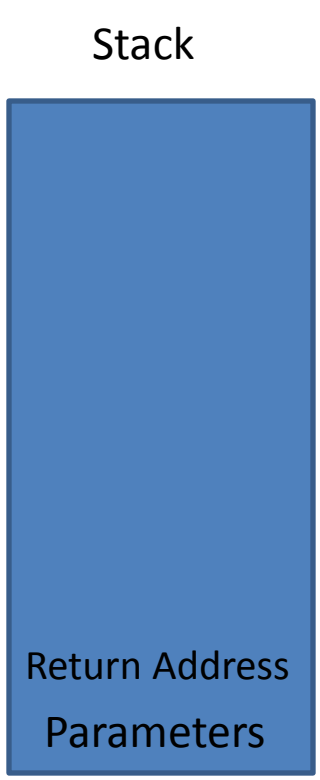
-**Y refers to sub2's Y**

The Run-time Stack

```
public class RTS{
    public static void main(String[] args){
        RTS1 tClass = new RTS1();
        tClass.testVar += tClass.testMethod(5);
        System.out.println(tClass.testVar);
    }
}

class RTS1{
    public int testVar;

    public int testMethod(int i){
        int j = i+1;
        return j;
    }
}
```

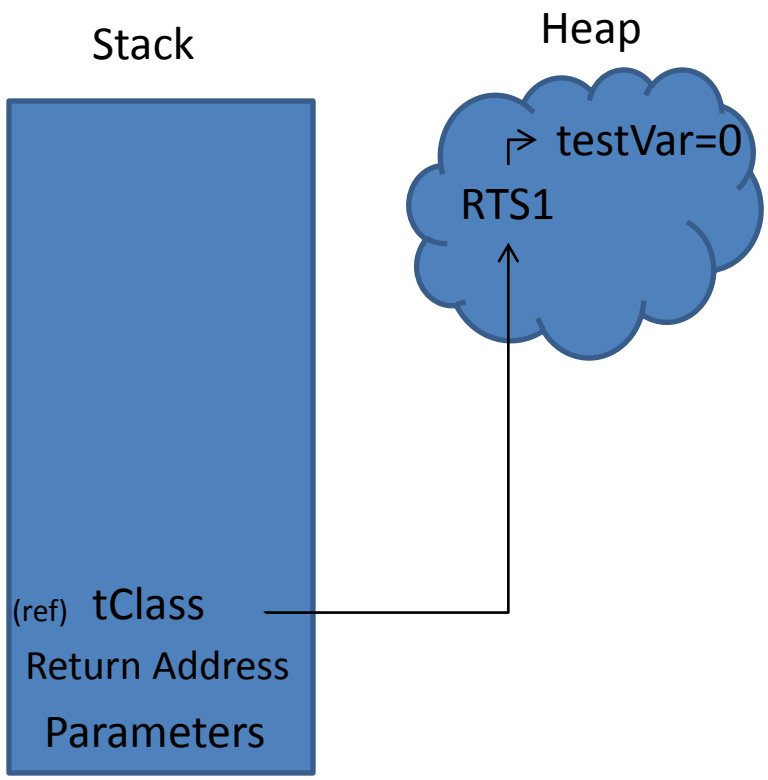


The Run-time Stack

```
public class RTS{
	public static void main(String[] args){
		RTS1 tClass = new RTS1();
		tClass.testVar += tClass.testMethod(5);
		System.out.println(tClass.testVar);
	}
}

class RTS1{
	public int testVar;

	public int testMethod(int i){
		int j = i+1;
		return j;
	}
}
```

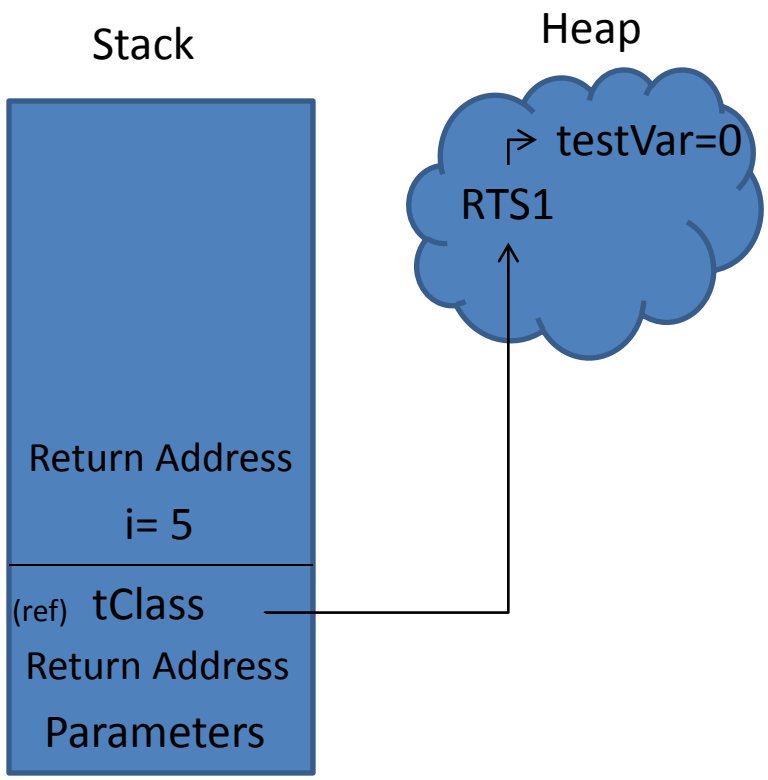


The Run-time Stack

```
public class RTS{
	public static void main(String[] args){
		RTS1 tClass = new RTS1();
		tClass.testVar += tClass.testMethod(5);
		System.out.println(tClass.testVar);
	}
}

class RTS1{
	public int testVar;

	public int testMethod(int i){
		int j = i+1;
		return j;
	}
}
```

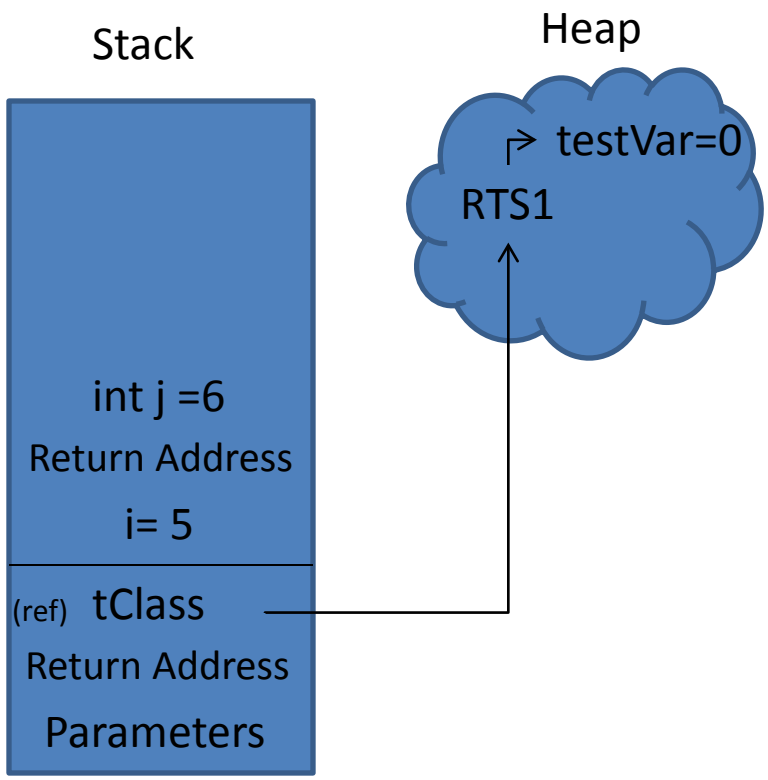


The Run-time Stack

```
public class RTS{
	public static void main(String[] args){
		RTS1 tClass = new RTS1();
		tClass.testVar += tClass.testMethod(5);
		System.out.println(tClass.testVar);
	}
}

class RTS1{
	public int testVar;

	public int testMethod(int i){
		int j = i+1;
		return j;
	}
}
```

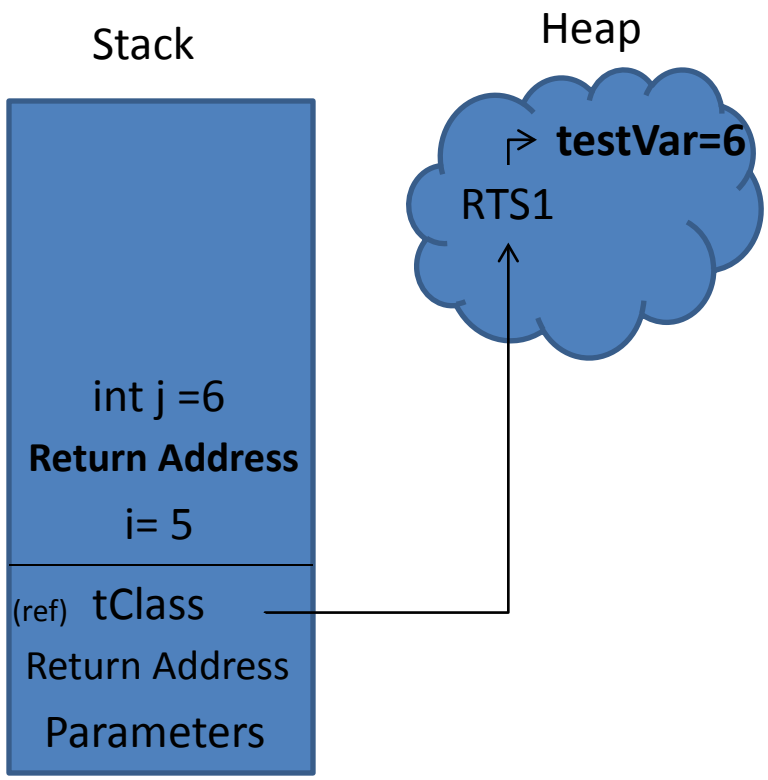


The Run-time Stack

```
public class RTS{
	public static void main(String[] args){
		RTS1 tClass = new RTS1();
		tClass.testVar += tClass.testMethod(5);
		System.out.println(tClass.testVar);
	}
}

class RTS1{
	public int testVar;

	public int testMethod(int i){
		int j = i+1;
		return j;
	}
}
```

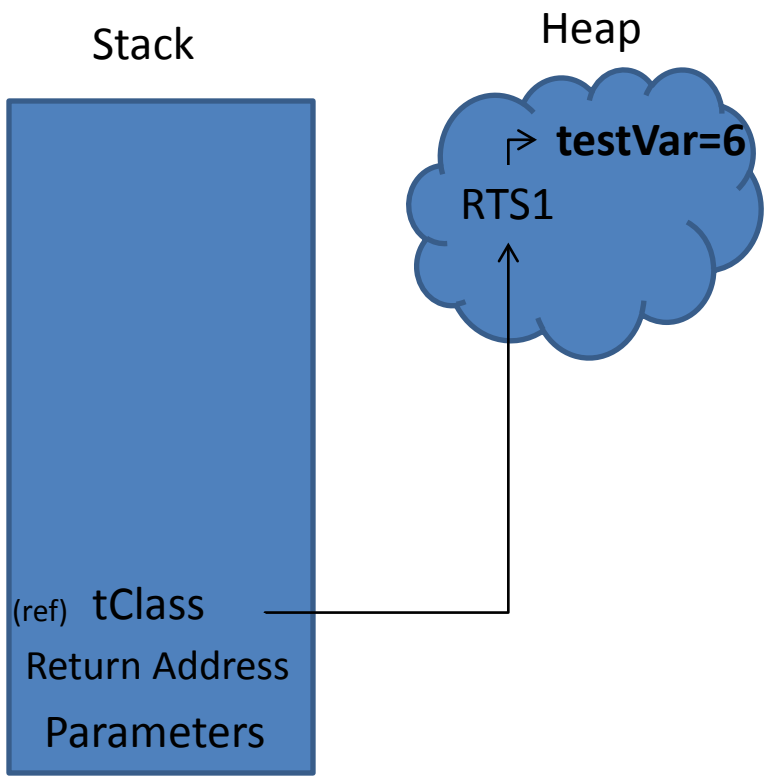


The Run-time Stack

```
public class RTS{
	public static void main(String[] args){
		RTS1 tClass = new RTS1();
		tClass.testVar += tClass.testMethod(5);
		System.out.println(tClass.testVar);
	}
}

class RTS1{
	public int testVar;

	public int testMethod(int i){
		int j = i+1;
		return j;
	}
}
```

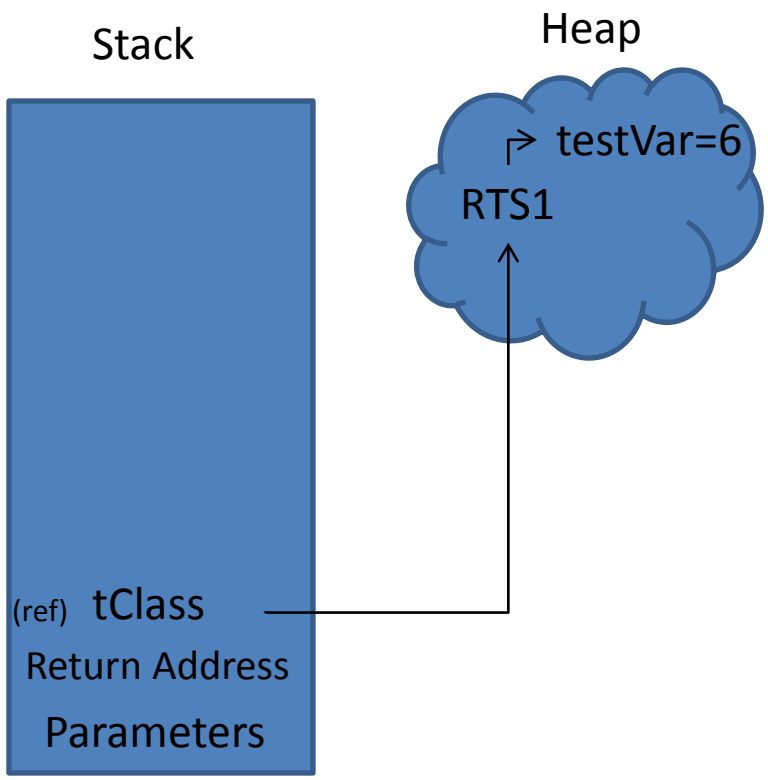


The Run-time Stack

```
public class RTS{
	public static void main(String[] args){
		RTS1 tClass = new RTS1();
		tClass.testVar += tClass.testMethod(5);
		System.out.println(tClass.testVar);
	}
}

class RTS1{
	public int testVar;

	public int testMethod(int i){
		int j = i+1;
		return j;
	}
}
```



A “problem” with C/C++

- Built-in arrays in C/C++ can’t be returned!

```
int[] test()
{
    int i[10];
    return i;
}
```

“arr.cpp:1: error: expected unqualified-id before [token”

Why not?

A “problem” with C/C++

- Built-in arrays in C/C++ can’t be returned!

```
int[] test()
{
    int i[10];
    return i;
}
```

“arr.cpp:1: error: expected unqualified-id before [token”

Why not?

A: They are on the stack!

What does Java do?

```
static int[] test(){
        int[] i = new int[10];
        return i;
}
```

This is fine! Why?

What does Java do?

```
static int[] test(){
        int[] i = new int[10];
        return i;
}
```

This is fine! Why?

A: “new int[10]” is on the heap. i (which is just a reference to the new int[10]) is on the stack!

One workaround for C/C++

```
int* test(){
    int* i;
    i = (int*)malloc(10);
    return i;
}
```

Do you see why this is OK?