CS 211: Java Syntax Tour

Chris Kauffman

Week 2-1

Logisitics

Labs

- Lab 1 Exercises Due Tonight
- Lab 2 Quiz: this week
- Exercise labs after week 1: attendance optional
- Quiz/Task labs: attendance required

Reading: See schedule

- BJP Ch 1-5: `for`/`if`/`while`, methods
- BJP 7: Arrays
- Lab Manual chapters

Project 1

- Posted, deadline Sunday 2/5
- Field questions Today

Goals Today

- Exercise: Write a static method which returns a reversed copy of a parameter array
- Discuss method declaration
- Discuss equality semantics

By Friday make sure you . . .

- Have a development environment (IDE or Command Line)
- Can create new `.java` files
- Experimented with hello world type programs
- Can zip a directory
- Finished/close to finishing Lab 1, submit to Blackboard

Basic Structure of a Java Source File

```
public class SomeClass {
  public static TypeR myMethod1(TypeA a, TypeB b){ // function/method
    TypeC c = some code;  // informative comment
    some more code;       // another informative comment
    return someR;
  }
  public static int calls = 0;  // global-ish variable
  public static int theAverage(int x, int y, int z){
    int a = x + y + z;
    a = a / 3;
    calls = calls + 1;
    return a;
  }
  public static void main(String [] args){ // main method
    int myAvg = theAverage(1,2,3);
    int metaAvg = SomeClass.theAverage(myAvg, 2*myAvg, 3*myAvg);
    System.out.println("The average is "+myAvg);
    System.out.println("The meta average is "+metaAvg);
    System.out.println("Calls to average: "+calls);
    return;  // optional
  }
}
```

Every Programming Language

Start by looking for the following

- ☐ Comments
- ☐ Statements/Expressions
- ☐ Variable Types
- ☐ Assignment
- ☐ Basic Input/Output
- ☐ Conditionals (if-else)
- ☐ Iteration (loops)
- ☐ Aggregate data (arrays, structs, objects, etc)
- ☐ Function Declarations
- ☐ Library System

Syntax Demo Program

- `Demo.java` in `02-basic-syntax.zip` contains examples for today
- Also several other programs in the zip

Note: All code examples are posted some time after class in the same spot as the lecture slides. *Where are the lecture slides posted?*

Conditionals

- `if/else`
  - `Demo.java`
  - Act on a `boolean`
  - Comparisons: ==, !=, <, >, <=, >=
  - Nesting
  - Chaining
- `switch/case`
  - Useful in some special cases, but not generally
  - Maybe we'll talk about it some time

Iteration

4 flavors

- Now - `Iteration.java`
  - `while`
  - Traditional `for`
- Maybe Later
  - `do while`
  - `for` each (collections)

while

```
while(condition)
  this gets done repeatedly;
this gets done once;

while(condition){
  this gets done repeatedly;
  as does this;
  and this;
}
this gets done once;
```

Look at Iteration.java

for

```
for(initialize; condition; update)
  do some stuff repeatedly;
then do this;

for(initialize; condition; update){
  do some stuff repeatedly;
  and some other stuff repeatedly;
}
then do this;
```

Question

Do you need both `for` and `while`?

Arrays - Multiple of the same kind of thing

See `ArrayDemo.java`

Define Now there's a type `bleh`, it looks like blah

- Done for you: part of the java language

Declare Here is a variable, it's type is `bleh`

```
int ia[] = new int[3];
double doubs[] = new double[10];
boolean [] bools = new boolean[4];
```

Assign Element foo of variable bar gets value blip

```
ia[0] = 1;
doubs[2] = 1.2345;
bools[3] = true;
```

Access Retrieve element foo of variable bar

```
int i = ia[1];
double d = doubs[4];
boolean b = bools[0];
```

Length

Arrays carry their `length`

It's an `int` (or `long`?).

```
int ia[] = new int[3];
System.out.println(ia.length);
int len = ia.length;

for(int i=0; i<ia.length; i++){
  System.out.print(ia[i]+" ");
}
```

Can't change `length`

```
// Compile ERROR
ia.length = 20;
```

Why not?

Can cause runtime errors

```
ia = new int[5];
ia[10] = 12;
Exception in thread "main"
java.lang.ArrayIndexOutOfBoundsException: 10
at ArrayDemo.main(ArrayDemo.java:23)
```

Easy Exam Questions to Write

Convert to `for`

```
double tol = 1e-4;
double S = 45.0;
double x = 45.0/2;
double err;

err = (S - x*x)*(S - x*x);
while(err > tol){
  x = (x + S/x) / 2.0;
  err = (S - x*x)*(S - x*x);
}
```

Answers in code pack

Convert to `while`

```
int x = 48;
int f = -1;
boolean found = false;

for(int i=x-1;
    i>1 && !found;
    i--)
  {
    if(x % i == 0){
      f = i;
      found = true;
    }
  }
```

Warm-up Exercise: Array Basics

How does one

- Declare an array called `myInts` which can hold integers 5?
- Set the element at index 3 of `myInts` to 10?
- Retrieve the contents of index 4 of `myInts` and store it in a variable named `i`?
  - What will be the value of `i`?
- Declare an array `myReals` which can hold 10 double precision floating point numbers?
- What results from retrieving index 10 from `myReals`?
- Compare the number of elements in `myInts` and `myReals` in an `if` condition?
- How can I allow `myReals` to hold more than 10 numbers?

Exercise: Reverse copy of an Array

Write a static method

```
public static int [] reverseCopy(int [] a){
  ... // YOUR CODE HERE
  return reversedArray;
}
```

which creates a reverse copy of the array a and returns it. You will need to do the following.

- Allocate space for `reversedArray`
- Iterate through a and copy elements to the corresponding positions in the `reversedArray`

```
int arr1[] = {5, 4, 3, 2, 1};
int rev1[] =
  ReverseArray.reverseCopy(arr1);
for(int i=0; i<rev1.length; i++){
  System.out.print(rev1[i] + " ");
}
System.out.println(); //newline
// Expect: 1 2 3 4 5
```

```
int [] arr2 = {2, 4, 6, 8};
int [] rev2 =
  ReverseArray.reverseCopy(arr2);
for(int i=0; i<rev2.length; i++){
  System.out.print(rev2[i] + " ");
}
System.out.println(); //newline
// Expect: 8 6 4 2
```

Arrays in Memory

Spend a moment diagramming how `reverseCopy(int [] a)` works in memory

- Separate areas of memory for the original array and new one
- Important to fully grasp things to come

Array Goodies

Declare and Initialize

```
int a[] = {1, 2, 3, 4};
```

Initialize Dynamically

```
int b[];
...
b = new int[]{7, 6, 5};
myFunc(new int[]{3,1,4,1,5,9});
```

Strings

- Strings are like arrays of characters
- Have an *immediate syntax* for initialization and assignment
- Access individual characters with method `charAt(int i)`
- Length retrieved with the `length()` method

```
String s = "Hello World";
//          01234567890
char c = s.charAt(4); // 'o'
char d = s.charAt(7); // 'o'
if(c == d){
  System.out.println("Equal");
}
else{
  System.out.println("Not");
}
int len = s.length();  // 11, note parens
int arr[] = new int[5];
int lenA= arr.length;  // 5, no parens for arrays
```