Knowing this is Required

Anatomy of a class

- A java program may start with import statements, e.g. import java.util.Arrays.
- A java program contains a class definition. This includes the word "class" followed by a name, and usually the word "public". Curly brackets follow that, e.g. "public class *Name* { ... }".
- Knowing this is required.

Anatomy of a class

- The body of a class is the part where the elipses is in this program:
  public class Hello { ... }
- The body of a program contains:
  - Zero or more Fields
  - Zero or more Methods
  - Zero or more Constructors

Anatomy of a field declaration

- A field may be public or private
- A field may be final
- A field may be static
- A field *always* has a type
- A field *always* has a name
- A field may be followed by an equal sign and an initial value
- A field delcaration *always* terminates with a semi-colon.
- Knowing this is required.

Anatomy of a type

- A type can be either primitive
  - byte, short, int, long
  - float, double,
  - char
  - boolean
- Or an object
  - arrays and strings are objects
  - all classes define objects
- Knowing this is required

Anatomy of a type

- If a type is an array, that fact is specified by the inclusion of square brackets, and that's part of the type, e.g.
  - int[] x;
  - double[] b;
- Note that we have type (int[]), a name (x), and a semi-colon.
- Knowing this is required

Anatomy of a value

- The types byte, short, int, and long are all "integers" but are stored using increasingly larger amounts of memory. You can initialize them with simple numbers, e.g. 1, 2, or 666.
- The types float, and double are "real numbers" but are stored using increasingly larger amounts of memory. You can initialize them with numbers that have decimals, e.g. 3.14.
- Knowing this is required.

Anatomy of a value

- Obects are (with a few exceptions) initialized with keyword "new"
- If you have created a class Hello, that is a type. Since it is an object, you can instantiate it as follows:
  - Hello h = new Hello();
  - h is called an "instance" of Hello
- The parenthesis are required. The parenthesis contain the arguments (a comma separated list of values) to the constructor.
- Knowing this is required.

Anatomy of a value

- Arrays may be instantiated with operator new and square brackets containing a number, e.g.
  - int[] x = new int[10];
- Arrays may be instantiated with operator new, square brackets, and a list of values inside curly brackets, e.g.
  - int[] x = new int[]{1,3,5}
- Knowing this is required

Anatomy of a method

- Methods may be public or private
- Methods may be static
- A method has a type or is void
- A method always has a name
- A method contains parenthesis filled with a comma separated list of arguments, these arguments have types and names
- A method has curly brackets, and a body
- Knowing this is required

Anatomy of a constructor

- Constructors may be public or private
- Constructors may *not* be static
- Constructors do *not* have a type
- Constructors have the same name as the class they are in
- A constructor contains parenthesis filled with a comma separated list of arguments, these arguments have types and names
- A constructor has curly brackets, and a body
- Knowing this is required

Calling a static method

- A method is called by supplying its name and a comma delimited list of values.
- A static method can simply be called using its class name and a dot, e.g.
  - int x = Hello.foo();
  - double y = Math.sin(3);
- If the static method is a member of the current class, it can simply be called, e.g.
  - int x = foo();
- Knowing this is required

Calling an instance method

- A method is called by supplying its name and a comma delimited list of values.
- An instance method can simply be called using an instance and a dot, e.g.
  - Hello h = new Hello();
  - int x = h.foo();
- If the instance method is called from inside an instance method of the current class, it can simply be called, e.g.
  - int x = foo();
- Knowing this is required

Calling a constructor

- A constructor is called with operator new and the class name.
- The arguments to the constructor are a comma delimited list of values, e.g.
  - Hello h = new Hello();
  - Hello h2 = new Hello(arg1);
- The names h and h2 denote instances
- Knowing this is required

A method body may contain...

- Variable declarations, e.g. int x = 4;
- Assignments of variables or fields, pi = 4*atan2(1,1)
- Invocations of methods, e.g. doSomething(arg1,arg2)
- if statements
- for loops
- while loops
- return statements
- Knowing this is required

Anatomy of an if statement

- An if statement starts with the word "if"
- An if statement is followed by parenthesis containing a conditional expression or test
- An if statement is followed by a single statement, or curly brackets containing one or more statements
- Knowing this is required

Anatomy of a for loop

- A for loop starts with the word "for"
- A for loop contains parenthesis with three inputs, each delimited by semicolons.
  - The first input runs before the loop
  - The second input is a test and runs before each iteration of the loop to determine whether the iteration will take place
  - The third runs after each iteration of the loop
- A for loop is followed by a statement or curly brackets containing one or more statements
- Knowing this is required

Anatomy of a while loop

- A while loop starts with the word "while"
- A while loop is followed by parenthesis that contain a conditional expression or test to determine whether to perform the body of the loop
- The body of the loop is either a single statement, or curly brackets containing multiple statements
- Knowing this is required

Anatomy of an array

- Arrays are of fixed length, if you want an array to be of a different length, you must allocate a new one
- Arrays have a final static field called length

Filling an array

```
int[] x = new int[10];
for(int i=0;i<x.length;i++) {
   x[i] = 2*i-3;
}
int[] y = new int[x.length];
for(int i=x.length-1;i>=0;i--) {
   y[i] = x[i];
}
// Understanding this is required
```

Finding a sum, or max

```
int[] x = new int[]{1,3,8,4,2,1};
int sum = 0;
for(int i=0;i<x.length;i++)
   sum += x[i];
int max = x[0];
for(int i=1;i<x.length;i++) {
   if(x[i] > max) {
      max = x[i];
 }} // Understanding this is required
```

Printing out an array

```
public class Array {
  int[] array;
  public Array(int n) { array = new int[n]; }
  public String toString() {
     StringBuilder sb = new StringBuilder();
     for(int i=0;i<array.length;i++) {
        if(i > 0) sb.append(',');
        sb.append(array[i]);
     }
     return sb.toString(); }}
// Understanding this is required
```

Removing Evens

```
int[] removeEvens(int[] a) {
  int numOdds = 0
  for(int i=0;i<a.length;i++)
    if(a[i] % 2 == 1) numOdds++;
  int[] odds = new int[numOdds];
  int n = 0;
  for(int i=0;i<a.length;i++)
    if(a[i] % 2 == 1)
      odds[n++] = a[i];
   return odds; }
// Understanding this is required
```

Reversing an array

```
void reverse(int[] array) {
  int lo = 0; int hi = array.length-1;
  while(lo < hi) {
     int tmp = array[lo];
     array[lo] = array[hi];
     array[hi]  = tmp;
     lo++; hi--;}}
// Understanding this is reqiured
```

Logical Arrays: Adding

```
// Contain an int and an array
int len=...; int[] array = ...;
int valueToAdd = 3;
if(len >= array.length) {
   int[] newArray = new int[array.length+10];
   for(int i=0;i<array.length;i++)
     newArray[i] = array[i];
   array = newArray; }
array[len] = valueToAdd; len++;
// Understanding this is required
```

Logical Arrays: Removing

```
// Contain an int and an array
int len=...; int[] array = ...;
int index = 3;
int tmp = array[index];
array[index] = array[len-1];
array[len-1] = tmp;
len--;
// Understanding this is required
```

Hexadecimal floating point values

- You can specify a floating point number in hexadecimal format. You start with "0x"
- ...then a hexadecimal digit, e.g. 0 through 9 or a through f. A decimal may be included.
- ...then p followed by a decimal number. If present, the value is multiplied by 2 to the power of that number.
- Example: 0x9a.bcp10 == 158448

Hexadecimal floating point values

- Knowing this is *not* required. :)