Structure & Format

• Just like last year’s midterm (online)
Data Types

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>int</td>
<td>numeric: whole numbers</td>
<td>int x = 123;</td>
</tr>
<tr>
<td>double, float</td>
<td>numeric: includes decimals</td>
<td>double y = 12.34;</td>
</tr>
<tr>
<td>boolean</td>
<td>True/False (yes/no, on/off)</td>
<td>boolean isEmpty = False;</td>
</tr>
<tr>
<td>char</td>
<td>Single letter or number character</td>
<td>char c = ‘h’;</td>
</tr>
<tr>
<td>String</td>
<td>Multiple letter/number characters</td>
<td>String greeting = “hello world”;</td>
</tr>
</tbody>
</table>

• normal order of operations
• Remember:
  • numeric operations require numeric data types
  • int / int = int    BUT    int / double = double  [see next slide]
  • math.Round() returns a long
Exercise

• fill in the blank

```java
public static double threeQuarters (int x) {
    return _____;
}
```
Exercise: Boolean “zen”

- possible to streamline boolean return values

```java
public static boolean isLeapYear (int x) {
    return _____;
}
```
Exercise: Chars and Ints

```java
public static void demo (int x) {
    int start = (int) 'a';
    int end = (int) 'z';
    for (int i = start; i <= end; i++) {
        System.out.println((char) i);
    }
}
```

What does this output?
Data Types: String vs Char Array

• Declaration
  • `char[] chs = new char[7]; chs[0] = 'a';` ...
  `char[] chs = {'a','b','c','d','e','f','g'};` // called initializer

• String `s = “abcdefg”;

• Method
  • `String
    • s.length(); s.charAt(int i); s.substring(int start, int end);
    • s.equals(String str2) vs s == str2;
  
• Character Array
  • `chs.length; chs[n];`
  • To compare arrays, must check each pair of elements
## Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>assignment</td>
<td>int x = 123;</td>
</tr>
<tr>
<td>==, !=</td>
<td>testing for equality or inequality</td>
<td>if (x == y) { ... }</td>
</tr>
<tr>
<td>&lt;=, &gt;=, &lt;, &gt;</td>
<td>less than (or equal to), greater than (or equal to)</td>
<td>if (x &lt;= y) { ... }</td>
</tr>
<tr>
<td>* , / , + , - , %</td>
<td>Arithmetic</td>
<td>int x = 1701 – 32;</td>
</tr>
<tr>
<td>+=, -=, *=, /=, %=</td>
<td>Inline arithmetic-assignment</td>
<td>x -= 5</td>
</tr>
<tr>
<td>!</td>
<td>'Not' – reverses true/false result of expression</td>
<td>if (! (a == b)) { ... }</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>boolean AND</td>
<td>if ((a == b) &amp;&amp; (b == c)) { ... }</td>
</tr>
</tbody>
</table>

• Order of operations?
Variables (Scope)

• **local vs global**

  ```java
  public class Hello{
      static String name = “World”; // Global Variable
      static final double PI = 3.14159;
      public static void main(String args[]){
          int numTimes = 5; // Local Variable

          for (int i = 0; i < numTimes; i++) {
              System.out.println(“Hello “ + name);
          }
      }
  }
  ```

• What is the scope of `numTimes`? what about `name`?

• What does `final` mean?
Variables in methods

• Passing variables through methods
  
  ```java
  public static int addOne(int num) {
      num = num + 1; // A
      System.out.println("1. number = " + num);
      return num;
  }
  
  public static void main(String[] args) {
      int x = 23;
      addOne(x); // B
      int y = addOne(x); // C
      System.out.println("2. x = " + x); // what is printed?
  }
  ```

• what are the values at the commented letters?
## Flow control

<table>
<thead>
<tr>
<th>Conditional Statement</th>
<th>Loop</th>
</tr>
</thead>
<tbody>
<tr>
<td>if (&lt;test&gt;) {</td>
<td>for (init; condition; increment) {</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>else if (&lt;test&gt;) {</td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>}</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>else {</td>
<td>while (condition) {</td>
</tr>
<tr>
<td>...</td>
<td>}</td>
</tr>
<tr>
<td>}</td>
<td>}</td>
</tr>
<tr>
<td>switch</td>
<td></td>
</tr>
</tbody>
</table>

- Control statements rely on evaluation of boolean expressions
- Nested `if/else` [next slide]
- Questions about `if/else` with `return` (“Java doesn’t have semantic analysis”) [next next slide]
Nested conditionals

```java
if (boolean_expression_1) {
    // Executes when the boolean expression 1 is true

    if (boolean_expression_2) {
        // Executes when the boolean expression 2 is true
    }
}
```
if/else with return

public static int max(int a, int b) {
    if (a > b) {
        return a;
    }
    // Error: not all paths return a value
}

public static int max(int a, int b) {
    if (a > b) {
        return a;
    } else if (a <= b) {
        return b;
    }
    // Error: syntax analysis missing a path
}

public static int max(int a, int b) {
    if (a > b) {
        return a;
    } else if (a <= b) {
        return b;
    }
    // OK
}

public static int max(int a, int b) {
    int myMax = a; // default
    if (b > a) {
        myMax = b;
    }
    return myMax; // OK
}

public static int max(int a, int b) {
    if (b > a) {
        return b;
    }
    return a; // OK
}
Switch

• Don’t worry too much about these - just be able to recognize them and say what they do.

```java
switch (<variable>) {
    case <value>:
        <statement>;
        break;
    case <value>:
        <statement>;
        break;
```

• Can only test for equality
Loops

- **for** is better when you know the number of iterations ahead of time
  ```java
  for (int i = 0; i < n; i++) {
      // there will be n iterations
      <statement>;
      ...
  }
  ```

- otherwise probably use **while**
  ```java
  <initialization>;
  do {
      <statement>;
      ...
      <update>;
  } while (<test>);
  // don’t forget ;
  ```

- but they are generally **interchangeable** (Q: how could one do this?)
Methods

• Defining
  
  ```
  public static <return-type> <name> (<type> <name>, ... ) {
    <statement>;
  }
  ```

• Calling
  • A non-void function ‘evaluates’ to something.
  • You can assign this value to a variable.
    ```
    int y = 4;
    int x = multByTwo(y);
    ```

• Parameters
  • arguments have a defined type
  • arguments and any variables declared inside have scope within the function

• Returning
  • how to give back a computed value from a function when its work is done
Overloading

• Allow multiple methods to share same name
• Their argument signatures differentiate them (e.g. type, number)

```java
class Sample {
    public static void display (char c) {
        System.out.println(c);
    }

    public static void display (int x, char c) {
        System.out.println(c + ": " + x);
    }
}
```

• Why would we want to do this?
More overloading

• Is the following valid overloading?

```java
public static int mymethod(int a, int b, float c) {
    ...
}

public static int mymethod(int var1, int var2, float var3) {
    ...
}
```

element from http://beginnersbook.com/2013/05/method-overloading/
Java Classes to know

- **String**
  - `length()`, `charAt()`, `substring()`, `indexOf()`, `split()`
- **Math**
  - `round()`, `max()`, `min()`, `sqrt()`
- **Scanner**
  - `next()`, `nextInt()`, `nextDouble()`
  - `hasNext()`
  - `hasNextInt()`
  - `hasNextDouble()`
# String class

<table>
<thead>
<tr>
<th>Method</th>
<th>Return type</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>length()</td>
<td>int</td>
<td>returns number of characters in String</td>
</tr>
<tr>
<td>charAt(int i)</td>
<td>char</td>
<td>returns character at i\textsuperscript{th} position in String</td>
</tr>
<tr>
<td>substring(int start, int end)</td>
<td>String</td>
<td>returns substring from start to end-1</td>
</tr>
<tr>
<td>indexOf(char c);</td>
<td>int</td>
<td>Overloaded method. Returns the index in String of the first occurrence of the c or str</td>
</tr>
<tr>
<td>indexOf(String str);</td>
<td>int</td>
<td></td>
</tr>
<tr>
<td>split(String regex)</td>
<td>String[]</td>
<td>Returns array of strings split by regex</td>
</tr>
<tr>
<td>toLowerCase(), toUpperCase()</td>
<td>String</td>
<td>self-explanatory. Returns a string.</td>
</tr>
<tr>
<td>equals()</td>
<td>bool</td>
<td>Compare two strings. <strong>DIFFERENT FROM ==</strong></td>
</tr>
<tr>
<td>equalsIgnoreCase()</td>
<td>bool</td>
<td>Tests whether two strings are equal, without considering case.</td>
</tr>
</tbody>
</table>
## Math class

<table>
<thead>
<tr>
<th>Method</th>
<th>Return type</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td>round(double x)</td>
<td>long</td>
<td>returns the integer-rounded value of x</td>
</tr>
<tr>
<td>max(int a, int b); min(int a, int b)</td>
<td>int</td>
<td>returns character at i-th position in String</td>
</tr>
<tr>
<td>sqrt(double x)</td>
<td>double</td>
<td>returns square root</td>
</tr>
</tbody>
</table>
## Scanner class

<table>
<thead>
<tr>
<th>Method</th>
<th>Return type</th>
<th>What it does</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>next()</code></td>
<td>String</td>
<td>This method finds and returns the next complete token from this scanner.</td>
</tr>
<tr>
<td><code>nextInt()</code></td>
<td>int</td>
<td>This method scans the next token of the input as an int</td>
</tr>
<tr>
<td><code>nextDouble()</code></td>
<td>double</td>
<td>This method scans the next token of the input as a double</td>
</tr>
<tr>
<td><code>hasNext()</code></td>
<td>boolean</td>
<td>This method returns true if this scanner has another token in its input.</td>
</tr>
<tr>
<td><code>hasNextInt()</code></td>
<td>boolean</td>
<td>This method returns true if the next token in this scanner's input can be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>interpreted as a int value using the <code>nextInt()</code> method.</td>
</tr>
<tr>
<td><code>hasNextDouble()</code></td>
<td>boolean</td>
<td>This method returns true if the next token in this scanner's input can be</td>
</tr>
<tr>
<td></td>
<td></td>
<td>interpreted as a double value using the <code>nextDouble()</code> method.</td>
</tr>
</tbody>
</table>

- **What is a token? What does `next()` split on?**
  - **Whitespace:** \n, \t, ‘ ’
System.out.println();
System.out.print();
System.out.printf(String template, <args>);

• Differences above?
• Placeholders in printf (and String.format)?
  • %d
  • %.2f
• Only one template string in printf!
StdDraw

• see class slides / review homework 4
Handling exceptions

• Exception: a mechanism to represent a runtime error
• Option 1: test before you do something
  • e.g. scanner.hasNextInt()
• Option 2: clean up errors after they arise

```java
public static void divide (int num1, int num2) {
    try {
        num1 = 0;
        num2 = 62 / num1;
        System.out.println("Try block message");
    } catch (ArithmeticException e) {
        System.out.println("Error: Don't divide a number by zero");
    }
    System.out.println("I'm out of try-catch block in Java.");
}
```

example from http://beginnersbook.com/2013/04/try-catch-in-java/
Handling exceptions

• You can have many catches for one try.
• What if we want to catch all exceptions?

```java
    catch (Exception e) { ... }
```

What about throw?

```java
    public static void demo(int x) throws IOException {
        throw new IOException;
    }
```

• most important thing is to know that the calling function must handle the exception
Arrays

• Creating an array
  • 2 ways
    int[] numbers = new int[5];  int[] numbers = {1, 1, 2, 3, 5}

• Accessing elements in an array
  int x = numbers[2]; // (assuming numbers declared with {1,1,2,..})
  • what is x?

• Modifying arrays
  • What does this piece of code do?
    for (int q = 0; q < numbers.length; q++) {
      numbers[q] *= 2;
    }
Arrays

• Comparing arrays
  • == will NOT work

• We can make our own method
  • Ideas?

```java
public static boolean isArrEqual (int[] arr1, int[] arr2) {
    if (arr1.length != arr2.length) {
        return false;
    }
    for (int index = 0; index < arr1.length; index++) {
        if (arr1[index] != arr2[index]) {
            return false;
        }
    }
    return true;
}
```
Tips for studying

• Review last year’s midterm!
Tips for taking the test

• Write anything you want to get graded (or maybe partially!)
• One page reference sheet is allowed
• Closed book exam
• Comments not required, but recommended for partial credit.
• Don’t stress.