public class Intro
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");
    }
}
What We're Doing

• In lecture
  – Sight read some Java, start to learn the syntax
  – Overview of very basic rules
  – Introduction to the basic elements

• In section
  – Try out CloudCoder
  – Try out DrJava
Topic 02: Intro to Java

- Sight reading exercise
- Case Sensitivity & Whitespace
  - Identifiers
    - 2.1.3
  - Comments
    - 2.1.6
In Class: Sight Reading

public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!"y);

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}

http://www.cs.arizona.edu/classes/cs127a/spring16/slides/class_exercises/
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}

In Class: Sight Reading

{} groups code together
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}

In Class: Sight Reading

{ }

{ } groups code together
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");
        int x;
        x = 1;
        int y;
        y = (x+1)*(x+1);
        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world! ");

        int x;
        x = 1;  // Statement: "Store the value 1 into the variable x"

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error ");
        }
    }
}
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
Topic 02: Intro to Java

- Sight reading exercise
- **Case Sensitivity & Whitespace**
- Whitespace
- Identifiers
- Comments

2.1.3

2.1.6
Case Sensitivity

• **ALL CODE** in Java is case-sensitive!

```
Foo = Bar + Baz;
foo = bar + baz;   // NOT the same!

if (x == y) ...    // works!
If (x == y) ...    // ERROR

int i;
I = 0;     // ERROR
```
Whitespace: Definition

• What is whitespace?
  – Spaces
  – Line breaks (a.k.a. “newline”)
  – Tabs

• Completely ignored by compiler

• Indentation is MEANINGLESS
What is Whitespace For?

- For Humans

- Try to make your code visually organized
- Use indentation to show logical flow
A good program

```plaintext
if (x == y)
{
    a = b+c;
}
else
{
    x = y+1;
}
```
Misuse of Whitespace!

Two bad examples of the same program:

```c
if (x == y) { a = b + c; } else { x = y + 1; }
```

```c
if (x == y) { a = b + c; } else { x = y + 1; }
```
Topic 02: Intro to Java

- Sight reading exercise
- Case Sensitivity & Whitespace
- **Identifiers** 2.1.3
- Comments 2.1.6
Identifiers

- Identifier is a name
  - Variable
  - Method
  - Class/Type

- Start with a letter
- Any number of letters, numbers, underscores
Identifier Examples

foo
bar
myMethod
index1
size12pt
widthInInches
COLUMNS_PER_PAGE
row1heightc
Remember Case Sensitivity?

- All of these are **different** identifiers!

  foo
  Foo
  F00
  f00
  F__0__0
Identifiers: Conventions

- Conventions make code easier to read
- For use by humans
- Compiler doesn't care
Conventions: Variable Names

• First letter lower case
• Camel case after that (capitalize first letter of each word)

• Examples
  
  foo
  numInputs
  userResponse
Conventions: Class Names

- First letter UPPER case
- Camel case after that (capitalize first letter of each word)

- Examples
  SightReading1
  String
  VmGUI
Conventions: Constants

- All caps, with underscores

- Examples
  - METERS_PER_MILE
  - MAX_ENTITIES
  - EXPANSION_THRESHOLD
Identifiers: Good Practice

• Use longer names – more descriptive

• Good
  
  weight
  numColumns

• Bad
  
  w
  nc
Keywords

- Words with special meaning
- Can never be used as identifiers

- Ones we've seen so far:
  
  if public static
  int class void
Topic 02: Intro to Java

- Sight reading exercise
- Case Sensitivity & Whitespace
- Identifiers 2.1.3
- Comments 2.1.6
Comments

• Completely ignored by the compiler

• Explain to other humans what you're doing and why
  – Your assumptions
  – Your intent
  – Anything tricky you did
The Value of Comments

What does this code do???

```c
if (t < 32)
    s = 0;
```
The Value of Comments

// if the temperature is less than freezing, then the player cannot move.

if (temp < 32)
    speed = 0;

Clear comment

Better variable names
Comments in Java

// ... one line comment

/* ... */ multi-line comment
One-Line Comments

// if the temperature is less than freezing, then the player cannot move.

if (temp < 32) speed = 0;

Three one-line comments
Multi-Line Comments

```java
/* if the temperature is less than freezing, then the player cannot move. */
if (temp < 32) {
    speed = 0;
}
```

One multi-line comment
Comment Style

- You are graded on code **clarity**
- Comments are required
- But I'm not a style Nazi
Recommended Comment Style

// use one-line for short comments

int i;  // OK to follow a line of code

/* use multi-line for longer.
  *
  * Line up asterisks to make it look nice.
  *
  * Put the close of the comment on its own line, like this:
  */
Topic 02: Intro to Java

- **Your First Java Program**
  - Declaring variables 2.1.1
  - `println()`
  - Assignment and basic arithmetic 2.1.4
  - `parseInt()` and `args[]`
- Comparison operators 3.2
- Logical operators 3.7
- Putting it all together
Your First Java Program

Q: How to get started writing a Java program?
A: Cut and paste!
public class SightReading1
{
    public static void main(String[] args)
    {
        System.out.println("Hello world!");

        int x;
        x = 1;

        int y;
        y = (x+1)*(x+1);

        if (x == y)
        {
            System.out.println("Math error");
        }
    }
}
Your First Java Program

```java
public class XXXXXX {
    public static void main(String[] args) {
        XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX XXXXXX
    }
}
```
Your First Java Program

public class MyClass
{
    public static void main(String[] args)
    {
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
        MyClass
    }
}

MUST match the filename!

MyClass.java
public class MyClass
Topic 02: Intro to Java

• Your First Java Program

• **Declaring variables** 2.1.1
  - `println()`

• Assignment and basic arithmetic 2.1.4
  - `parseInt()` and `args[]`

• Comparison operators 3.2

• Logical operators 3.7

• Putting it all together
Declaring an `int` Variable

```c
int foo;
```

- **Type:** `int`
- **Identifier:** `foo`
- **Don't forget the semicolon!**
What is `int`?

- Integer (a.k.a. whole number)
- 32 bits, signed

**Range:**
- Min: \(-2,147,483,648\)
- Max: \(+2,147,483,647\)
Where Are My Variables?

Your Computer's Memory:

Your computer's memory is **gigantic**. It is full of little boxes, each the right size for a single variable.
Where Are My Variables?

Your Variables:

You don't know where your variables are.

You simply tell Java the type and name of the variable you want – and it keeps track of where they are.
Declaring a **double** Variable

```java
double bar;
```

- **Type:** `double`
- **Identifier:** `bar`

- Don't forget the semicolon!
What is **double**?

- Floating point number
  - 1.5
  - .00000123
  - 12345000000000000000000

- 64 bits (IEEE 754)
  - Basically, scientific notation

- Warning: Rounding!
So Why Use `double`?

- If you need fractions
- If you need huge numbers
- If you need tiny numbers

- Just remember that all `double` answers are approximate!

- Avoid `double` when you can. Use `int` instead.
Topic 02: Intro to Java

- Your First Java Program
- Declaring `int` and `double` variables 2.1.1
- `println()`
- Assignment and basic arithmetic 2.1.4
- `parseInt()` and `args[]`
- Comparison operators 3.2
- Logical operators 3.7
- Putting it all together
println()

- `println()` prints 1 line of output
- Use strings for simple messages
- Use + to chain things together
- **You MUST** use the full name:
  
  ```java
  System.out.println("foo");
  ```
println() Examples

Code
System.out.println(“Hello world!”);

Output:
Hello world!
Warning

- You **MUST** use double-quotes!
- Single quotes do something else!

**This works:**

```java
System.out.println("Hello world!");
```

**This does not:**

```java
System.out.println('Hello world!');
```
println() Examples

**Code**

```java
int foo = 1;
System.out.println("foo is \"+foo\")
```

**Output:**

foo is 1
println() Examples

Code

```java
int foo = 1;
int bar = 2;
System.out.println("foo is "+foo);
System.out.println("bar is "+bar);
```

Output:

foo is 1
bar is 2
print() (Buggy Example)

- Just like println(), except no newline

**Code**

```java
System.out.print("Hello");
System.out.print("world!");
System.out.println();
```

**Output:**

Helloworld!
**print() (Fixed)**

- Just like `println()`, except no newline

**Code**

```java
System.out.print("Hello ");
System.out.print("world!");
System.out.println();
```

**Output:**

Hello world!
print() Example

Code
System.out.print("foo is " + foo);
System.out.println("", bar is " + bar);

Output after 1st line:
f

Output after 2nd line:
foo is 1, bar is 2
printf()

• Experienced programmers can use `printf()`
• Works just like C
  
  ```java
  System.out.printf("%d\n", foo);
  System.out.printf("%d %d\n", foo, bar);
  ```

• Details later
Topic 02: Intro to Java

- Your First Java Program
- Declaring \texttt{int} and \texttt{double} variables 2.1.1
- \texttt{println()}
- \textbf{Assignment and basic arithmetic} 2.1.4
- \texttt{parseInt()} and \texttt{args[]} 
- Comparison operators 3.2
- Logical operators 3.7
- Putting it all together
Assignment

\[ a = b; \]

- Reads the value from variable \( b \)
- Stores the value into variable \( a \)

- Don't forget the semicolon!
Rules of Assignment

• Variables must be the same type
  – (there are exceptions)

• Left side must be a simple variable

• Right side can be a complex expression
Assignment Examples

```c
int x, y, z;
x = 0; // set var to a value
x = y;  // copy value from y to x
```
Left Side Must Be Simple

**Legal:**

\[ x = y+1; \]

**NOT Legal:**

\[ y+1 = x; \]
Arithmetic Operators

+  addition
−  subtraction
*  multiplication
/  division (WARNING: rounds int DOWN)
%  modulo
int x, y, z;

x = x+1;  // increment x

y = (z+1)*(z+1);
Topic 02: Intro to Java

- Your First Java Program
- Declaring int and double variables 2.1.1
- println()
- Assignment and basic arithmetic 2.1.4
- parseInt() and args[]
- Comparison operators 3.2
- Logical operators 3.7
- Putting it all together
public class MultiplyTheArgs 
{
    public static void main(String[] args) 
    {
        int x;
        int y;
        int z;

        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);
        z = x * y;

        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("z = " + z);
    }
}

http://www.cs.arizona.edu/classes/cs127a/spring16/slides/class_exercises/
public class MultiplyTheArgs {
    public static void main(String[] args) {
        int x;
        int y;
        int z;

        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);
        z = x * y;

        System.out.println("x = "+x);
        System.out.println("y = "+y);
        System.out.println("z = "+z);
    }
}
public class MultiplyTheArgs
{
    public static void main(String[] args)
    {
        int x;
        int y;
        int z;

        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);
        z = x * y;

        System.out.println("x = "+x);
        System.out.println("y = "+y);
        System.out.println("z = "+z);
    }
}
public class MultiplyTheArgs
{
    public static void main(String[] args)
    {
        int x;
        int y;
        int z;

        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);
        z = x * y;

        System.out.println("x = "+x);
        System.out.println("y = "+y);
        System.out.println("z = "+z);
    }
}
public class MultiplyTheArgs
{
    public static void main(String[] args)
    {
        int x;
        int y;
        int z;

        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);
        z = x * y;

        System.out.println("x = " + x);
        System.out.println("y = " + y);
        System.out.println("z = " + z);
    }
}
public class MultiplyTheArgs
{
    public static void main(String[] args)
    {
        int x;
        int y;
        int z;

        x = Integer.parseInt(args[0]);
        y = Integer.parseInt(args[1]);
        z = x * y;

        System.out.println("x = "+x);
        System.out.println("y = "+y);
        System.out.println("z = "+z);
    }
}
Topic 02: Intro to Java

- Your First Java Program
- Declaring `int` and `double` variables 2.1.1
- `println()`
- Assignment and basic arithmetic 2.1.4
- `parseInt()` and `args[]`
- **Comparison operators** 3.2
- Logical operators 3.7
- Putting it all together
Comparison Operators

==  equals
!=  not equals
<  less than
<=  less than or equals
>  greater than
>=  greater than or equals
Comparison Operator Examples

if (x == y)
  ...
if (x < 0)
  ...
if (0 <= x)
  ...
if (x+1 > y+2)
  ...
if (x*x != x+y)
  ...

...
Comparison Operator Examples

```java
if (x == y) {
    ...
}
if (x < 0) {
    ...
}
if (0 <= x) {
    ...
}
if (x+1 > y+2) {
    ...
}
if (x*x != x+y) {
    ...
}
```

Compare a variable to another variable.
Comparison Operator Examples

if (x == y)
...
if (x < 0)
...
if (0 <= x)
...
if (x+1 > y+2)
...
if (x*x != x+y)
...
Comparison Operator Examples

```plaintext
if (x == y)
...
if (x < 0)
...
if (0 <= x)
...
if (x+1 > y+2)
...
if (x*x != x+y)
...
```

== for comparison
=
for assignment
Comparison Operator Examples

```
if (x == y)
  ...
if (x < 0)
  ...
if (0 <= x)
  ...
if (x+1 > y+2)
  ...
if (x*x != x+y)
  ...
```

Complex expressions on the left side are perfectly OK!
Comparison Operator Examples

if (x == y)
   ...
if (x < 0)
   ...
if (0 <= x)
   ...
if (x+1 > y+2)
   ...
if (x*x != x+y)
   ...

OK for the same variable to show up on both sides
Topic 02: Intro to Java

- Your First Java Program
- Declaring `int` and `double` variables 2.1.1
- `println()`
- Assignment and basic arithmetic 2.1.4
- `parseInt()` and `args[]`
- Comparison operators 3.2
- Logical operators 3.7
- Putting it all together
Logical Operators

\& \& \hspace{1cm} \text{AND}

\| \| \hspace{1cm} \text{OR}

! \hspace{1cm} \text{NOT}
Logical Operator Examples

```java
if (x == y && y == z)
...
if (x < 0 || z != 1)
...
if (!(0 == x))
...
```

this AND that
Logical Operator Examples

if (x == y && y == z)
...
if (x < 0 || z != 1)
...
if (! (0 == x))
Logical Operator Examples

```c
if (x == y && y == z)
    ...

if (x < 0 || z != 1)
    ...

if (!((0 == x))
    ...
```

NOT that
Logical Operator Examples

```plaintext
if (x == y && y == z)
...
if (x < 0 || z != 1)
...
if (!(0 == x))
...
```

Same as

```plaintext
0 != x
```
Topic 02: Intro to Java

• Your First Java Program
• Declaring `int` and `double` variables 2.1.1
• `println()`
• Assignment and basic arithmetic 2.1.4
• `parseInt()` and `args[]`
• Comparison operators 3.2
• Logical operators 3.7
• Putting it all together
Putting it All Together

Remember the “sort3” example?

• Input:
  – 3 integers, called a, b, c

• Output:
  – All three integers in order, with their names
  – If there is a tie, then always list them in name order: a, b, c

Let's implement this in Java!
Putting it All Together

- What are our steps, at a very high level?

- What form will our program take – a loop, or a single pass?
Putting it All Together

- What are our steps, at a very high level?
  - Read input
  - Do comparisons
  - Print out results

- What form will our program take – a loop, or a single pass?
  - Single pass (only need to compare 3 numbers)
Putting it All Together

Remember this?

```plaintext
if (a is minimum)
    do all printouts
else
    if (b is minimum)
        do all printouts
    else
        do all printouts
```

This is the high-level pseudocode we wrote in slide deck 01.

Remember, you should not turn in pseudocode like this. But it's a good starting point!
Putting it All Together

Remember this?

```python
if (a <= b && a <= c)
    print "a: "+a
    if (b <= c)
        print "b: "+b
        print "c: "+c
    else
        print "c: "+c
        print "b: "+b
else ...
```

This is the low-level pseudocode we also wrote, showing what happens if a is the minimum.

We can copy this (with small changes) to work for the other 2 cases.
Putting it All Together

**Pseudocode:**

\[
\text{if } (a \leq b \text{ && } a \leq c)
\]

**Java:**

```java
if (a <= b && a <= c)
{
    ...
}
```

Let's convert the pseudocode to Java!
Putting it All Together

**Pseudocode:**

print "a: " + a

**Java:**

```java
System.out.println("a: "+a);
```

Let's convert the pseudocode to Java!
Putting it All Together

**Pseudocode:**

if (test)
    stmt1
else
    stmt2

**Java:**

if (test)
{
    stmt1
}
else
{
    stmt2
}
Putting it All Together

Pseudocode:
if (a <= b && a <= c)
    print "a: " + a
if (b <= c)
    print "b: " + b
    print "c: " + c
else
    print "c: " + c
    print "b: " + b
else ...

Java:
if (a <= b && a <= c)
{
    S.o.p("a: "+a);
    if (b <= c)
    {
        S.o.p("b: "+b);
        S.o.p("c: "+c);
    }
    else
    {
        S.o.p("c: "+c);
        S.o.p("b: "+b);
    }
}
else ...

Putting it All Together

Java:
if (a <= b && a <= c) {
    S.o.p("a: "+a);
    if (b <= c) {
        S.o.p("b: "+b);
        S.o.p("c: "+c);
    } else {
        S.o.p("c: "+c);
        S.o.p("b: "+b);
    }
} else ...

WARNING:
S.o.p is a common abbreviation for System.out.println.

I occasionally use it on slides, but I will not accept it on quizzes or exams!
Putting it All Together

Time to add some missing pieces:

- Wrap it all in a class and a `main()` method
- Declare variables
- Read input
public class PuttingItTogether {
    public static void main(String[] args) {
        ...
    }
}
public class PuttingItTogether
{
    public static void main(String[] args)
    {
        int a;
        int b;
        int c;

        ...

    }
}

Next we declare our variables.

The problem didn't explicitly say that we would only use integers. But let's assume that for simplicity.

We could change it later to use double, if we need.
public class PuttingItTogether
{
    public static void main(String[] args)
    {
        int a;
        int b;
        int c;

        a = Integer.parseInt(args[0]);
        b = Integer.parseInt(args[1]);
        c = Integer.parseInt(args[2]);

        ...
    }
}
public class PuttingItTogether {
    public static void main(String[] args) {
        int a;
        int b;
        int c;

        a = Integer.parseInt(args[0]);
        b = Integer.parseInt(args[1]);
        c = Integer.parseInt(args[2]);

        if (a <= b && a <= c) {
            ...
        } else {
            if (b <= c) {
                ...
            } else {
                ...
            } } 
    }
}

Next, start copying over the design from the pseudocode.

Well written pseudocode should be easy to translate into Java. If it's not easy, then you need better pseudocode!

Final version is at: http://www.cs.arizona.edu/classes/cs127a/spring16/slides/class_exercises/
Topic 02: Intro to Java

- Your First Java Program
- Declaring `int` and `double` variables  2.1.1
- `println()`
- Assignment and basic arithmetic  2.1.4
- `parseInt()` and `args[]`
- Comparison operators  3.2
- Logical operators  3.7
- Putting it all together

Summary