Panel 1

Last time

Declaring variables

Variable names

Assigning values

Basic operations +, -, *, /, and %

x % y is remainder after dividing x by y as int.

Ex: 5 % 2 = 1

15 % 4 = 3

1

Panel 2

Homework

(1) What does the % operator do?

(2) What is the value of z:

```
int x = 5;
int y = 2;
int z = y / x;   0
```

Explain!

Rule: if both arguments are int, answer is int (!)

if one (or both) are double, answer is double (good)

2

Panel 3

Suppose we have variables declared and initialized as follows:

```
int i1 = 10, i2 = 3;
double x1 = 8.0, x2 = 5.0;
```

What are the resulting values and types of the computations:

→ `i1 * i2;` 30

`x1 / x2;` 1.6

`i1 / i2;` 3

`i1 - x1;` 2.0

`i1 % i2;` 1

`i1 % x1;` 2.0

3

Panel 4

Short cut Notation

Frequently want to: "add 5 to x"

$x = x + 5$ is not equation, because = is assignment

If $x = 2$ and I run $x = x + 5$ (first) then x is 7

Short-cut: ✓ $x += 5$ ⇔ $x = x + 5$ So C++

✓ $x -= 5$ $x = x - 5$

$x *= 5$ $x = x * 5$

$x /= 5$ $x = x / 5$

"one better than C"

Worse: $x{+}{+}$ ⇒ $x = x + 1$ (increment)

$x{-}{-}$ ⇒ $x = x - 1$ (decrement)

4

Panel 5

What is the value of z at the end of this code fragment?

```
int z = 23;
int x = 7, y = 53;
x *= y;
y /= 2;
x++;
z += (x + y);
```

x = x * y = 7 * 53 = 371

y = y/2 = 53/2 = 26

x = x + 1 = 372

z = z + (x + y) = 398 + 23 = 421

x = 372

y = 26

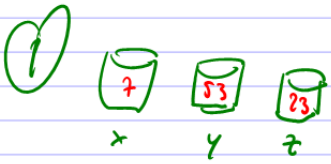
z = 23

```
int x
System.out.println(x)
```

a) 83

b) 398

c) 378



5

Panel 6

Advanced Arithmetic for Numeric Types

To use a mathematical function, preface it by the word `Math`, *followed by a dot, and the function's name. Each function returns the type* `double` *unless otherwise indicated.*

Trig functions and their inverses	
`cos(number)`	cosine of `number` (input in radians)
`sin(number)`	sine of `number` (input in radians)
`tan(number)`	tangent of `number` (input in radians)
`acos(number)`	arc (inverse) cosine of `number`, between 0.0 and Pi
`asin(number)`	arc (inverse) sine of `number`, between -Pi/2 and Pi/2
`atan(number)`	arc (inverse) tangent of `number`, between -Pi/2 and Pi/2
`atan2(a, b)`	angle between x-axis and the vector (b, a), between –Pi and Pi.
`exp(number)`	e = 2.718... raised to the power of `number`
`log(number)`	Natural logarithm (base e) of `number`

Ex: Find sin of 2.0

```
double t = Math.sin(2.0)
```

Ex: Find cos of π

```
Math.cos(Math.PI)
```

Other functions	
`abs(number)`	Absolute value of `number`; returns same type as `number`
`max(a, b)`	Greater value of `a` or `b`; returns same type as input values
`min(a, b)`	Smaller value of `a` or `b`; returns same type as input values
`pow(base, exp)`	The number `base` raised to the power of `exp`
`sqrt(number)`	Square root of `number`
`random()`	A random number between 0.0 and 1.0
`ceil(number)`	Smallest (closest to negative infinity) value that is not less than `number` and is equal to a mathematical integer
`floor(number)`	Largest (closest to positive infinity) value that is not greater than `number` and is equal to a mathematical integer
`round(number)`	The value of `number` rounded to the nearest integer value. Returns `int` if `number` is `float`, and `long` if `number` is `double`

Ex: $2^{16} - 1$

```
Math.pow(2,16) - 1
```

Mathematical Constants	
`Math.E`	The value that is closer than any other to e, the base of the natural logarithms.
`Math.PI`	The value that is closer than any other to π, the ratio of the circumference of a circle to its diameter.
`Double.POSITIVE_INFINITY`	A representation of positive infinity of type double
`Double.NEGATIVE_INFINITY`	A representation of negative infinity of type double
`Double.NaN`	A representation of "Not-a-Number"

6

Panel 7

Example:
Write some Java code segments to compute:

1. **The circumference and area of a circle with radius `3.0`.**

2. **The length of the hypotenuse of a right triangle given its base and height.**

3. **The angle in degrees, given that `rAngle` is the angle in radians.**

4. **The angle in radians, given that `dAngle` is the angle in degrees**

5. **What is `exp(4000.0)`? What is `10.0/0.0`? What is `-10.0/0.0`? What is `0.0/0.0`?**

7

Panel 8

Logical Operators and Examples

Logical variables can not be added, subtracted, etc.

Instead:

and	&&
or	||
not	!

Ex: boolean x = true;
boolean y = false;

Sys... println (x && y) F

println ((!x) || (!y))
F || T True

Comparison Operators

<, >, <=, >=

== is equal

!= not equal

8

Panel 9

Recall logic tables:

and	x	y	x && y
	T	T	T
	T	F	F
	F	T	F
	F	F	F

or	x	y	x || y
	T	T	T
	T	F	T
	F	T	T
	F	F	F

9

Panel 10

1. Determine if the following statements are `true` or `false`, using:

```
int x = 5, y = 9, z = 2;
boolean r = false, result;
```

a. `result = ( 5 >= x);` true

c. `result = ( y < z );` false

c. `result = (y == 9) && ( (x >= 2) || r );`

c. `result = !( ( r || ( y < x ) ) && ( 5 >= x ) );`

c. `result = ( ( x + y ) >= ( 16 - z ) );`

c. `result = ( x == ( (z++) + 2 ) );`

c. `result = ( x == ( (++z) + 2 ) );`

c. `result = ( !r && ( (--y) > (z + x) ) );`

c. `result = (((21 - z) == (((y++) + x + z) + 1)) && (!r == true));`

10

HW