Museum Lab Tutoring Schedule

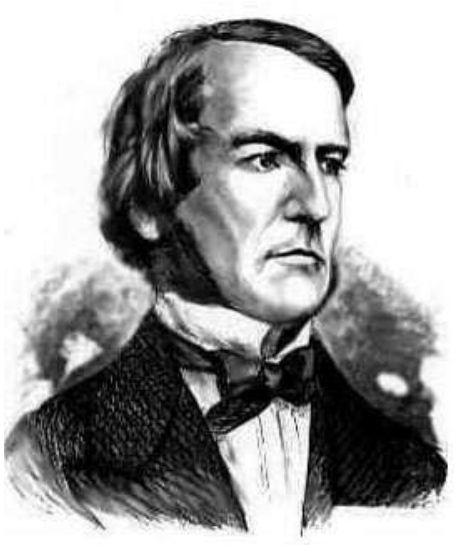
Fall 2011

	Monday	Tuesday	Wednesday	Thursday	Friday
	Open 10-2	Open 10-2	Open 10-2	Open 10-2	Open 10-2
10 – 11	Tyler Lee	Chris Tenda	Zach Wormgoor	Chris Tenda	Tyler Lee
11 - 12	Jordon Yates	Jordon Yates	Zach Wormgoor	Jordon Yates	Zach Wormgoor
12 - 1	Matt Morris	Jordon Yates	Zach Wormgoor	Jordon Yates	Matt Morris
1 - 2	Matt Morris	Matt Morris	Matt Morris	Tyler Lee	Chris Tenda

Recommended practice programs

- http://introcs.cs.princeton.edu/java/12types/
- Creative Exercises (with solutions):
  25. Wind chill
  29. Day of week
  30. Uniform random numbers
- Web Exercises (with solutions):
  1. Distance
  20. Divisibility

Variables, comparisons, main()



logical AND	logical OR	logical NOT
&&	||	!

```
public static void main(String [] args)
```

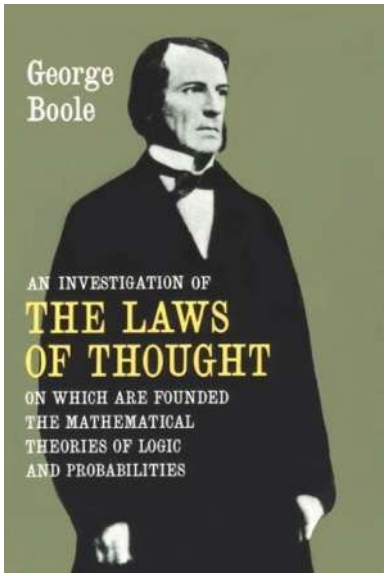
Variables and data types

- Last time: **`String`**, **`int`**, and **`double`**

Java built-in type	what it stores	example values	operations
int	integer values	42 1234	add, subtract, multiply, divide, remainder
double	floating-point values	9.95 3.0e8	add, subtract, multiply, divide
boolean	truth values	true false	and, or, not
char	characters	'a', 'b', '!'	compare
String	sequence of characters	"Hello world!" "I love this class!"	concatenate

Booleans

- **`boolean`** data type
  - Either `true` or `false`
  - Controls logic and flow of control in programs
  - Operations:



logical AND	logical OR	logical NOT
`&&`	`||`	`!`

Booleans

- **`boolean`** data type

logical AND	logical OR	logical NOT
`&&`	`||`	`!`

`!a` → "Is `a` set to false?"
`a && b` → "Are both `a` *and* `b` set to true?"
`a || b` → "Is either `a` *or* `b` (or both) set to true?"

a	!a
true	false
false	true

a	b	a && b	a || b
false	false	false	false
false	true	false	true
true	false	false	true
true	true	true	true

Comparisons

- Given two numbers → return a **`boolean`**

operator	meaning	true example	false example
==	equal	`7 == 7`	`7 == 8`
!=	not equal	`7 != 8`	`7 != 7`
<	less than	`7 < 8`	`8 < 7`
<=	less than or equal	`7 <= 7`	`8 <= 7`
>	greater than	`8 > 7`	`7 > 8`
>=	greater than or equal	`8 >= 2`	`8 >= 10`

Is the sum of `a`, `b` and `c` equal to `0`?
```
(a + b + c) == 0
```

Is `grade` in the B range?
```
(grade >= 80.0) && (grade < 90.0)
```

Is `sumItems` an even number?
```
(sumItems % 2) == 0
```

Leap year

- Years divisible by 4 but not 100 → leap year
- Years divisible by 400 → leap year

```
public class LeapYear
{
   public static void main(String [] args)
   {
      int year = Integer.parseInt(args[0]);
      boolean isLeapYear;
      // Leap year if divisible by 4 by not 100
      isLeapYear = (year % 4 == 0) && (year % 100 != 0);
      // But also leap year if divisible by 400
      isLeapYear = isLeapYear || (year % 400 == 0);
      System.out.println(isLeapYear);
   }
}
```

```
% java LeapYear 2000
true
```

Characters

- **`char`** data type
  - Holds a single character
  - Single apostrophe, e.g. 'a', 'z'

```
public class CharExample
{
   public static void main(String [] args)
   {
      char ch1 = 'y';
      char ch2 = 'o';
      String result = "" + ch1;
      result += ch2;
      result += ch2;
      result += ch2;
      System.out.println(result);
   }
}
```

```
% java CharExample
yooo
```

main method

```
public class CostCalc
{
    public static void main(String [] args)
    {

    }
}
```

All the action goes here (for now)

Extra things from the command line

Allows programs output to depend on its input

```
% java CostCalc bananas 12 0.21
To buy 12 bananas you will need $2.52
```

args array

```
public static void main(String [] args)
```

```
% java CostCalc bananas 12 0.21
To buy 12 bananas you will need $2.52
```

identifier	meaning	value	type
`args[0]`	1st thing on command line after Java class name	`"bananas"`	`String`
`args[1]`	2nd thing on command line	`"12"`	`String`
`args[2]`	3rd thing on command line after Java class	`"0.21"`	`String`
`args.length`	# of things on command line	`3`	`int`

Static methods

- Java has lots of "helper" methods
  - Math functions
  - Random number generation
  - Type conversion: `String` → `int`
    `String` → `double`
- For now, we'll stick to `static` methods
- Methods live in a Java class
  - e.g. `Math`, `Integer` or `Double`
  - Call using class name followed by dot
- Methods take value(s) and return one

A few of Java's Math methods

method	description
`double abs(double a)`	absolute value of a
`double max(double a, double b)`	maximum of a and b
`double min(double a, double b)`	minimum of a and b
`double sqrt(double a)`	square root of a
`double pow(double a, double b)`	raise a to the b^{th} power
`long round(double a)`	round a to the nearest integer
`double random()`	random number in [0, 1)

```
double a = 3.14;
double b = -1.0;
System.out.println("abs(b) = "    + Math.abs(b));
System.out.println("min(a, b) = " + Math.min(a, b));
System.out.println("random() = "  + Math.random());
```

```
abs(b) = 1.0
min(a, b) = -1.0
random() = 0.11515749532842645
```

Converting text

method	description
`Integer.parseInt(String a)`	converts text `a` into an integer
`Double.parseDouble(String a)`	convert text `a` into a double

```
public class CostCalc
{
   public static void main(String [] args)
   {
      String product = args[0];
      int    qty     = Integer.parseInt(args[1]);
      double cost    = Double.parseDouble(args[2]);

      double total = qty * cost;

      System.out.print("To buy " + qty);
      System.out.print(" " + product);
      System.out.println(" you will need $" + total);
   }
}
```

```
% java CostCalc elections 2 1e6
To buy 2 elections you will need $2000000.0
```

Errors

runtime errors

```
% java CostCalc apples 6 -10
To buy 6 apples you will need $-60.0

% java CostCalc apples 6 foo
Exception in thread "main" java.lang.NumberFormatException: For input
string: "foo"
        at sun.misc.FloatingDecimal.readJavaFormatString
        (FloatingDecimal.java:1222)
        at java.lang.Double.parseDouble(Double.java:510)
        at CostCalc.main(ArgsExample.java:7)

% java CostCalc apples 6.0 0.25
Exception in thread "main" java.lang.NumberFormatException: For input
string: "6.0"
        at java.lang.NumberFormatException.forInputString
        (NumberFormatException.java:48)
        at java.lang.Integer.parseInt(Integer.java:458)
        at java.lang.Integer.parseInt(Integer.java:499)
        at CostCalc.main(ArgsExample.java:6)
```

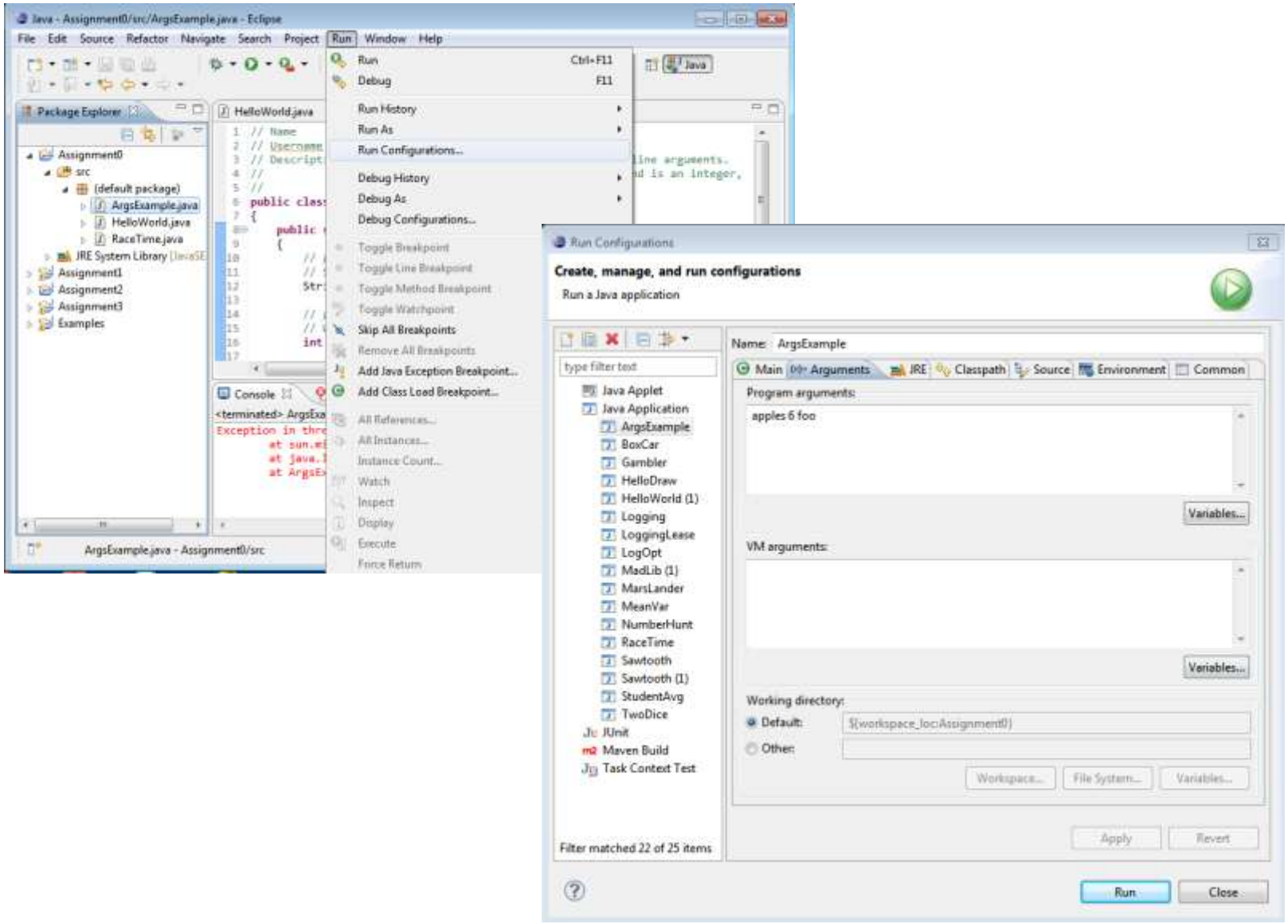
Errors

compile time error

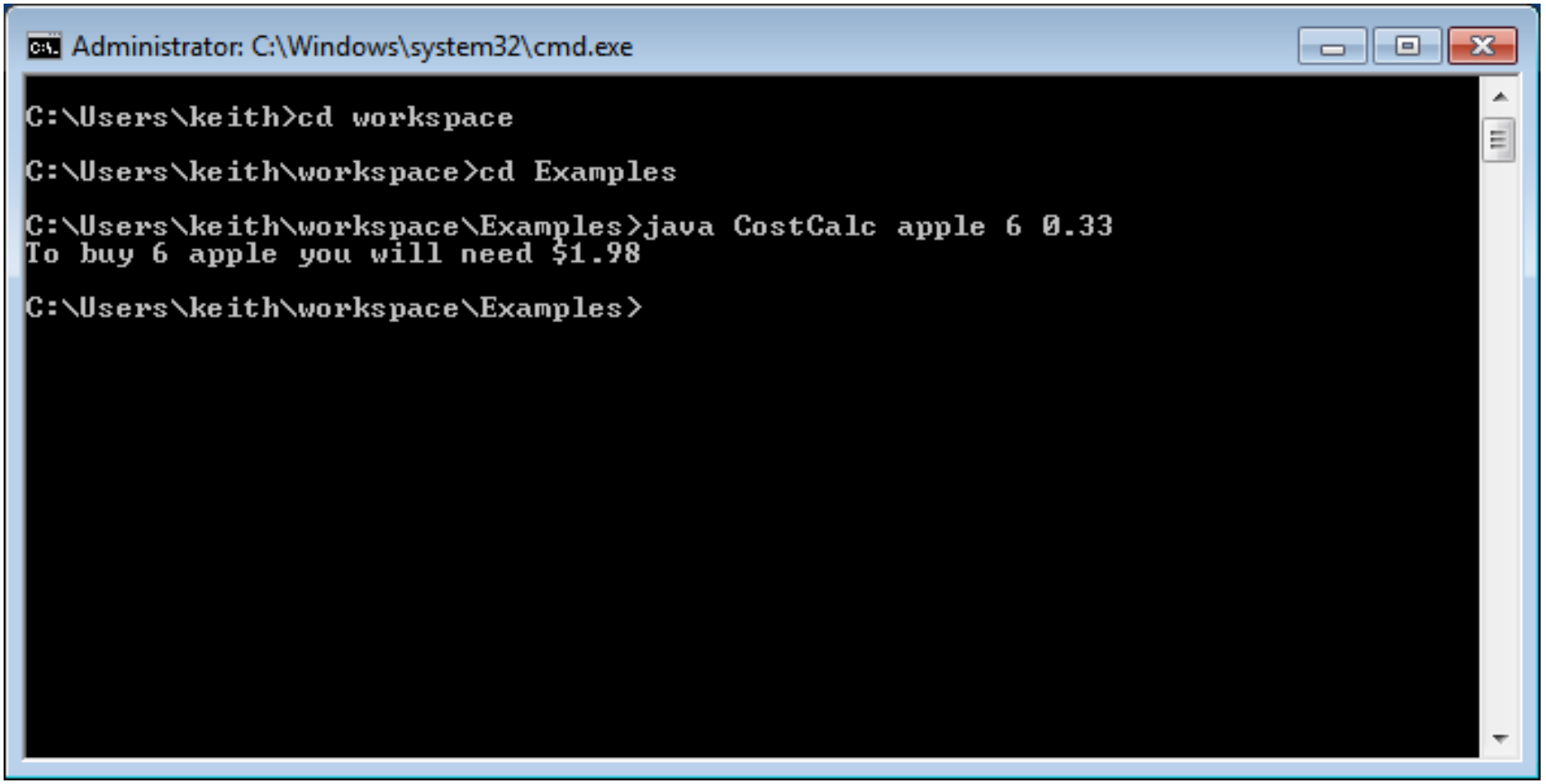
```
public class CostCalc
{
    public static void main(String [] args)
    {
        String  product = args[0];
        int     qty     = args[1];
        double  cost    = args[2];
        double  total   = qty * cost;
        System.out.print("To buy " + qty);
        System.out.print(" " + product);
        System.out.println(" you will need $" + total);
    }
}
```

```
% javac CostCalc.java
CostCalc.java:6: incompatible types
found   : java.lang.String
required: int
        int     qty     = args[1];
                          ^
CostCalc.java:7: incompatible types
found   : java.lang.String
required: double
        double  cost    = args[2];
                          ^
2 errors
```

Command line args in Eclipse



Command line args in command shell



Type conversion quiz

- Automatic: no loss of precision
  - `int` will convert to a `double` if need be
  - `double` cannot automatically convert to `int`
- Manual: cast or using a method

expression	resulting type	resulting value
`(int) 3.14159`		
`Math.round(3.6)`		
`2 * 3.0`		
`2 * (int) 3.0`		
`(int) 2 * 3.0`		

Type conversion quiz

- Automatic: no loss of precision
  - `int` will convert to a `double` if need be
  - `double` cannot automatically convert to `int`
- Manual: cast or using a method

expression	resulting type	resulting value
`(int) 3.14159`	`int`	`3`
`Math.round(3.6)`	`long`	`4`
`2 * 3.0`	`double`	`6.0`
`2 * (int) 3.0`	`int`	`6`
`(int) 2 * 3.0`	`double`	`6.0`

String conversion quiz

- `String` conversion, using:
  - `Integer.parseInt()`
  - `Double.parseDouble()`

expression	resulting type	resulting value
`Integer.parseInt("30")`		
`Double.parseDouble("30")`		
`Integer.parseInt("30.1")`		
`Double.parseDouble("30.1")`		
`Integer.parseInt("$30")`		
`Double.parseDouble(3.14)`		

String conversion quiz

- `String` conversion, using:
  - `Integer.parseInt()`
  - `Double.parseDouble()`

expression	resulting type	resulting value
`Integer.parseInt("30")`	`int`	`30`
`Double.parseDouble("30")`	`double`	`30.0`
`Integer.parseInt("30.1")`	(error, can't parse as int)	
`Double.parseDouble("30.1")`	`double`	`30.1`
`Integer.parseInt("$30")`	(error, can't parse as int)	
`Double.parseDouble(3.14)`	(error, 3.14 not a String)	

String concatenation quiz

- + is addition for numeric types
- + is concatenation for `String` type
- numeric types convert to `String` if needed
  - Strings never (automatically) goes back to number

expression	resulting type	resulting value
`"testing " + 1 + 2 + 3`		
`"3.1" + 4159`		
`"2" + " + " + "3"`		
`1 + 2 + 3 + "66"`		

String concatenation quiz

- + is addition for numeric types
- + is concatenation for `String` type
- numeric types convert to `String` if needed
  - Strings never (automatically) go back to numeric

expression	resulting type	resulting value
`"testing " + 1 + 2 + 3`	`String`	`"testing 123"`
`"3.1" + 4159`	`String`	`"3.14159"`
`"2" + " + " + "3"`	`String`	`"2 + 3"`
`1 + 2 + 3 + "66"`	`String`	`"666"`

Randomness

- Simulate roll of two 6-sided dice
- Generate two random #'s between 1 and 6

`Math.random()` → number in [0, 1.0)
e.g. 0.0, 0.312, 0.9999999

`Math.random()*6` → number in [0, 6.0)
e.g. 0.0, 1.872, 5.9999994

`(Math.random()*6)+1` → number in [1, 7.0)
e.g. 1.0, 2.872, 6.9999994

`(int)(Math.random()*6)+1` → number in set {1, 2, 3, 4, 5, 6}
e.g. 1, 2, 6

Randomness

- Simulate roll of two 6-sided dice
- Generate two random #'s between 1 and 6

```
public class TwoDice
{
   public static void main(String [] args)
   {
      int dice1 = (int) (Math.random() * 6) + 1;
      int dice2 = (int) (Math.random() * 6) + 1;
      int sum   = dice1 + dice2;

      System.out.println(dice1 + " + " +
                         dice2 + " = " +
                         sum);
   }
}
```

Summary

- Variables
  - Allows us to store and compute on data
  - `String`, `int`, `double`, `boolean`, `char`
  - Boolean operators for logic and program flow control (more on this next time!)
- Static methods for math, text conversion
- args array for reading input
- Type conversion
  - Automatic and explicit via casting/methods
  - Important: prevent bugs, useful for dice rolling