Data Types

Java supports two main types of data: ***primitive types*** like `int` and `double` that represent a single value, and ***reference types*** like `String` and `Scanner` that represent more complex data.

Manager:		Recorder:	
Presenter:		Reflector:	

Content Learning Objectives

After completing this activity, students should be able to:

- Explain how using roles improves the team's success.
- Name Java's primitive data types and give examples of each one.
- Identify illegal assignment statements, and explain why they are illegal.
- Describe what it means for variables to store a reference to an object.

Process Skill Goals

During the activity, students should make progress toward:

- Providing feedback on how well other team members are working. (Teamwork)

Facilitation Notes

The meta activity reinforces the importance of roles. Successful teams are able to accomplish all the tasks outlined on the Role Cards, and there's too many things for one person to keep track. Ask the reflectors to pay special attention to their role during today's activity, and invite them at some point to report to share what they have observed.

On #6, students may struggle with letters used in primitive values (i.e., F and L). Rather than just say what everything means, direct students back to Model 1 and have them read out loud the type names. #7 is good for report-out, allowing students to collectively develop a rule for when an assignment is allowed. The solution files *Assign1.java* and *Assign2.java* are available for demonstration. When reporting out, introduce the term ***literal***.

Model 2 introduces a technique for drawing memory diagrams to show the difference between primitive and reference types. On #15, have multiple teams draw their diagrams on the board. Address misconceptions that arise (e.g., drawing multiple string objects). It might be helpful at this point to direct students to Java Visualizer, which draws similar diagrams. Select the option "Show String/Integer/etc objects, not just values" for it to display references as arrows.

Key questions: #7, #13, #15

Source files: *TaylorSwift.java*

Meta Activity: Team Disruptions

Common disruptions to learning in teams include: talking about topics that are off-task, teammates answering questions on their own, entire teams working alone, limited or no communication between teammates, arguing or being disrespectful, rushing to complete the activity, not being an active teammate, not coming to a consensus about an answer, writing incomplete answers or explanations, ignoring ideas from one or more teammates.

Questions (10 min) **Start time:**

1. Pick four of the disruptions listed above. For each one, find something from the role cards that could help improve the team's success. Use a different role for each disruption.

a) Manager:

limited communication between teammates

b) Presenter:

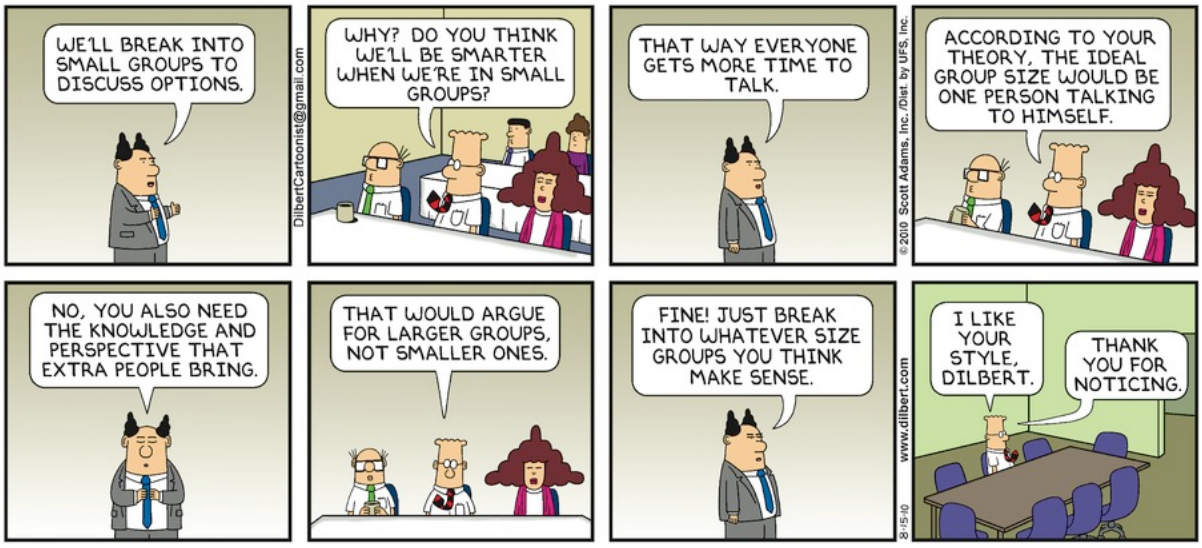
ignoring ideas from one or more teammates

c) Recorder:

writing incomplete answers or explanations

d) Reflector:

teammates answering questions on their own



Dilbert by Scott Adams. © Andrews McMeel Syndication. http://dilbert.com/strip/2010-08-15

Model 1 Primitive Types

Keyword	Size	Min Value	Max Value	Example
`byte`	1 byte	-128	127	`(byte) 123`
`short`	2 bytes	$-32,768$	$32,767$	`(short) 12345`
`int`	4 bytes	-2^{31}	$2^{31}-1$	`1234567890`
`long`	8 bytes	-2^{63}	$2^{63}-1$	`123456789012345L`
`float`	4 bytes	-3.4×10^{38}	3.4×10^{38}	`3.14159F`
`double`	8 bytes	-1.8×10^{308}	1.8×10^{308}	`3.141592653589793`
`boolean`	1 byte	N/A	N/A	`true`
`char`	2 bytes	0	65,535	`'A'`

Note that 1 byte is 8 bits, i.e., eight "ones and zeros" in computer memory. Since there are only two possible values for each bit, you can represent $2^8 = 256$ possible values with 1 byte.

Questions (15 min) Start time:

2. Which of the primitive types are integers? Which are floating-point?

Integers: byte, short, int, long. Floating-point: float, double.

3. Why do primitive types have ranges of values? What determines the range of the data type?

The range of values depends on the size, i.e., how many bytes are used to store the value.

4. Why can't computers represent every possible number in mathematics? Will they ever be able to do so?

Computers have finite memory, but there are an infinite number of numbers. There will always be a number larger than what computers can store.

5. Since a `byte` can represent 256 different numbers, why is its max value 127 and not 128?

One of the 256 values is the number zero. So 128 negatives, plus 1 zero, plus 127 positives equals 256 values.

6. What is the data type for each of the following values?

1.14159	double	7.2E-4	double	-128	int
0	int	0.0	double	'0'	char
-1.0F	float	-13L	long	false	boolean
123	int	'H'	char	true	boolean

7. Based on the examples below, when does Java allow you to assign one type of primitive variable to another?

```
int int_ = 3;
long long_ = 3L;
float float_ = 3.0F;
double double_ = 3.0;

int_ = int_;
int_ = long_;      // illegal
int_ = float_;     // illegal
int_ = double_;    // illegal

long_ = int_;
long_ = long_;
long_ = float_;    // illegal
long_ = double_;   // illegal
```

```
float_ = int_;
float_ = long_;
float_ = float_;
float_ = double_;  // illegal

double_ = int_;
double_ = long_;
double_ = float_;
double_ = double_;

int_ = '0';
int_ = false;      // illegal
double_ = '0';
double_ = false;   // illegal
```

The types have to be compatible (e.g., you can't assign numeric to boolean), and you can only assign from smaller to larger (e.g., from float to double, or int to double).

8. Given the following variable declarations, which of the assignments are not allowed?

```
byte miles;
short minutes;
int checking;
long days;
float total;
double sum;
boolean flag;
char letter;
```

```
checking = 56000;
total = 0;
sum = total;
total = sum;
checking = miles;
sum = checking;
flag = minutes;
days = '0';
```

All are okay except:

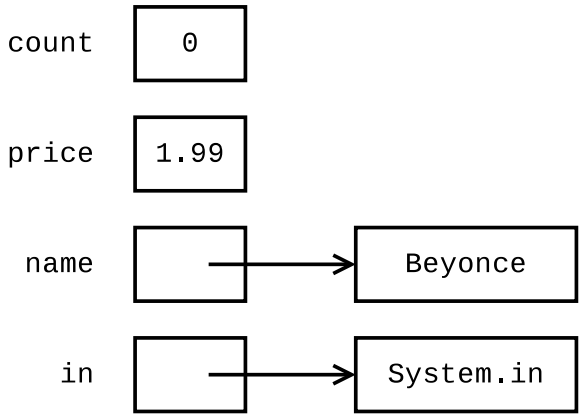
```
total = sum;
flag = minutes;
```

Note that assigning `'0'` to `days` is legal, but the value is actually stored as 48L (Unicode for the digit zero character).

Model 2 Reference Types

```
int count;
double price;
String name;
Scanner in;

count = 0;
price = 1.99;
name = "Beyonce";
in = new Scanner(System.in);
```



Java has eight primitive types (see Model 1). All other types of data are called ***reference*** types, because **their value is a memory address**. When drawing memory diagrams, use an arrow to reference other memory locations (rather than make up integer values for the actual addresses).

Questions (20 min) Start time:

9. What are the names of the reference types in the example above?

String and Scanner

10. Notice the pattern Java uses for type names like `int` and `String`:

a) Are reference type names all lowercase or capitalized? Capitalized

b) Are primitive type names all lowercase or capitalized? All lowercase

11. Variables in Java can use at most eight bytes of memory. Explain why the values `"Beyonce"` and `System.in` cannot be stored directly in the memory locations for `name` and `in`.

Both values are much larger than eight bytes, so they need to be stored somewhere else.

12. What is the value of the variable `count`? What is the value of the variable `price`?

The values are 0 and 1.99. They are stored directly in the variable's memory.

13. What is the value of the variable `name`? What is the value of the variable `in`?

The values are memory addresses. They reference the location where the actual data is stored.

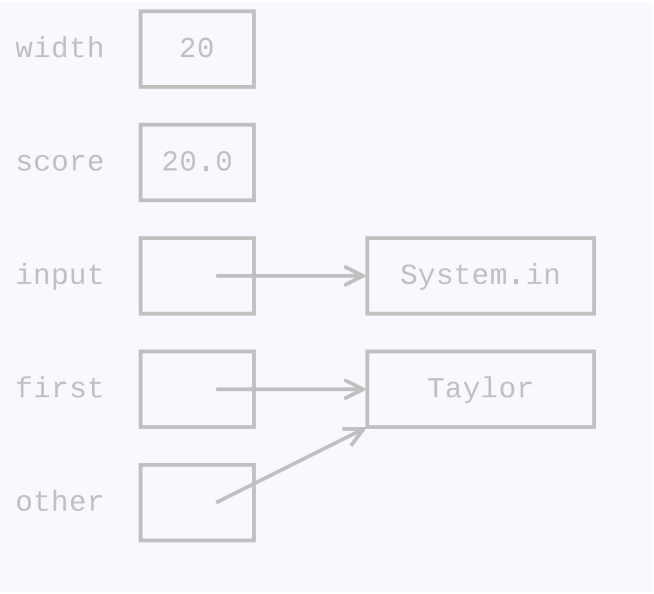
14. Carefully explain what it means to assign one variable to another. For example, what does the statement `price = count;` do in terms of memory?

Assignment simply copies the value of one variable to another. In the case of reference types, it only copies the memory location.

15. Draw a memory diagram for the following code. Make sure your answer is consistent with what you wrote for #14.

```
int width;
double score;
Scanner input;
String first;
String other;

width = 20;
score = 0.94;
input = new Scanner(System.in);
first = "Taylor";
score = width;
other = first;
```



16. What is the output of the following statements after running the code above? Explain your answer using the diagram.

```
first = "Swift";
System.out.println(other);
```

The output is `Taylor`, because changing the value (i.e., reference) of `first` does not affect the value of `other`.

17. (Optional) Paste the contents of *TaylorSwift.java* into Java Visualizer. What differences do you notice between the diagram in Java Visualizer and yours from #15?

Answers might include:

- The variables are drawn in (method) frames.
- There are no boxes drawn around the objects.