Recap: Assignment Statements

Here's an occasional point of confusion:

```
a = 7;  // what's in a?
b = a;  // what's in b?  what's in a now??
System.out.println("a is " + a + " b is " + b);
```

```
a = 8;
System.out.println("a is " + a + " b is " + b);
```

Recap: Expressions

- expression is combination of:
  - one or more operators and operands

- operator examples: +, -, /, ...

- operand examples: numbers, variables, ...

- precedence: multiply/divide higher than add/subtract

```
a = 7;  // what's in a?
b = a;  // what's in b?  what's in a now??
System.out.println("a is " + a + " b is " + b);
```

Recap: Converting Between Types

- Doubles can simply be assigned ints
- ints are subset of doubles
- Casting: convert from one type to another with information loss
- Converting from real to integer
  - int shows = (int) 1.5;
  - Truncation: fractional part thrown away
  - Show = (int) 1.75;
- Rounding: must be done explicitly
  - Show = Math.round(1.89);

Recap: Avoiding Magic Numbers

- magic numbers: numeric constants directly in code
- almost always bad idea!
- hard to understand code
- hard to make changes
- typos possible
- use constants instead
Programming with Classes

What if data we want to work with is more complex than these few primitive data types?

We can make our own data type: create a class

This specifies nature of data we want to work with

Operations defined within a class called methods

Programming with Objects

Object: specific instance of a class

Classes are templates for objects

Programmers define classes

Objects created from classes

Object Example

```java
public class StringTest
{
    public static void main (String[] args)
    {
        String firstname;
        String lastname;
        firstname = new String ("Kermit");
        lastname = new String ("theFrog");
        System.out.println("I am not " + firstname + " " + lastname);
    }
}
```

Constructors

Constructor: method with same name as class

Always used with new

Actually creates object

Typically initializes with data

```
fname = new String ("Kermit");
```
Object Example

```java
public class StringTest{
    public static void main (String[] args) {
        String firstname = new String("Alphonse");
        System.out.println("I am not " + firstname + " " + lastname);
    }
}
```

Objects vs. Primitives

- **References**
  - Frog object
  - String object
  - String vs. direct storage

- **Primitives**
  - int favoriteNum
  - FamousFrog Frog object

More String Methods

- `replace(char oldChar, char newChar)`: Returns a new String object where all instances of oldChar have been changed into newChar.
- `substring(int beginIndex, int endIndex)`: Returns new String object starting from beginIndex position and ending at endIndex position.

API Documentation

Online Java library documentation at [http://java.sun.com/javase/6/docs/api/](http://java.sun.com/javase/6/docs/api/)

Methods and Parameters

- Class definition says what kinds of data and methods make up object
  - object is specific instance of class
  - methods are how objects are manipulated

Parameters

- Methods can have multiple parameters
  - API specifies how many, and what type
  - Example: `replace(char oldChar, char newChar)`: String object is method which character in the String object we're interested in changing.

Explicit vs. Implicit Parameters

- Explicit parameters given between parentheses
  - Implicit parameter is object itself
  - Example: `substring(int beginIndex, int endIndex)`: String object is method which character in the String object we're interested in changing.

Class Libraries

- Before making new class yourself, check to see if someone else did it already
  - Libraries are written by other programmers
  - Java has single-character primitive data type
  - What if want to work with sequence of characters

Questions?

- Can consolidate declaration, assignment
  - just like with primitive data types

Methods and Parameters

- Class definition says what kinds of data and methods make up object
  - object is specific instance of class
  - methods are how objects are manipulated
Return Values

- Methods can have return values
- Example: `charAt` method result
  - return value, the character 'n', is stored in `thirdchar`

```java
String firstname = "kangaroo";
char thirdchar = firstname.charAt(2);
```

- Not all methods have return values
- Example: `println` method does not return anything
  - prints character 'n' on the monitor, but does not return that value
  - printing value and returning it are not the same thing!

```java
System.out.println(thirdchar);
```

- Again, API docs tell you
  - how many explicit parameters
  - whether method has return value
  - what return value is, if so

- No return value indicated as `void`

Constructors and Parameters

- Many classes have more than one constructor, taking different parameters
- Use API docs to pick which one to use based on what initial data you have

```java
animal = new String();
animal = new String("kangaroo");
```

Accessors and Mutators

- Method that only retrieves data is accessor
  - read-only access to the value
  - example: `charAt` method of String class
- Method that changes data values internally is mutator
  - Stay tuned for examples of mutators, we haven't seen any yet
  - String class has no mutator methods
- Accessor often called getters
- Mutators often called setters
  - names often begin with get and set, as in `getWhatever` and `setWhatever`