1. Review
   1. Math
      1. Static methods, like in the Math class, can only be called on the Math class
   2. Scanner
      1. For the scanner class there has to be an import to be able to reference the class
      2. Purpose: user input
      3. Reader.nextInt() etc.
   3. String
      1. Can be instantiated as a primitive type
      2. charAt(), length(), indexOf(), etc.
      3. strings can be manipulated
2. Using Objects (continued…)
   1. Assignment and Aliasing
      1. Alias: nickname, pseudonym
         1. Multiple references can be assigned to the same objects
         2. One object can have multiple references
         3. Assign null to a reference then the variable now refers to nothing

Ex:

Dog d1 = new Dog(“Chester”);

Dog d2 = new Dog(“Chester”);

Dog d3 = d2;

d3.bark(); //refers to dog d2

d2.bark(); // refers to dog d3

d2 = null; //removes the link between d2 and the actual dog

d3 = d1;

/\*Removes link between d3 and add a link to the first dog, deletes the second dog forever because you can no longer reference it

“Garbage collection” is when the computer deletes the second dog because there are no more references\*/

d3=d2;

d3=d1;

d2=null;

d2 = new Dog(“Bradley”); //d3 is still equal to null

d1=null; //removes link to first dog

d3=null; //removes link to first dog, this kills the first dog object because there is no more to reference it

1. Assignment with Primitives

int x=5;

int y=7;

int z=x;

x=3;

sout(x+””+y+””+z);

Output: 375

Ex:  
Fake Number Class

Methods

-setNumber(int n); //assign “the number” to n

-getNumber(); //returns the number

Number x = new Number(5);

Number y = new number(7);

y=x; //Number y is garbage collected

y= new Number(8); //new number

y=x; //y number is gone again

y.setNumber(9); //x and y are equal to 9

sout(x.getNumber()+””+y.getNumber());

OUTPUT: 99

If we reassign a reference to another object there is no effect to the alias of an object.

If we call a method using one reference to an object then all references to that object would see that change.

Number x = new Number(3);

x.setNumber(5);

Number y = new Number(4);

y=x;

x.setNumber(3);

sout(x.getNumber() + “” + y.getNumber());

OUTPUT: 33

1. Strings
   1. Strings are immutable: unable to change

String x = new String(“Serena”); //new String object named Serena

x= “Trey”; //This deletes the old String and creates a new one that stores “Trey” instead of “Serena”

x= “tre”; /Deletes the old objects and creates a new tre

String y=x; //new reference to the String object x

x= “Jack”; //changes the x reference to the String object

sout(x.toUppercase()); //this prints “JACK” but the object is still “Jack”, the uppercase stuff just goes away because it is an unneeded copy

* 1. Strings can be shared (objects can be shared too)
     1. a new string is made when a method returns something