Objects and Classes

• recall **encapsulation**: user of class does not need to know how class is implemented. Details of implementation are *encapsulated*, also known as **class hiding** (similar to *data hiding* with use of *private* instance variables)

• class also known as an **abstract data type (ADT)** since details are hidden from user

• **association**: a class can use another class

• **Wrapper classes**: objects associated with primitive variables, so can use classes that expect objects

• why have primitive variables at all? better performance than objects

• `int` ➔ `Integer`; `double` ➔ `Double`; `char` ➔ `Character`; `boolean` ➔ `Boolean`; etc.

• wrapper classes have methods that convert (sometimes automatically) between primitive and objects
  automatic conversion called **boxing** and **unboxing**

• wrapper classes are immutable - no methods that change the value, makes new one

• new `Integer(5)` makes object containing primitive variable
Objects and Classes

• **String** class is actually immutable. Contents do not change — new string is created instead of modifying.

• **String** and array of characters are not same thing. Can convert from one to another using `char [] c = String.toCharArray(string)` or `String s = new String(charArray);`

• To convert String to Numeric Value, e.g. `Double.parseDouble(string)`

• Can make formatted strings the same way `System.out.printf()` formats output

• **Stringbuilder** class like Strings but objects can be modified

• **Stringbuffer** like **Stringbuilder** but for parallel execution (involves some overhead)
Stack Class

• stack is a LIFO (last in, first out) data structure. We’ve already seen this with stack of activation records

• only allowed access to top item on stack

• putting a new item on stack is called push

• taking the top item off the stack called pop

• stack can be used for: reversing a string or a number, undoing editor commands, parentheses matching, implement recursion, …

• we will define a stack class and implement it using an array. (other data structures are possible you will see them in Data Structures 102, e.g. linked list)