Review: Primitive types

- Primitive types store a single value
- A primitive type does not have any defined functionality; it stores a value, but it can’t do anything by itself
- If we want to manipulate a primitive value, we have to apply operators to it or pass it into method calls, e.g.
  ```java
  age = age + 1;
  grade = (int) Math.round(points / total);
  ```

Objects

- An *object* is an entity that we use to represent more complicated concepts
- An object is different from a primitive value in two primary ways:
  - It can contain many pieces of data of many different types
  - It can have its own *methods* that define how we can interact with and manipulate it

Example: String

- We’ve already seen that Strings can hold a sequence of characters
- Strings also have a number of methods that we can use to manipulate the String and to obtain information about it

Strings: length

- We can find the length of a String by using its length method:
  ```java
  String hello = “Hello, world!”;
  int len = hello.length(); //len = 13
  ```
- Note that the method is called on the String variable hello
- We are asking the String to tell us its length

Methods

- Methods define behaviors for an object
- Methods have:
  - An identifier, i.e. its name
  - A return type
  - One or more parameters that provide information that the method uses in performing its computation
- Methods can access the internal data of the object and perform computations with it or modify it
- You don’t actually need to know the details of the way the method is implemented in order to use it!
Methods (cont.)

- Object methods can be used in mathematical expressions:

  ```java
  String word1 = "hello";
  String word2 = "world";
  int totalLength = hello.length() + world.length();
  ```

Object Methods

- We've already seen other object methods:

  ```java
  Scanner reader = new Scanner(System.in);
  int x = scanner.nextInt();
  // Tells the Scanner object to wait for an int to be entered in the console
  ```

- Some methods do not return anything; their return type is `void`:

  ```java
  System.out.println("hi!");
  // Tells System.out, an object of type PrintStream, to print hi to the console.
  ```

Some String Methods

- `length()`
  - Returns the length of the String
  - No parameters
  - Return type: `int`

  ```java
  String alpha = "abcdefghijklmnopqrstuvwxyz";
  int numLetters = alpha.length();
  System.out.println(numLetters); //prints 26
  ```

- `charAt(pos)`
  - Returns the character at the given position in the String
  - Position number begin at ZERO, not one!
  - `pos` is an `int`
  - Return type: `char`

  ```java
  String alpha = "abcdefghijklmnopqrstuvwxyz";
  System.out.println(alpha.charAt(2)); //prints c
  System.out.println(alpha.charAt(25)); //prints z
  ```

- `replace(oldChar, newChar)`
  - Creates a new String in which all instances of oldChar have been replaced by newChar
  - oldChar and newChar have type `char`
  - Return type: `String`

  ```java
  String hello = "Hello World!";
  System.out.println(hello.replace('l','!');
  //prints He!!o Wor!d!
  ```

- Note: in Java, character literals are denoted by single quotes `'`, not double quotes `"`

Some String Methods

- `substring(start, pastEnd)`
  - Creates a new String that contains part of the String
  - `start` and `pastEnd` are integers
  - Return type: `String`

  ```java
  String alpha = "abcdefghijklmnopqrstuvwxyz";
  String sub = alpha.substring(3,6);
  //sub is "def"
  ```

- `note:` character literals are denoted by single quotes `'`, not double quotes `"`
Some String Methods

- `toLowerCase()`
  - Returns a new String in which all upper case letters are converted to lower case (numbers and symbols are ignored)
  - No parameters
  - Return type: String

```
String hello = "HELLO CompSci 6!";
System.out.println(hello.toLowerCase()); //prints hello compsci 6!
```

Questions:

- Suppose we call:
  ```
  String hello = "Hello";
  String sub = hello.substring(3,3);
  ```
  - What does sub contain?
- We've seen that scanners have methods for reading ints, doubles, and Strings, but they do not have a method for reading in a single char.
  - How can we do this?