Instantiation vs. Initialization

- Objects (aka Classes) are instantiated, variables are initialized
  Example of instantiation:
  ```
  new Guitarist();
  ```
  "We have instantiated Guitarist"
  Some examples of initialization:
  ```
  Guitarist guitarist = new Guitarist();
  _guitarist = new Guitarist();
  int num = 3;
  ```
  "We have initialized the instance variable '_guitarist' and the local variables 'guitarist' and 'num'"
- Initializing a variable means assigning it some initial value so that it is not null.
- Declaring a variable is not the same thing as initializing it
  - `private Guitarist _guitarist;` _guitarist is null, or has no value
  - `Guitarist guitarist;` again, guitarist is null.
- This is initializing a variable:
  - `int num = 5;` here, `num` is given an initial value of 5.
  - Remember that "`=`" means assignment or "gets," not "is equal to"
- `_guitarist = new Guitarist();` this is instantiating an object. We create a new instance of a `Guitarist` and initialize `_guitarist` to be the value of the new `Guitarist`

Constructors

- Think of a class's Constructor as a "blueprint" for building an instance of that class/type. Just like declaring an instance variable, declaring the Constructor in your class file does not make an object. You must always call `new <Type>();` in another class in order for an actual instance of that class to be created. This is the only way to create an instance of, or to *instantiate*, an object.
- Just like a regular method, all the code in the Constructor is executed when you call `new <Type>();`
- `super()` is a special example, and rare exception, of calling a Constructor. It does not require the `new` keyword because we are not actually instantiating the superclass. However, just like any Constructor, `super()` might need to take parameters, so when you call `super()`, you have to make sure you pass in the object(s) that the superclass's Constructor might be expecting. Also remember that `super()` is always the very first thing called in the Constructor of the subclass.
- A class's Constructor is almost always the place where you will initialize your instance variables. Instance variables can be *initialized* to parameters passed into the Constructor, to newly instantiated objects, or to literal/constant values like integers. You also might want to give initial properties to certain objects using accessors and mutators in the Constructor as well.

→ Cool, so now I know how to instantiate objects using Constructors, but what do I do with them after they're created?

Glad you asked! When you instantiate an object, you have three options. You can…

1. Store it in an instance variable

```
package RockBand;

public class RockBand { // This is RockBand's class declaration

        // Declare your instance variable of type Guitarist. At this point _guitarist is null.
        private Guitarist _guitarist;

        public RockBand() { // This is the RockBand's Constructor
                // Instantiate a Guitarist and initialize your _guitarist instance variable with it
                _guitarist = new Guitarist();
        }

        public void party() {
                _guitarist.drinkChampagne();
                /* Other band members elided */
        }

        public void soundCheck() {
                _guitarist.tuneGuitar();
                _guitarist.playGuitar();
        }

        // Other methods elided
}
```

Why would I want to do that?

Instance variables are super useful because they can be used anywhere in the class they were instantiated in. That means that `_guitarist` can be used in any method through the RockBand.java file as long as it has been initialized with an instance of type `Guitarist`. As you can see, `RockBand`'s instance of `Guitarist` is used in both the `soundCheck` and `playSong` methods.

Note: Instance variables should always start with an underscore followed by a lowercase letter (examples: `_variable`, `_myVariable`, `_thisIsMyVariable`).

2. Store it in a local variable

```
package RockBand;

public class RockBand { // This is RockBand's class declaration

    // Declare your instance variable of type Guitarist. At this point _guitarist is null.
    private Guitarist _guitarist;

    public RockBand() { // This is the RockBand's Constructor
        // Instantiate a Guitarist and initialize your _guitarist instance variable with it
        _guitarist = new Guitarist();
    }

    // Other methods elided

    public void party() {
        // Declare a local variable of type Champange on the left and initialize it with a new
        instance of Champange on the right.
        Champagne champagne = new Champagne();
        _guitarist.spray(champagne);
        _guitarist.drink(champagne);
        _guitarist.spill(champagne);
    }


}
```

Why would I want to do that?
Local variables are very efficient. If you are only going to use a variable within one method, it is best to use a local variable. This will keep your code concise and easier to follow. In this case, the `RockBand` only needs `Champagne` when they `party`, so there is no need to make it an instance variable.

Note: Local variables should always start with a lowercase letter, not an underscore (examples: `variable`, `myVariable`, `thisIsMyVariable`).

3. Don't store it at all

```
public class RockBand { // This is RockBand's class declaration

     // Declare your instance variable of type Guitarist. At this point _guitarist is null.
     private Guitarist _guitarist;

     public RockBand() { // This is the RockBand's Constructor
          // Instantiate a Guitarist and initialize your _guitarist instance variable with it
          _guitarist = new Guitarist();
     }

     // Other methods elided
     public void party() {
          Champagne champagne = new Champagne();
          _guitarist.spray(champagne);
          _guitarist.drink(champagne);
          _guitarist.spill(champagne);
          // Instantiate an instance of Beyonce directly in hangWithCeleb's parentheses.
          _guitarist.hangWithCelebs(new Beyoncé());
     }
}
```

Why would I want to do that?
Sometimes we want an instance of an object but don't ever need to reference it within that class or method again. Here we tell `_guitarist` to hang out with celebs and pass in a new instance of Beyoncé. We are never going to use Beyoncé ever again in this class (because the `_guitarist` can't handle being in Beyoncé's presence more than once obviously), so we don't need use a variable to store our new instance of Beyoncé.

Note: We can also instantiate objects outside of parameters and never do anything with them at all. A classic example of this in your App.java file when you instantiate your top-level class.
Examples:

- `new LiteBrite();`
- `new Guitarist(new Guitar());`