Exercise Solution: Time Counter

I hope you figured out how to refactor the Counter and update changes.

Take a look at one way of doing this:

```
export default function(state, action) {
  switch (action.type) {
    case "SET_ACTIVE_SESSION":
      return {
        ...state,
        activeSession: action.payload
      };
    case "INCREASE_COUNTER":
      //retrieve the activeSession from the action payload
      //however this comes in all caps so convert to lower Case
      const activeSession = action.payload.toLowerCase();
      return {
        ...state,
        [activeSession]: state[activeSession] + 1
      };
    case "DECREASE_COUNTER":
      //retrieve the activeSession (called session here) from the action payload
      const session = action.payload.toLowerCase();
      return {
        ...state,
        [session]: Math.max(0, state[session] - 1)
      };
    default:
      return state;
  }
}
```

Once again, there isn't much complexity in the solution.

We have a three types of actions being handled in the **reducers/index.js**.

INCREASE_COUNTER is used to increment the counter.

```
[activeSession]: state[activeSession] + 1
```

DECREASE_COUNTER is used to decrement the counter.

```
[session]: Math.max(0, state[session] - 1)
```

Math.max() simply takes the larger value between zero and **state[session] - 1** in order to avoid negative values in the counter.

The initial state is set in **store.js** as always.

The **handleCounter** is an action creator being used in **App.js** for dispatching our desired action once the button is clicked.

That's it for this section of the course. Next up, we'll be moving on to a more complex Redux app. See you!