Verbs and Keywords

Monadic functions should be writen in theverb/noun pair. For example :***write(name)***

assertEquals(expected, actual)
vs
assertExpectedEqualsActual(expected, actual)

Have No Side Effects

Side Effects are lies.

Your functions should not do hidden things like:
make unexpected changes to the variables of its own class,
change the he parameters passed into the function or change or change the system globals

Have No Side Effects

Consider the following code:

https://github.com/prestrepoh/Clean-Code-Course/blob/master/2-functions/2-have-no-side-effects/hiddensideeffect.java

It simply matches a user to its password, but it also has a hidden side effect.

Have No Side Effects

The name does not imply that it initializes the session!.

CheckPassword can only be called at certain times.

If it is
called out of order, session data may be inadvertently lost

Command Query Separation

Functions should either do something or answer something, but not both.

Consider the function:
public boolean set(String attribute, String value); // set the attribute and returns true if succesful.

The author intended set to be a verb, but in the context of the if statement it feels like
an adjective.

Command Query Separation

It would be much better if it was:

```
if (attributeExists("username")) {
setAttribute("username", "unclebob");
...
}
```

Prefer Exceptions to Returning Error Codes

It would be much better if it was:

```
if (attributeExists("username")) {
setAttribute("username", "unclebob");
...
}
```

Prefer Exceptions

Returning error codes from command functions promotes commands being used as expressions in the predicates of if statements:

```
if (deletePage(page) == E_OK)
```

Prefer Exceptions

The Error.java Dependency Magnet:

```
public enum Error {
OK,
INVALID,
NO_SUCH,
LOCKED,
OUT_OF_RESOURCES,
WAITING_FOR_EVENT;
}
```

Prefer Exceptions

https://github.com/prestrepoh/Clean-Code-Course/blob/master/2-functions/3-prefer-exceptions-to-returning-error-codes/nestederrorhandling.java

vs

https://github.com/prestrepoh/Clean-Code-Course/blob/master/2-functions/3-prefer-exceptions-to-returning-error-codes/notnestederrorhandling.java

Extract Try/Catch Blocks

Try/catch blocks are ugly in their own right.

it is better to extract the bodies of the try and catch blocks out into functions of their own.

https://github.com/prestrepoh/Clean-Code-Course/blob/master/2-functions/4-extract-try-catch-blocks/trycatchblockextracted.java

Error Handling Is One Thing

Error handing is one thing.

A function that handles
errors should do nothing else.

if the keyword
try exists in a function, it should be the
very first word in the function and that
there
should be nothing after the catch/finally
blocks

Don't Repeat Yourself

Object Oriented programming is in essence a strategy to avoid redundancy.

There was hidden redundancy in our example code.

How do you write functions like this?

Nobody writes functions this way to start.

You should write your functions like a letter. Frist write the draft, and then massage it until it's clean.

Conclusion

Master programmers think of systems as stories to be told rather than programs to be written.

your real goal is to tell the story of the system.

Final version of our cleaned up example:

https://github.com/prestrepoh/Clean-Code-Course/blob/master/2-functions/longfunctionfinalversioncomplete.java