Clean Code: Meaningful Names

01219245/01219246
Individual Software Process

Code and intention

- Consider these fragments of codes

```
a = c( s );
e = d( a );
a += e;
```

```
a = c( s ); // compute price for products s
e = d( a ); // compute tax
a += e;     // add tax
```

```
price = computePrice( products );
tax = computeTax( price );
price += tax;
```

What do you see from code?

A piece of text

Interactions of variables

Intention

Use intention-revealing names

```
var d = 10;  // elapsed time in days
```

```
var elapsedTimeInDays = 10;
var daysSinceCreation = 10;
var fileAgeInDays = 10;
```

- A good name goes a long way.
- Using **d** as a variable name does not mean anything. The name does not say what you mean.

Example (from the CC book)

```
public List<int []> getThem() {
  List<int []> list1 = new ArrayList<int []>();
  for(int[] x: theList)
    if(x[0] == 4)
      list1.add(x);
  return list1;
}
```

These are valuable information that could have been presented in the code.

- Do you understand the code?
- Is there any question do you want to ask?

First rewrite

```
public List<int []> getFlaggedCells() {
  List<int []> flaggedCells = new ArrayList<int []>();
  for(int[] cell: gameBoard)
    if(cell[STATUS_VALUE] == FLAGGED)
      flaggedCells.add(cell);
  return flaggedCells;
}
```

- Is it better?
- We might encapsulate each cell as an object of class `Cell` and further improve the code.

Second rewrite

```
public List<Cell> getFlaggedCells() {
  List<Cell> flaggedCells = new ArrayList<Cell>();
  for(Cell cell: gameBoard)
    if(cell.isFlagged())
      flaggedCells.add(cell);
  return flaggedCells;
}
```

- Is it better?

Avoid disinformation

- Having false clues are worst than having no clues.

```
var shipList = new Ship();
```

```
ship.moveItToTheRightBySmallAmountAndStop();
ship.moveItToTheRightBySmallAmountAndJump();
```

Same method!

Make meaningful distinctions (1)

```
var copy = function(a1, a2, n) {
    for ( var i = 0; i < n; i++ ) {
        a2[i] = a1[i];
    }
}
```

```
var copy = function(source, destination, n) {
    for ( var i = 0; i < n; i++ ) {
        destination[i] = source[i];
    }
}
```

Example of ...

```
copy( students, temp, 100 );
move( oldStudents, temp, 50 );
```

```
var copy = function(src, dest, n) { .. }
var move = function(dest, src, n) { .. }
```

Make meaningful distinctions (2)

```
var info1 = getAccount();
var info2 = getAccountInfo();
var info3 = getAccountData();
```

Quick rules

- Make similar things look similar.
- Make different things look different.
- ***Don't fool yourself.***

Communication

- Good names communicates better
  - Use pronounceable names

```
ship.vxyaxy = ...
```

  - Use searchable names

```
var G = -1;
```

```
var Ship.G = -1;
```

  - Avoid encodings

```
var nameString = ”fdfdfdf”;
var gradeFloat = 3.2;
```

Things to distinguish in code

- Class names, variable names, function names
- Object, objects
- ***Use good names to help you.***

Class names

- Classes are concepts for **things**.
- Class names should be nouns or noun phrases:
  - Ship
  - GameLayer
  - ShipMovementController
- Don't use verbs as class names.

Method names

- Method names should be verbs or verb phrases
  - runAction
  - addChild
  - save

Other guidelines

- Avoid mental mappings
- Don't be cute
- Pick one word per concept
- Don't pun
- Use solution domain names, use problem domain names
- Contexts