# Naming. Say what you mean, mean what you say.

## 1. Use Intention-Revealing names.

If a name requires a comment, then that name does not reveal its intent:

```
int d; // elapsed time in days
```

In this case, we should choose a name that specifies what is being measured and the unit of that measurement:

```
int elapsedTimeInDays;
int daysSinceCreation;
int daysSinceModification;
int fileAgeInDays;
```

Choosing names that reveal intent can make it much easier to understand and change code. For example, what is the purpose of this code?

```
// bad naming example
private double methodA(int a) {
	return methodB(a) * 3.14d;
}
```

### Can you answer these questions about the previous code?

1. What is the purpose of the code?
2. What is the significance of variable *a*?
3. What is the significance of value 3.14d?

### Proper naming solves the problem:

```
// proper named code tells what it does
private int circleArea(int radius){
	return square(radius) * PI;
}
```

## 2. Use pronounceable names

Compare:

```
class DtaRcrd102 {
	private Date genymdhms;
	private Date modymdhms;
	private final String pszqint = "102";
	/* ... */
}
```

To:

```
class Customer {
	private Date generationTimestamp;
	private Date modificationTimestamp;
	private final String recordId = "102";
	/* ... */
}
```

## Class Names

Classes and objects should have noun or noun phrase names like *Customer , WikiPage, Account, and AddressParser.* Avoid words like Manager, Processor, Data, or Info in the name of a class. **A class name should not be a verb.**

## Method Names

Methods should have verb or verb phrase names like *postPayment, deletePage, or save.* Accessors, mutators, and predicates should be named for their value and prefixed with *get, set, and is,* according to the javabean standard:

```
String name = employee.getName();
customer.setName("mike");
if (paycheck.isPosted())...
```

## Interfaces and Implementations

**If possible *do not* name interface classes with the *"I"* prefix.**

For example, say you are building an *AbstractFactory* for the creation of shapes (This example will make more sense after the design patterns lecture). This factory will be an interface and will be implemented by a concrete class. How should you name them? *IShapeFactory* or *ShapeFactory*? Prefer to leave interfaces unadorned. The preceding *"I"*, so common into today's code, is a distraction at best and too much information at worst. We don't want users knowing that we are handing them an interface. We just want them to know that it's a *ShapeFactory*. So if you must encode either the interface or the implementation, choose the implementation.