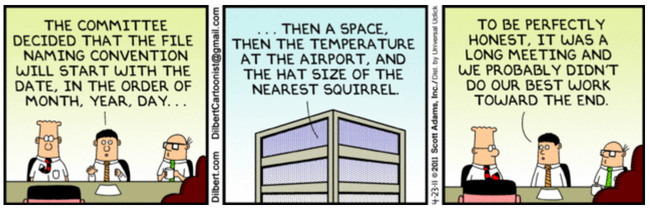

THE COMMITTEE DECIDED THAT THE FILE NAMING CONVENTION WILL START WITH THE DATE, IN THE ORDER OF MONTH, YEAR, DAY...
... THEN A SPACE, THEN THE TEMPERATURE AT THE AIRPORT, AND THE HAT SIZE OF THE NEAREST SQUIRREL.
TO BE PERFECTLY HONEST, IT WAS A LONG MEETING AND WE PROBABLY DIDN'T DO OUR BEST WORK TOWARD THE END.
Dilbert.com DilbertCartoonist@gmail.com
4-23-11 © 2011 Scott Adams, Inc./Dist. by Universal Uclick

Meaningful Names

www.cs.uoi.gr/~zarras/soft-devII.htm

from Clean Code by R. C. Martin, a.k.a "Uncle Bob"

Use Intention Revealing Names

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

Use Intention Revealing Names

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

If a name requires a comment, then the name does not reveal its intent.

Use Intention Revealing Names

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

The name d reveals nothing. It does not evoke a sense of elapsed time, nor of days.

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;
```

Use Intention Revealing Names

The name of a variable, function, or class, should answer all the big questions.

It should tell you

- **why it exists,**
- **what it does,**
- **and how it is used.**

Use Intention Revealing Names

What is the purpose of this code?

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

Use Intention Revealing Names

What is the purpose of this code?

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

Why is it hard to tell what this code is doing?

The code implicitly requires that we know the answers to questions such as:
- What kinds of things are in theList?
- What is the significance of the zero-th subscript of an item in theList?
- What is the significance of the value 4?
- How would we use the list being returned?

Use Intention Revealing Names

Say that we're working in a mine sweeper game. We find that the board is a list of cells called theList.

Each cell on the board is represented by a simple array.

We further find that the zero-th subscript is the location of a status value

and that a status value of 4 means "flagged."

Just by giving these concepts names we can improve the code considerably:

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

Use Intention Revealing Names

Once again compare

```
for (int j=0; j<34; j++) {
  s += (t[j]*4)/5;
}
```

Use Intention Revealing Names

Once again compare

```
for (int j=0; j<34; j++) {
  s += (t[j]*4)/5;
}
```

to

```
int realDaysPerIdealDay = 4;
const int WORK_DAYS_PER_WEEK = 5;
int sum = 0;
for (int j=0; j < NUMBER_OF_TASKS; j++) {
  int realTaskDays = taskEstimate[j] * realDaysPerIdealDay;
  int realTaskWeeks = (realTaskDays / WORK_DAYS_PER_WEEK);
  sum += realTaskWeeks;
}
```

Avoid Disinformation

What is accountList ??

```
if(accountList == null){
   throw new BankException();
}
```

Avoid Disinformation

What is accountList ??

```
Account searchForAccount(Account accountList[]){
….. . . . .
  if(accountList == null){
     throw new BankException();
  }
…………
}
```

The word list means something specific to programmers.

If the container holding the accounts is not actually a List, it may lead to false conclusions.

So accountGroup or bunchOfAccounts or just plain accounts would be better.

Avoid Disinformation

What does this code do ??

```
int a = l;
if (O == l)
  a = O1;
else
  l = 01;
```

Avoid Disinformation

What does this code do ??

```
int a = l;
if (O == l)
  a = O1;
else
  l = 01;
```

A truly awful example of disinformative names would be the use of lower-case L or uppercase O as variable names, especially in combination.

Such things do exist !!!.

In one case the author of the code suggested using a different font so that the differences are more clear !!!!

Make Meaningful Distinctions

what is the purpose of the parameters ???

```
public static void copyChars(char a1[], char a2[]) {
  ………
}
```

Make Meaningful Distinctions

what is the purpose of the parameters ???

```
public static void copyChars(char a1[], char a2[]) {
  for (int i = 0; i < a1.length; i++) {
    a2[i] = a1[i];
  }
}
```

Number-series naming (a1, a2, .. aN) is the opposite of intentional naming. Such names are not disinformative—they are non-informative; they provide no clue to the author's intention.

This function reads much better when source and destination are used for the parameter names.

Make Meaningful Distinctions

what is the purpose of the classes ???

```
class Product {…}

class ProductInfo {…}

class ProductData {…}
```

Make Meaningful Distinctions

what is the purpose of the classes ???

```
class Product {…}

class ProductInfo {…}

class ProductData {…}
```

Noise words are another meaningless distinction.

We have made the names different without making them mean anything different !!!!

→ Distinguish names in such a way that the reader knows what the differences offer.

Use Pronounceable Names

Try to read this !!

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

Use Pronounceable Names

Compare these two ….

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

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

If you can't pronounce it, you can't discuss it !!!!

This matters because programming is a social activity !!!!

Use Searchable Names

Imagine that you search for a variable, named **e** in a file that contains the following code

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

Use Searchable Names

Imagine that you search for a variable, named **e** in a file that contains the following code

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

you would get > 10 matches …
the name e is a poor choice for any variable for which a programmer might need to search.
It is the most common letter in the English language and likely to show up in every passage of text in every program.

Avoid Encodings

What do these variable names mean ???

```
bBusy
chInitial
dLightYears
iSize
fpPrice
dbPi
piFoo
```

Avoid Encodings

What do these variable names mean ???

```
boolean bBusy;
char chInitial;
double dLightYears;
int iSize;
float fpPrice;
double dbPi;
int *piFoo;
```

In the old years we often used encodings like these; In Hungarian Notation, a variable name starts with a group of lower-case letters which are mnemonics for the type of that variable.

Today there is a trend toward **smaller classes** and **shorter functions** so that people can usually see the point of declaration of each variable they're using.

So encodings are unnessesary burden; plus encoded names are seldom pronounceable.

Avoid Encodings

Suppose we use encodings, what is the type of this variable ??

```
if(stringPhone == null)
  throw new NullPointerException();
```

Avoid Encodings

Suppose we use encodings, what is the type of this variable ??

```
PhoneNumber stringPhone;
// name not changed when type changed!

…….
if(stringPhone == null)
  throw new NullPointerException();
```

encoded names are easy to mis-type !!!!

Avoid Encodings

Yet another kind of encoding …

```
public class Part {
  private String m_dsc; // The textual description

  void setName(String name) {
    m_dsc = name;
  }
}
```

Avoid Encodings

Yet another kind of old-times encoding …

```
public class Part {
  private String m_dsc; // The textual description

  void setName(String name) {
    m_dsc = name;
  }
}
```

You also don't need to prefix member variables with m_ anymore.

Your classes and functions should be small enough that you don't need them.

And you should be using an editing environment that highlights or colorizes members to make them distinct.

Avoid Encodings

Yet another kind of encoding …

```
public class Part {
  private String m_dsc; // The textual description

  void setName(String name) {
    m_dsc = name;
  }
}
```

Besides, people quickly learn to ignore the prefix (or suffix) to see the meaningful part of the name.

much better…

The more we read the code, the less we see the prefixes.

```
public class Part {
  String description;
  void setDescription(String description) {
    this.description = description;
  }
}
```

Avoid Mental Mapping

Readers shouldn't have to mentally translate your names into other names they already know.

This problem generally arises from a choice to use neither problem domain terms nor solution domain terms.

This is a problem with single-letter variable names.

Certainly a loop counter may be named i or j or k (though never l!) if its scope is very small and no other names can conflict with it.

However, in most other contexts a single-letter name is a poor choice; it's just a place holder that the reader must mentally map to the actual concept.

Class Names

Classes are concepts of the problem domain.

Hence, Classes and objects should have noun or noun phrase names like:

```
Customer, WikiPage, Account, and AddressParser
```

A class name should not be a verb.

Method Names

Methods are actions performed by the objects. Hence, methods should have verb or verb phrase names like:

```
 postPayment, deletePage, save
```

Accessors, mutators, and predicates should be named for their value and prefixed with get, set, and is.

```
string name = employee.getName();
customer.setName("mike");
if (paycheck.isPosted()) { …. }
```

Method Names

When constructors are overloaded, sometimes it is useful to use static factory methods with names that describe the arguments.

```
Complex lowPoint = new Complex(23.0);

Complex highPoint = new Complex(12.5, 23.0);
```

Method Names

When constructors are overloaded, sometimes it is useful to use static factory methods with names that describe the arguments.

```
Complex lowPoint = Complex.createFromRealNumber(23.0);
```

vs.

```
Complex lowPoint = new Complex(23.0);


Complex highPoint = new Complex.createFromRealAndImaginary(12.5, 23.0);
```

vs.

```
Complex highPoint = new Complex(12.5, 23.0);
```

How can we enforce the use of the factory methods, instead of the overloaded constructors to the programmers that use our class ??

Method Names

When constructors are overloaded, sometimes it is useful to use static factory methods with names that describe the arguments.

```
Complex lowPoint = Complex.createFromRealNumber(23.0);
```

vs.

```
Complex lowPoint = new Complex(23.0);

Complex highPoint = new Complex.createFromRealAndImaginary(12.5, 23.0);
```

vs.

```
Complex highPoint = new Complex(12.5, 23.0);
```

How can we enforce the use of the factory methods, instead of the overloaded constructors to the programmers that use our class ??

We can just make the constructors private !

Don't be Cute

What is the purpose of these methods ?

```
interface CuteThread {
   whack();
   hitTheRoad();
   letItRoll();
}
```

Don't be Cute

What is the purpose of these methods ?

```
interface CuteThread {
   whack();
   hitTheRoad();
   letItRoll();
}
```

If you use clever or humorous names, they will be memorable only to people who share your sense of humor, and only as long as these people remember the joke.

```
interface ClearThread {
   kill();
   run();
   execute();
}
```

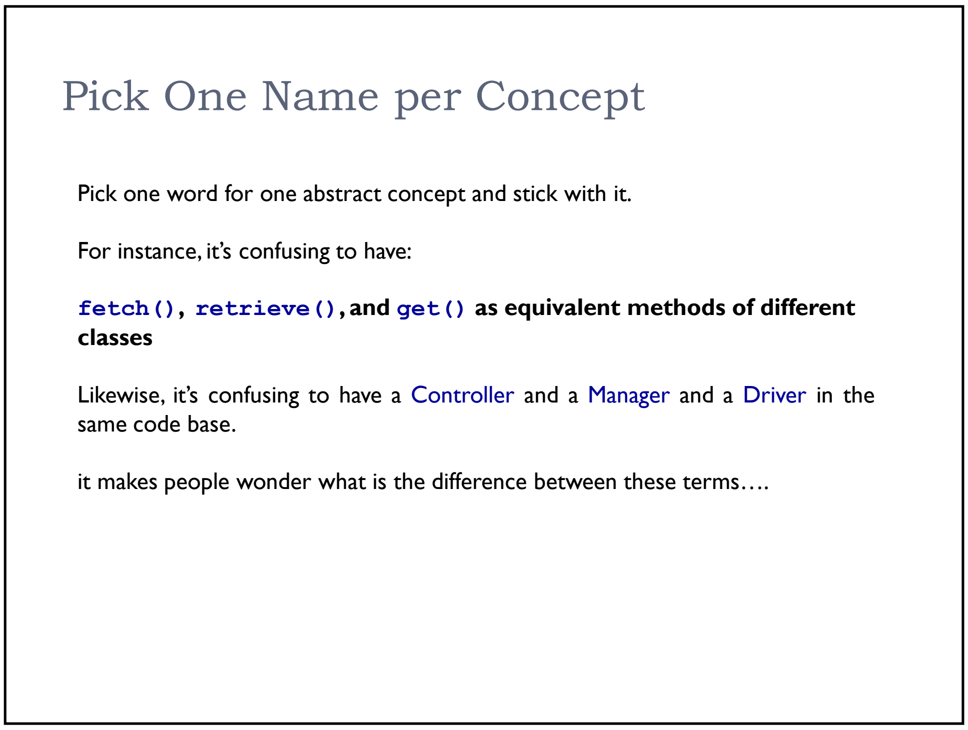
still there is a problem here … can you spot it ???

Don't be Cute

```
interface ClearThread {
   kill();
   run();
   execute();
}
```

still there is a problem here … can you spot it ???

there is no clear distinction between run and execute



Pick One Name per Concept

Pick one word for one abstract concept and stick with it.

For instance, it's confusing to have:

`fetch()`, `retrieve()`, and `get()` as equivalent methods of different classes

Likewise, it's confusing to have a Controller and a Manager and a Driver in the same code base.

it makes people wonder what is the difference between these terms….