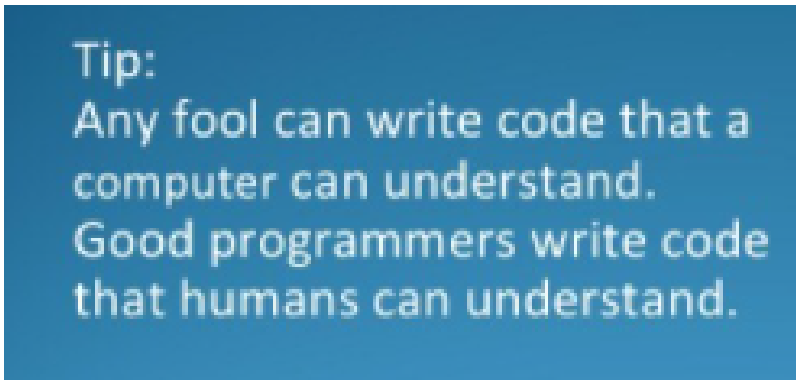
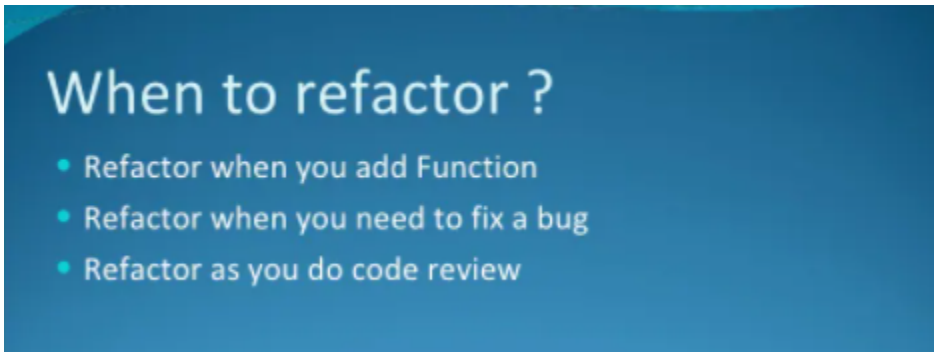
Refactoring:

Refactoring consists of improving the internal structure of an existing program's source code while preserving its external behavior.





When you should not refactor?

- There are times when the existing code is such a mess that although you could refactor it, it would be easier to start from the beginning.
- The other time you should avoid refactoring is when you are close to a deadline. At that point the productivity gain from refactoring would appear after the deadline and thus be too late.

Fowler's refactoring concepts and their applications:

Who is Martin Fowler ?

Martin Fowler is an author and international speaker on software development, specializing in object-oriented analysis and design, UML, patterns, and agile software development methodologies, including extreme programming.

Fowler is a member of the *Agile Alliance* and helped create the Manifesto for Agile Software Development in 2001, along with more than 15 co-authors. Martin Fowler was born in Walsall England, and lived in London a decade before moving to United States in 1994.

- M.F:Whenever I do refactoring, the first step is always the same. I need to build a solid set of tests for that section of code. The tests are essential because even though I follow refactorings structured to avoid most of the opportunities for introducing bugs, I'm still human and still make mistakes. Thus I need solid tests.

Tip:

Before you start refactoring, check that you have a solid suite of tests. These tests must be self-checking

Extract Method

Before refactoring

```
void printOwing(double amount)
{
    printBanner();

    //print details
  WriteLine("name:" + _name);
  WriteLine("amount" + amount);
}
```

After Refactoring

```
void printOwing(double amou
{
    printBanner();
    printDetails(amount);
}
void printDetails(double amou
{
 WriteLine("name:" + _name);
 WriteLine("amount" + amount
}
```

Inline Method

Before Refactoring

```
int getRating() {
return
   (moreThanFiveLateDeliveries()) ?
   2 : 1;
}
boolean
   moreThanFiveLateDeliveries() {
return _numberOfLateDeliveries > 5;
}
```

After Refactoring

```
int getRating() {
return (_numberOfLateDeliverie
   5) ? 2 : 1;
}
```