Clean Code

Clean code is simple and direct.

Clean code reads like well-written prose.

Grady Booch

Clean code always looks like it was written by someone who cares.

Michael Feathers

How to Clean Code?

- meaningful names
- reasonable abstractions
- appropriate formatting
- knowing how to use comments

Confusing names

Class Names

```
public class CtxSwitchGen104Impl {
   …
}
```

Variable Names

```
private int gd;
private int genymdhms;
```

Method Names

```
public void doStuff() {
   …
}
```

Meaningful names

Class Names

```
public class EmailSender {
   …
}
```

Variable Names

```
private int heatFactor;
```

Method Names

```
public void sendEmail() {
   …
}
```

Reasonable abstraction

```
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

Reasonable abstraction

🚫

```
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

```
public String printStatement() {
 return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
 return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
 return rentals.stream()
     .map(this::printLine)
     .collect(joining());
}

private String printFooterStatement() {
 return "Amount owed is " + calculateTotalPrice() + "\n" +
   "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
 return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
 return rentals.stream()
     .mapToDouble(Rental::getPrice)
     .sum();
}

private int calculateFrequentRenterPoints() {
 return rentals.stream()
     .mapToInt(Rental::calculateFrequentRenterPoints)
     .sum();
}
```

☺

Reasonable abstraction

```
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

```
public String printStatement() {
 return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
 return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
 return rentals.stream()
    .map(this::printLine)
    .collect(joining());
}

private String printFooterStatement() {
 return "Amount owed is " + calculateTotalPrice() + "\n" +
   "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
 return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
 return rentals.stream()
    .mapToDouble(Rental::getPrice)
    .sum();
}

private int calculateFrequentRenterPoints() {
 return rentals.stream()
    .mapToInt(Rental::calculateFrequentRenterPoints)
    .sum();
}
```

Reasonable abstraction

```java
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

```java
public String printStatement() {
 return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
 return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
 return rentals.stream()
    .map(this::printLine)
    .collect(joining());
}

private String printFooterStatement() {
 return "Amount owed is " + calculateTotalPrice() + "\n" +
   "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
 return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
 return rentals.stream()
    .mapToDouble(Rental::getPrice)
    .sum();
}

private int calculateFrequentRenterPoints() {
 return rentals.stream()
    .mapToInt(Rental::calculateFrequentRenterPoints)
    .sum();
}
```

Reasonable abstraction

```
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

```
public String printStatement() {
 return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
 return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
 return rentals.stream()
    .map(this::printLine)
    .collect(joining());
}

private String printFooterStatement() {
 return "Amount owed is " + calculateTotalPrice() + "\n" +
   "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
 return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
 return rentals.stream()
    .mapToDouble(Rental::getPrice)
    .sum();
}

private int calculateFrequentRenterPoints() {
 return rentals.stream()
    .mapToInt(Rental::calculateFrequentRenterPoints)
    .sum();
}
```

Reasonable abstraction

```java
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

```java
public String printStatement() {
 return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
 return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
 return rentals.stream()
    .map(this::printLine)
    .collect(joining());
}

private String printFooterStatement() {
 return "Amount owed is " + calculateTotalPrice() + "\n" +
   "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
 return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
 return rentals.stream()
    .mapToDouble(Rental::getPrice)
    .sum();
}

private int calculateFrequentRenterPoints() {
 return rentals.stream()
    .mapToInt(Rental::calculateFrequentRenterPoints)
    .sum();
}
```

Reasonable abstraction

```
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

```
public String printStatement() {
 return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
 return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
 return rentals.stream()
    .map(this::printLine)
    .collect(joining());
}

private String printFooterStatement() {
 return "Amount owed is " + calculateTotalPrice() + "\n" +
  "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
 return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
 return rentals.stream()
    .mapToDouble(Rental::getPrice)
    .sum();
}

private int calculateFrequentRenterPoints() {
 return rentals.stream()
    .mapToInt(Rental::calculateFrequentRenterPoints)
    .sum();
}
```

Reasonable abstraction

```java
public String printStatement() {
 double totalAmount = 0;
 int frequentRenterPoints = 0;
 final Iterator<Movie> rentals = this.rentals.keySet().iterator();
 String result = "Rental Record for " + getName() + "\n";
 while (rentals.hasNext()) {
  double thisAmount = 0;
  final Movie each = rentals.next();
  final int dr = this.rentals.get(each);
  switch (each.getPriceCode()) {
  case Movie.REGULAR:
   thisAmount += 2;
   if (dr > 2)
    thisAmount += (dr - 2) * 1.5;
   break;
  case Movie.NEW_RELEASE:
   thisAmount += dr * 3;
   break;
  case Movie.CHILDRENS:
   thisAmount += 1.5;
   if (dr > 3)
    thisAmount += (dr - 3) * 1.5;
   break;
  }
  frequentRenterPoints++;
  if (each.getPriceCode() != null &&
    (each.getPriceCode() == Movie.NEW_RELEASE) &&
    dr > 1)
   frequentRenterPoints++;

  result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
  totalAmount += thisAmount;
 }
 result += "Amount owed is " + totalAmount + "\n";
 result += "You earned " + frequentRenterPoints + " frequent renter points";
 return result;
}
```

```java
public String printStatement() {
 return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
 return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
 return rentals.stream()
    .map(this::printLine)
    .collect(joining());
}

private String printFooterStatement() {
 return "Amount owed is " + calculateTotalPrice() + "\n" +
   "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
 return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
 return rentals.stream()
    .mapToDouble(Rental::getPrice)
    .sum();
}

private int calculateFrequentRenterPoints() {
 return rentals.stream()
    .mapToInt(Rental::calculateFrequentRenterPoints)
    .sum();
}
```

Reasonable abstraction

```
public String printStatement() {
  double totalAmount = 0;
  int frequentRenterPoints = 0;
  final Iterator<Movie> rentals = this.rentals.keySet().iterator();
  String result = "Rental Record for " + getName() + "\n";
  while (rentals.hasNext()) {
    double thisAmount = 0;
    final Movie each = rentals.next();
    final int dr = this.rentals.get(each);
    switch (each.getPriceCode()) {
    case Movie.REGULAR:
      thisAmount += 2;
      if (dr > 2)
        thisAmount += (dr - 2) * 1.5;
      break;
    case Movie.NEW_RELEASE:
      thisAmount += dr * 3;
      break;
    case Movie.CHILDRENS:
      thisAmount += 1.5;
      if (dr > 3)
        thisAmount += (dr - 3) * 1.5;
      break;
    }
    frequentRenterPoints++;
    if (each.getPriceCode() != null &&
        (each.getPriceCode() == Movie.NEW_RELEASE) &&
        dr > 1)
      frequentRenterPoints++;
    result += "\t" + each.getTitle() + "\t" + thisAmount + "\n";
    totalAmount += thisAmount;
  }
  result += "Amount owed is " + totalAmount + "\n";
  result += "You earned " + frequentRenterPoints + " frequent renter points";
  return result;
}
```

```
public String printStatement() {
  return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
  return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
  return rentals.stream()
      .map(this::printLine)
      .collect(joining());
}

private String printFooterStatement() {
  return "Amount owed is " + calculateTotalPrice() + "\n" +
    "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
  return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
  return rentals.stream()
      .mapToDouble(Rental::getPrice)
      .sum();
}

private int calculateFrequentRenterPoints() {
  return rentals.stream()
      .mapToInt(Rental::calculateFrequentRenterPoints)
      .sum();
}
```

Appropriate formatting

```
public String printStatement() { return printHeaderStatement()+printBodyStatement()+printFooterStatement(); }
private String printHeaderStatement() { return "Rental Record for " + name + "\n"; }
private String printBodyStatement() { return rentals.stream() .map(this::printLine) .collect(joining()); }
private String printFooterStatement() { return "Amount owed is " + calculateTotalPrice() + "\n" +
"You earned " + calculateFrequentRenterPoints() + " frequent renter points"; }
private String printLine(Rental rental) { return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";}
private double calculateTotalPrice() { return rentals.stream() .mapToDouble(Rental::getPrice) .sum(); }
private int calculateFrequentRenterPoints() { return
rentals.stream().mapToInt(Rental::calculateFrequentRenterPoints).sum();}
```

Appropriate formatting

```
public String printStatement() {
  return printHeaderStatement() + printBodyStatement() + printFooterStatement();
}

private String printHeaderStatement() {
  return "Rental Record for " + name + "\n";
}

private String printBodyStatement() {
  return rentals.stream()
      .map(this::printLine)
      .collect(joining());
}

private String printFooterStatement() {
  return "Amount owed is " + calculateTotalPrice() + "\n" +
    "You earned " + calculateFrequentRenterPoints() + " frequent renter points";
}

private String printLine(Rental rental) {
  return "\t" + rental.getMovie().getTitle() + "\t" + rental.getPrice() + "\n";
}

private double calculateTotalPrice() {
  return rentals.stream()
      .mapToDouble(Rental::getPrice)
      .sum();
}

private int calculateFrequentRenterPoints() {
  return rentals.stream()
      .mapToInt(Rental::calculateFrequentRenterPoints)
      .sum();
}
```

Knowing how to use comments

- Comments = failure to write self-explanatory code
- Try to avoid commenting at all costs!
- Sometimes neccessary evil

Knowing how to use comments

- Inserting ToDo’s while WIP
  //TODO: Clean up this mess!

- Explaining complex algorithms

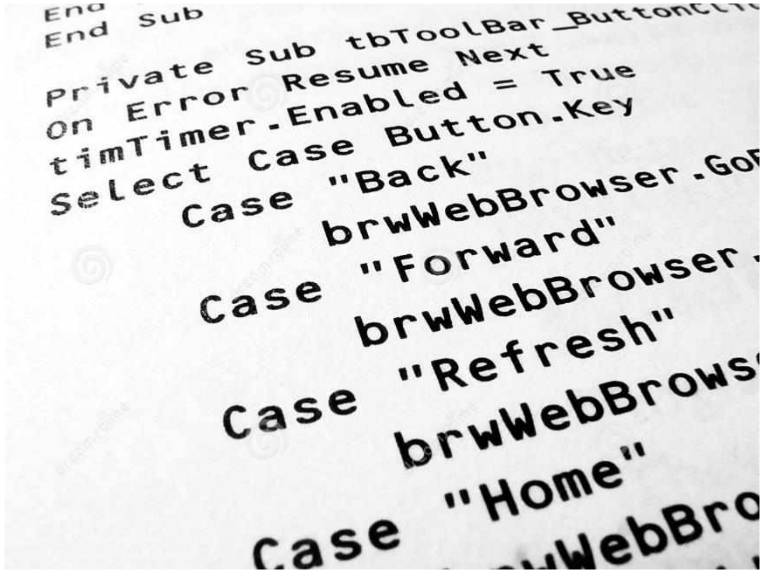
```
/**
* This class generates prime numbers
* up to a user specified maximum.
*/
```

- Explaining unorthodox approaches
  // Neccessary due to bug in apache library 2.0.5

Who do we code for?

Developer

Maschine language



```
End
End Sub

Private Sub tbToolBar_ButtonCl
On Error Resume Next
timTimer.Enabled = True
Select Case Button.Key
    Case "Back"
        brwWebBrowser.Go
    Case "Forward"
        brwWebBrowser.
    Case "Refresh"
        brwWebBrows
    Case "Home"
        wWebBro
```

Any fool can write code that a computer can understand.

Good programmers write code that humans can understand.

Martin Fowler