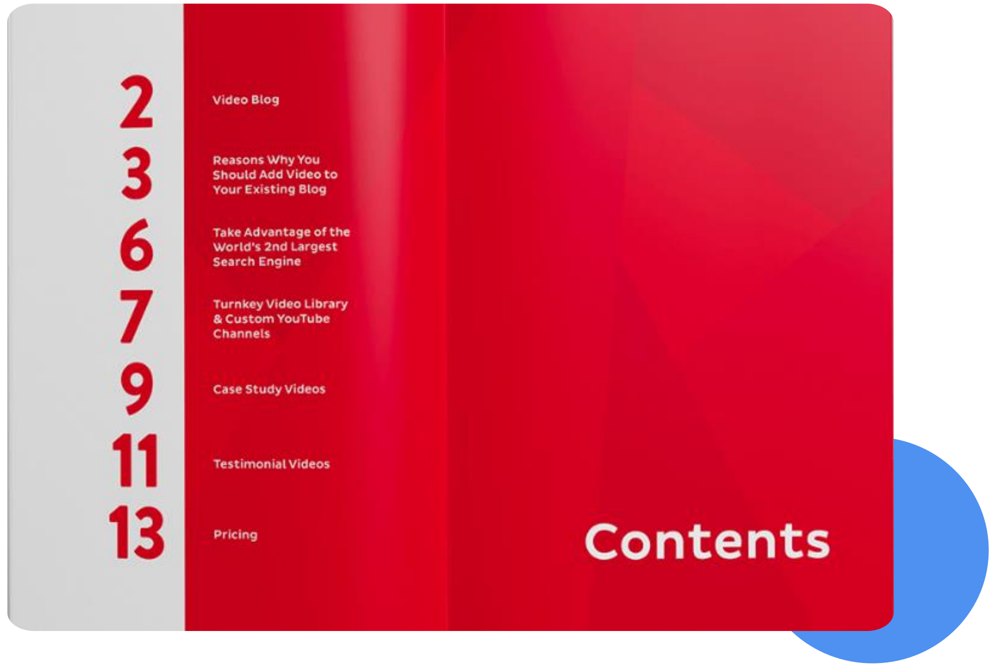
Code Quality

Vikrant Singh
University of Paderborn
PG- Data Portals
6864044

"We like to think we spend our time power typing,
but we actually spend most of our time staring into the abyss."

- Douglas Crockford
principal discoverer of JSON

Table of Contents

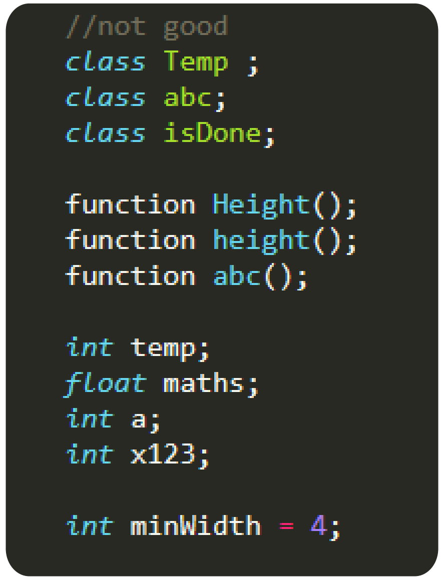

2 Video Blog
3 Reasons Why You Should Add Video to Your Existing Blog
6 Take Advantage of the World's 2nd Largest Search Engine
7 Turnkey Video Library & Custom YouTube Channels
9 Case Study Videos
11 Testimonial Videos
13 Pricing
Contents

Coding Conventions

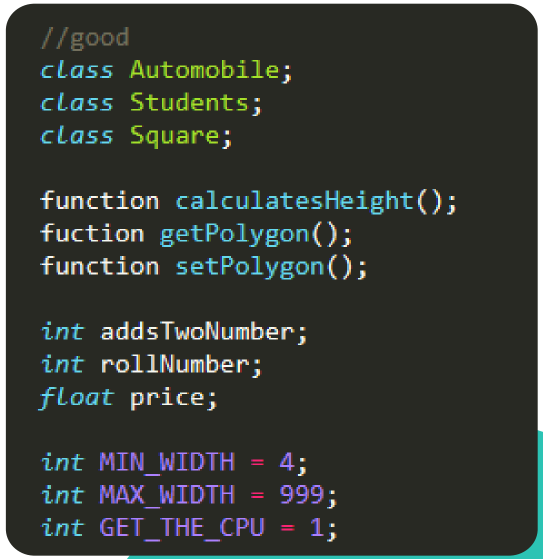
”write code that minimizes the time it would take someone to understand it.”

Naming Conventions

- Name should convey the meaning as what does this class, method or variable do.
- **Class Name** – Noun and UpperCamelCase.
- **Method Name** – Verb and lowerCamelCase.
- **Variable Name** – short and meaningful and lowerCamelCase.
- **Constants** – Uppercased with words separated with underscores.



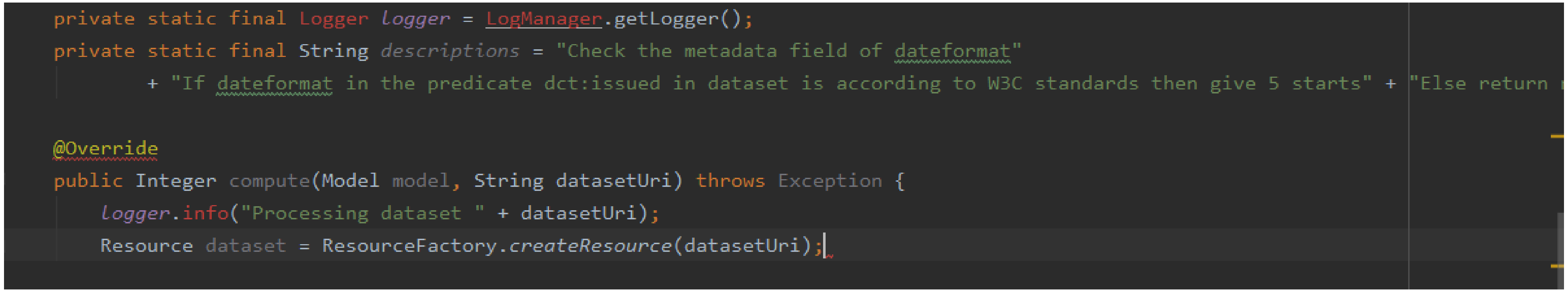
```
//not good
class Temp ;
class abc;
class isDone;

function Height();
function height();
function abc();

int temp;
float maths;
int a;
int x123;

int minWidth = 4;
```



```
//good
class Automobile;
class Students;
class Square;

function calculatesHeight();
fuction getPolygon();
function setPolygon();

int addsTwoNumber;
int rollNumber;
float price;

int MIN_WIDTH = 4;
int MAX_WIDTH = 999;
int GET_THE_CPU = 1;
```

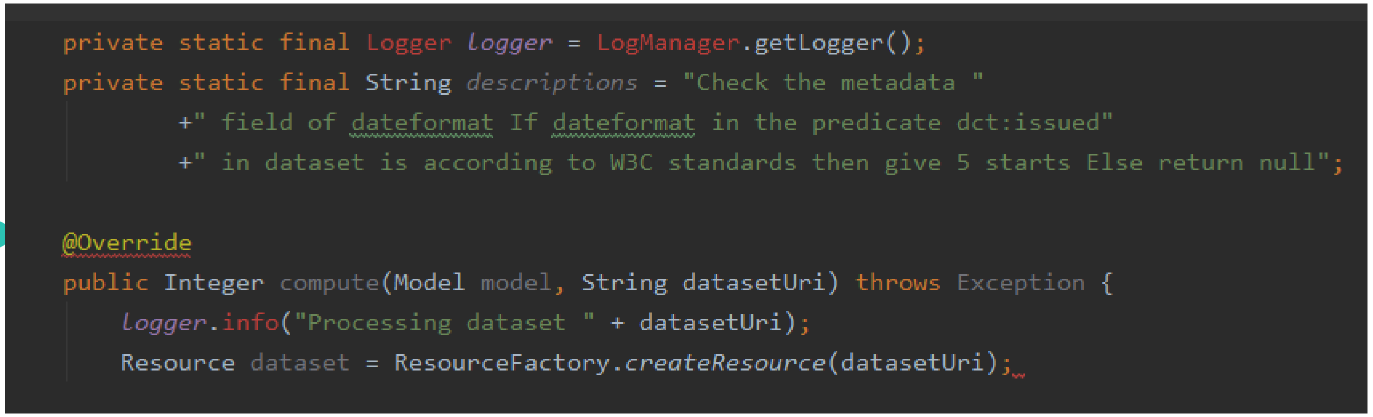
Indentation

- Four spaces should be used as the unit of indentation.
- **Line Length** : 60-80 Char per line.



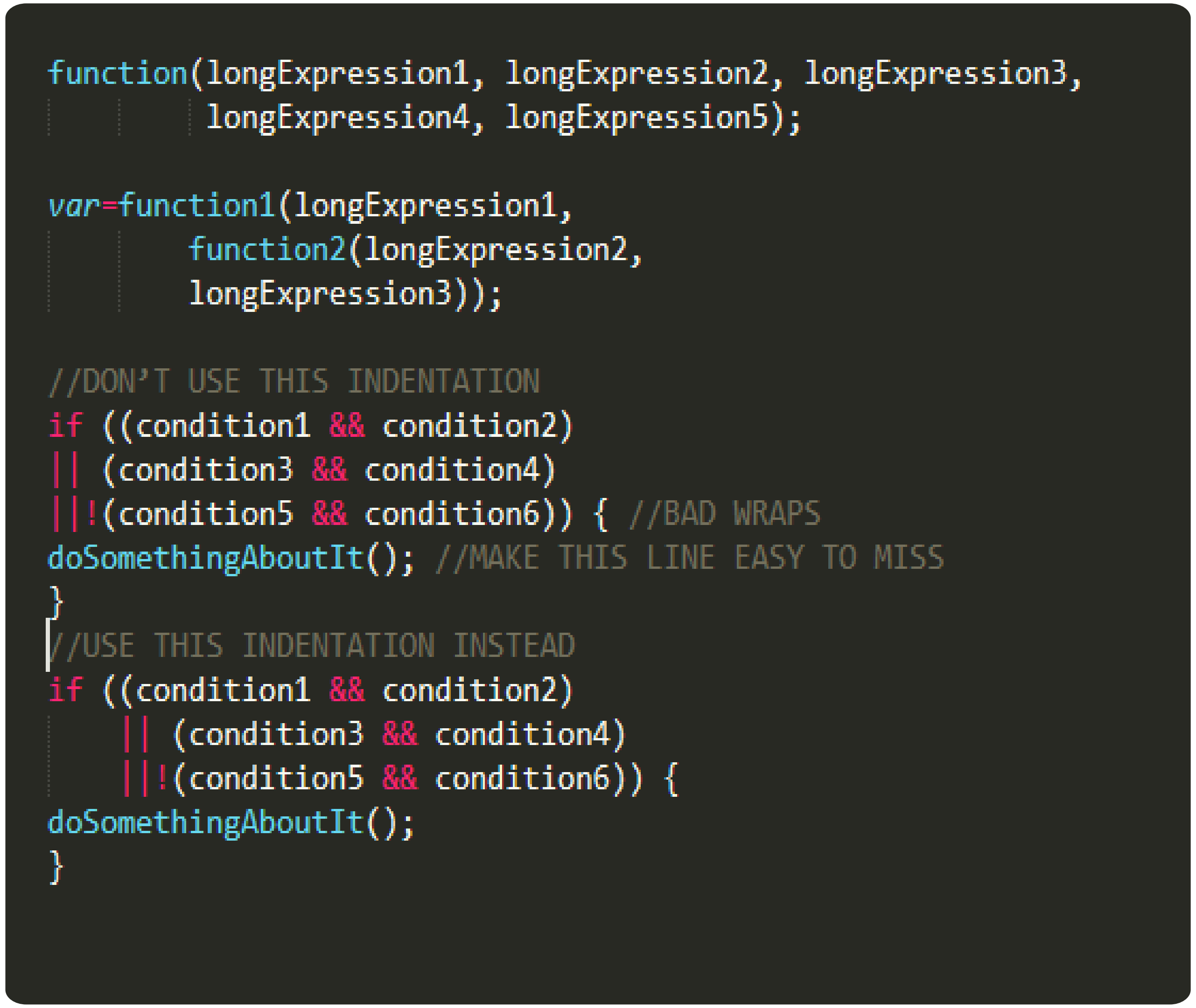
```
private static final Logger logger = LogManager.getLogger();
private static final String descriptions = "Check the metadata field of dateformat"
        + "If dateformat in the predicate dct:issued in dataset is according to W3C standards then give 5 starts" + "Else return 

@Override
public Integer compute(Model model, String datasetUri) throws Exception {
    logger.info("Processing dataset " + datasetUri);
    Resource dataset = ResourceFactory.createResource(datasetUri);
```

Fig 1) Too much chars in the line



```
private static final Logger logger = LogManager.getLogger();
private static final String descriptions = "Check the metadata "
        +" field of dateformat If dateformat in the predicate dct:issued"
        +" in dataset is according to W3C standards then give 5 starts Else return null";

@Override
public Integer compute(Model model, String datasetUri) throws Exception {
    logger.info("Processing dataset " + datasetUri);
    Resource dataset = ResourceFactory.createResource(datasetUri);
```

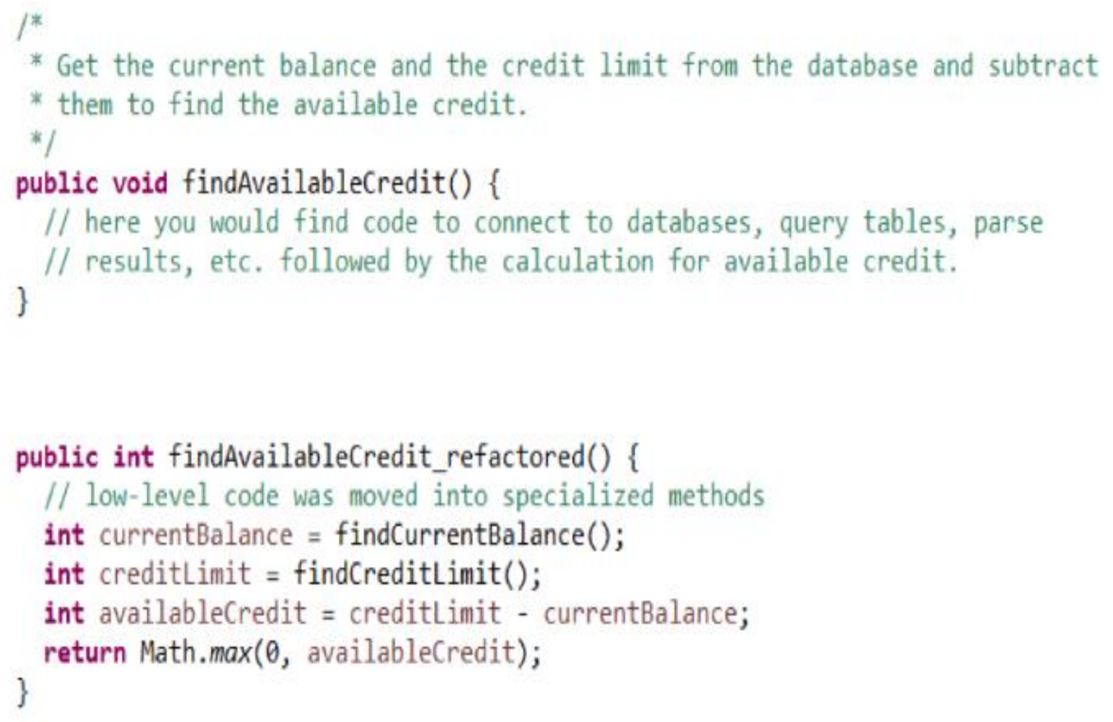
Fig 2) After Indentation



```
function(longExpression1, longExpression2, longExpression3,
        longExpression4, longExpression5);

var=function1(longExpression1,
        function2(longExpression2,
        longExpression3));

//DON'T USE THIS INDENTATION
if ((condition1 && condition2)
|| (condition3 && condition4)
||!(condition5 && condition6)) { //BAD WRAPS
doSomethingAboutIt(); //MAKE THIS LINE EASY TO MISS
}
//USE THIS INDENTATION INSTEAD
if ((condition1 && condition2)
    || (condition3 && condition4)
    ||!(condition5 && condition6)) {
doSomethingAboutIt();
}
```

Fig.3) Examples to wrap the lines for better indentation

- **Wrapping Lines** –
- Break after a comma , before an operator.



```
/*
 * Get the current balance and the credit limit from the database and subtract
 * them to find the available credit.
 */
public void findAvailableCredit() {
  // here you would find code to connect to databases, query tables, parse
  // results, etc. followed by the calculation for available credit.
}


public int findAvailableCredit_refactored() {
  // low-level code was moved into specialized methods
  int currentBalance = findCurrentBalance();
  int creditLimit = findCreditLimit();
  int availableCredit = creditLimit - currentBalance;
  return Math.max(0, availableCredit);
}
```

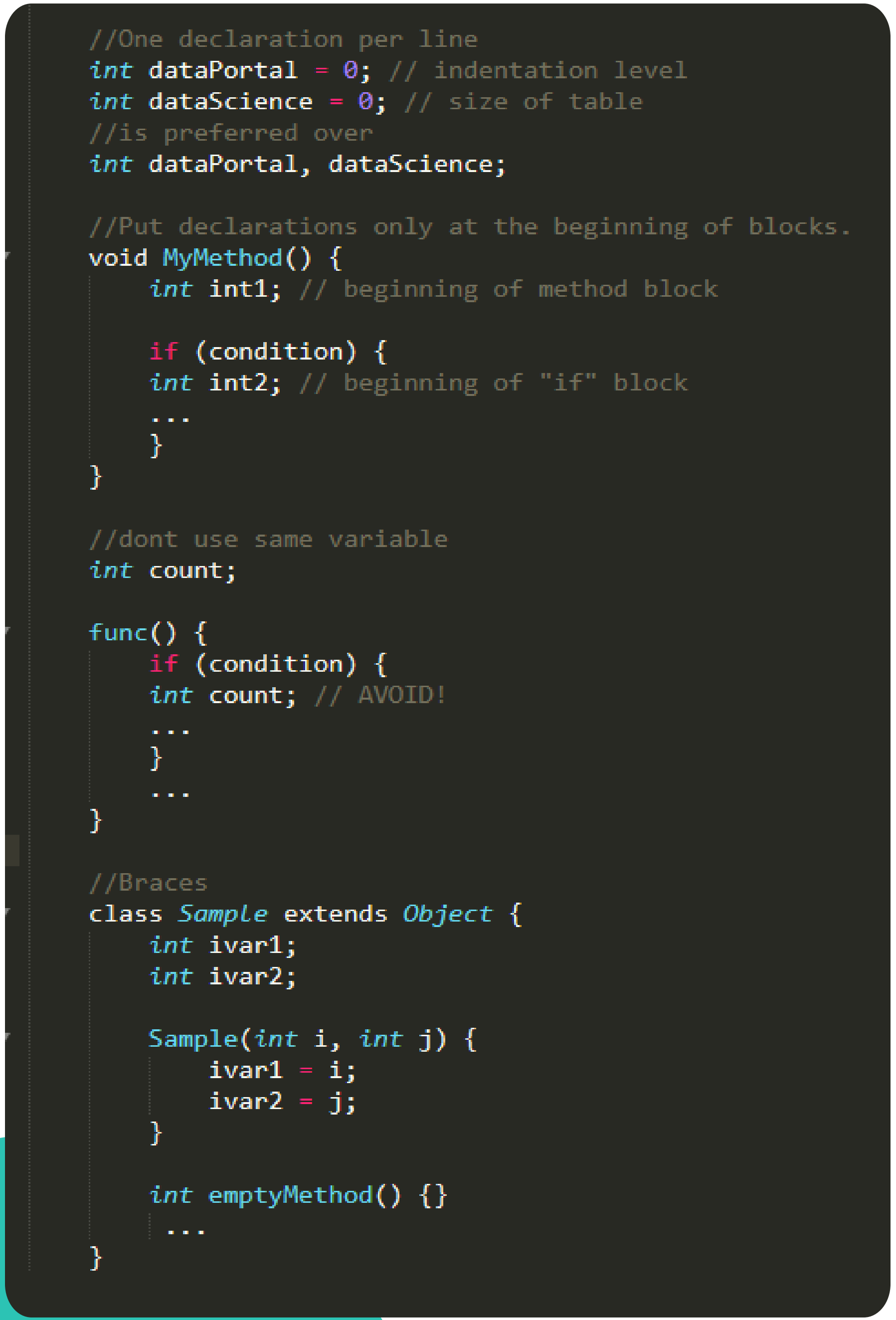
Fig.4) Image showing how to write more code and less comments.

Comments in Code

- **"The proper use of comments is to compensate for our failure to express our self in code."**

— *Robert C. Martin, "Clean Code: A Handbook of Agile Software Craftsmanship"*

- Code should speak for itself.
- Comment only when necessary.

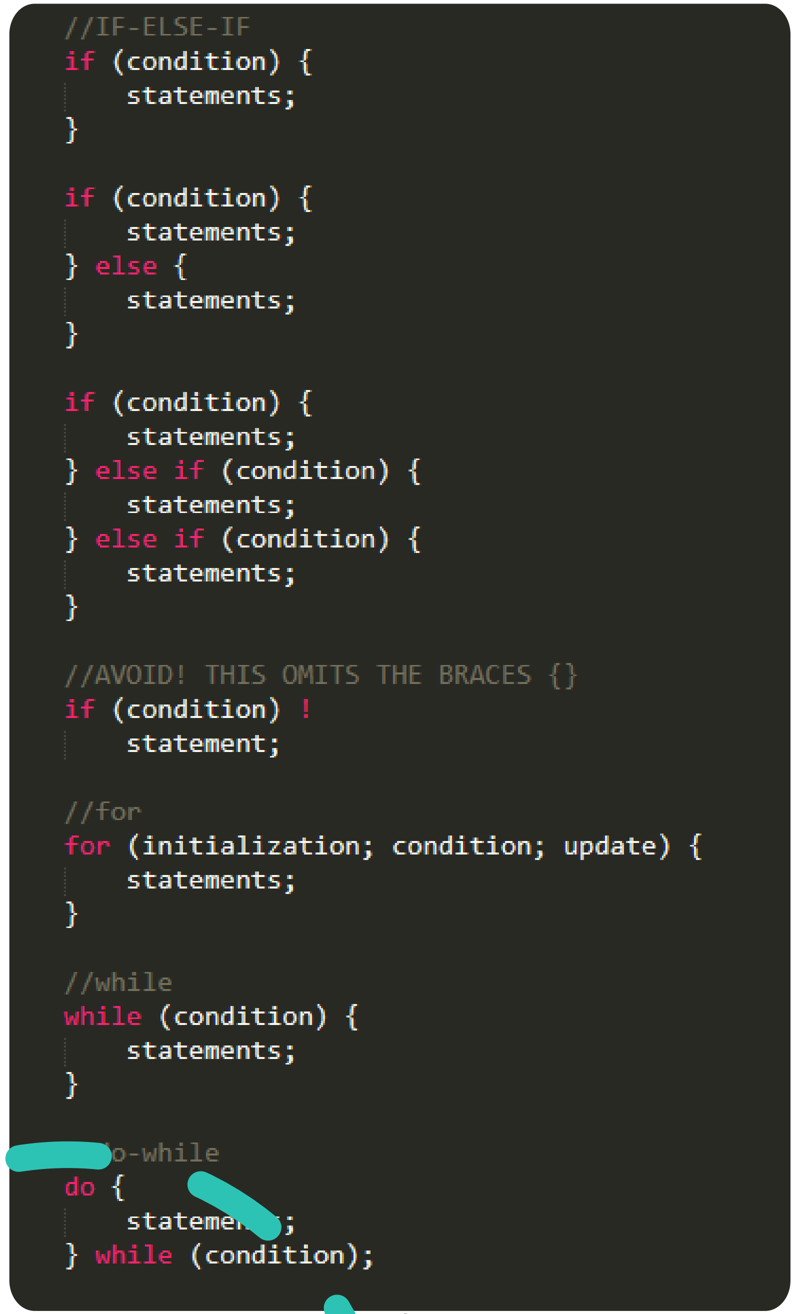


```
//One declaration per line
int dataPortal = 0; // indentation level
int dataScience = 0; // size of table
//is preferred over
int dataPortal, dataScience;

//Put declarations only at the beginning of blocks.
void MyMethod() {
    int int1; // beginning of method block

    if (condition) {
    int int2; // beginning of "if" block
    ...
    }
}

//dont use same variable
int count;

func() {
    if (condition) {
    int count; // AVOID!
    ...
    }
    ...
}

//Braces
class Sample extends Object {
    int ivar1;
    int ivar2;

    Sample(int i, int j) {
        ivar1 = i;
        ivar2 = j;
    }

    int emptyMethod() {}
     ...
}
```

Fig.5) Proper way of Declaring Vars.

Declarations

- One declaration per line .
- Initialize local variable when declared.
- Put declarations only at the beginning of blocks.
- Open brace "{" appears at the end of the same line as the declaration statement.
- Closing brace "}" should match its corresponding opening statement.

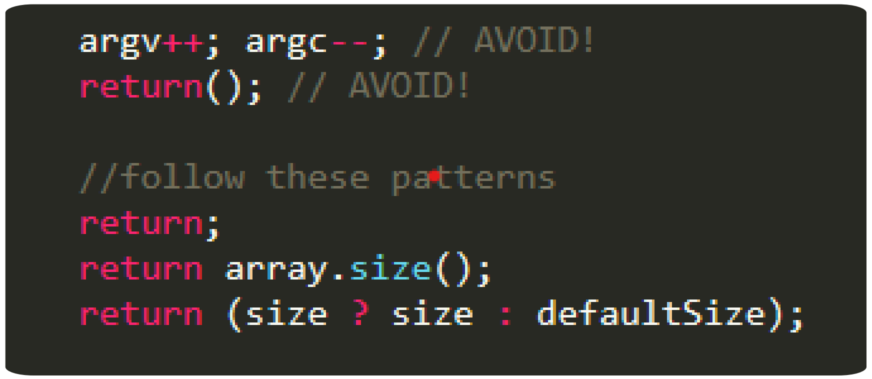


```
//IF-ELSE-IF
if (condition) {
    statements;
}

if (condition) {
    statements;
} else {
    statements;
}

if (condition) {
    statements;
} else if (condition) {
    statements;
} else if (condition) {
    statements;
}

//AVOID! THIS OMITS THE BRACES {}
if (condition) !
    statement;

//for
for (initialization; condition; update) {
    statements;
}

//while
while (condition) {
    statements;
}

//do-while
do {
    statements;
} while (condition);
```

Fig.6) Proper way of Using statements .

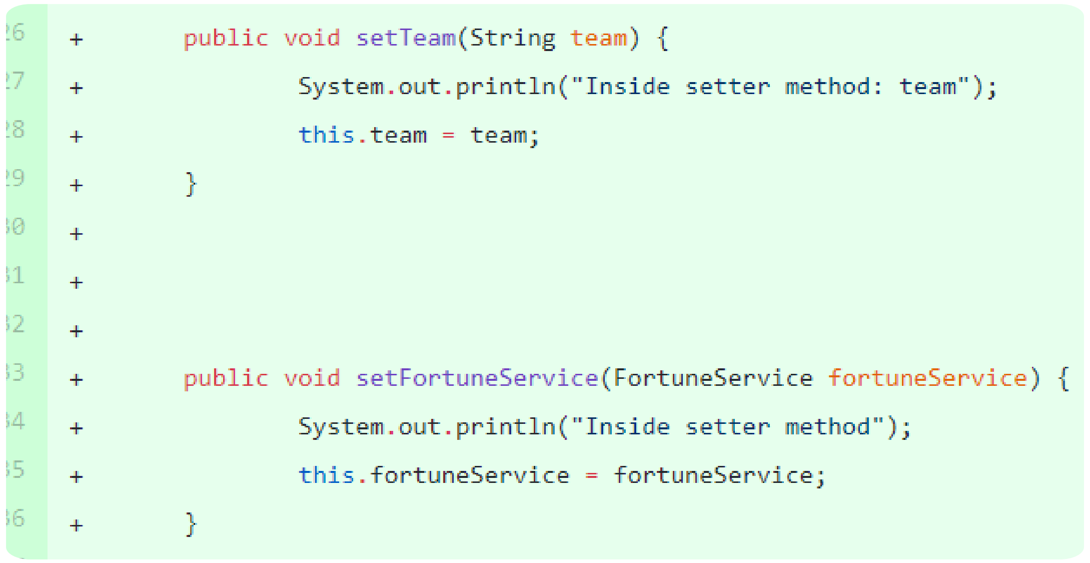
Statements

- One statement per line.
- Return statement without any value should be without parentheses.
- If-else , for , while , do-while Statements should have the form:



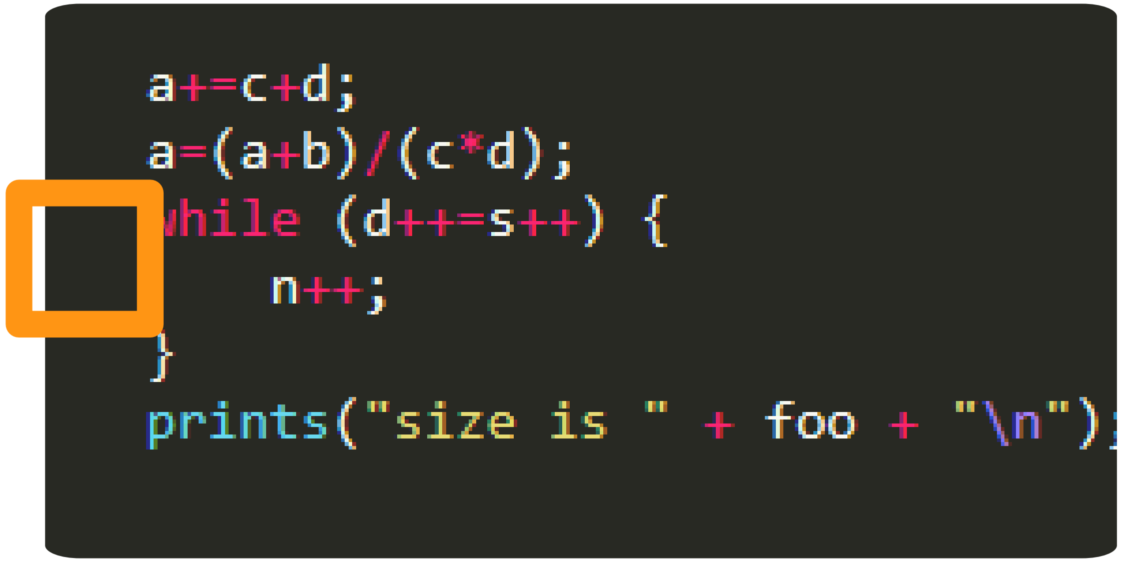
```
argv++; argc--; // AVOID!
return(); // AVOID!

//follow these patterns
return;
return array.size();
return (size ? size : defaultSize);
```

Fig.7) Proper way of Using return statements .



```
+       public void setTeam(String team) {
+               System.out.println("Inside setter method: team");
+               this.team = team;
+       }
+
+
+
+       public void setFortuneService(FortuneService fortuneService) {
+               System.out.println("Inside setter method");
+               this.fortuneService = fortuneService;
+       }
```

Fig.8) Too many blank lines.



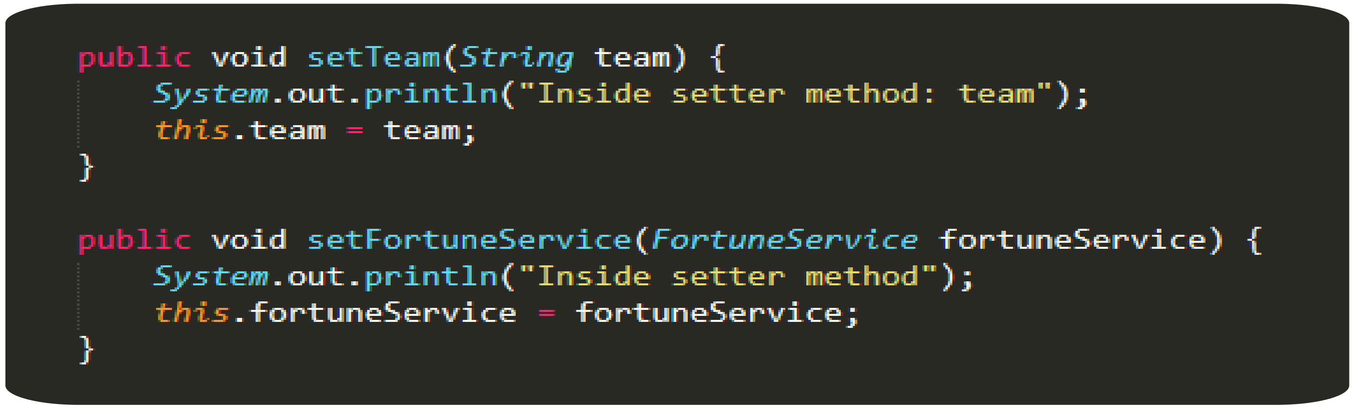
```
a+=c+d;
a=(a+b)/(c*d);
while (d++=s++) {
    n++;
}
prints("size is " + foo + "\n");
```

Fig.9) Improper way of using blank spaces.

White Spaces

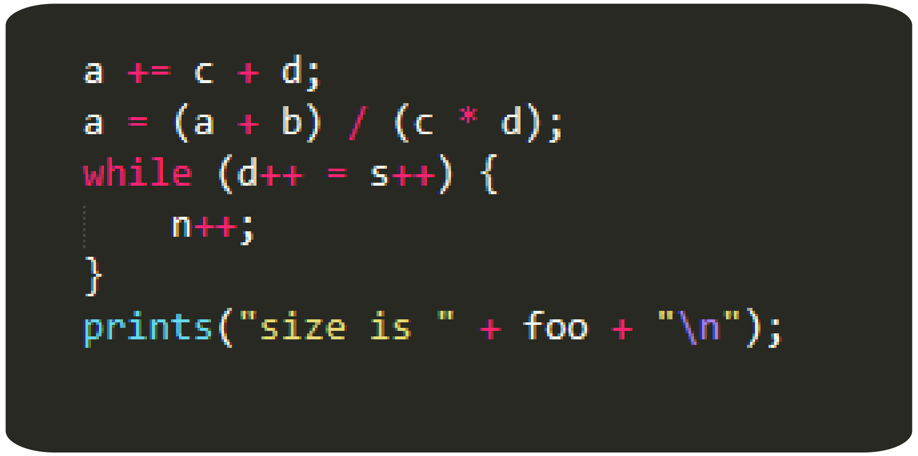
- Blank lines improve readability by setting off sections of code that are logically related.
- Put blank lines between methods , local variable and first statement and before comments.

- A keyword followed by a parenthesis should be separated by a space.
- Blank spaces should come between all binary operators(+,/,*,-,=) but not for ++,-- .



```java
public void setTeam(String team) {
    System.out.println("Inside setter method: team");
    this.team = team;
}

public void setFortuneService(FortuneService fortuneService) {
    System.out.println("Inside setter method");
    this.fortuneService = fortuneService;
}
```

Fig.10) Proper way of blank lines.

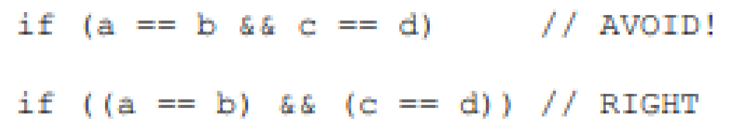


```
a += c + d;
a = (a + b) / (c * d);
while (d++ = s++) {
    n++;
}
prints("size is " + foo + "\n");
```

Fig.11) Proper way of Using blank spaces.

Miscellaneous Practices

- Delete Unused Variables.
- @Override: always use.
- Parentheses



```
if (a == b && c == d)     // AVOID!
if ((a == b) && (c == d)) // RIGHT
```

- Write Generic and Re-useable Code .
  If a particular code is getting used too often ,wrap that in a method.
- Less lines ,More Classes.
- Use Enums wherever possible.
- Write Unit Tests.
- Hard-Code as less as possible.*(Esp for UI-Design Implementation)*

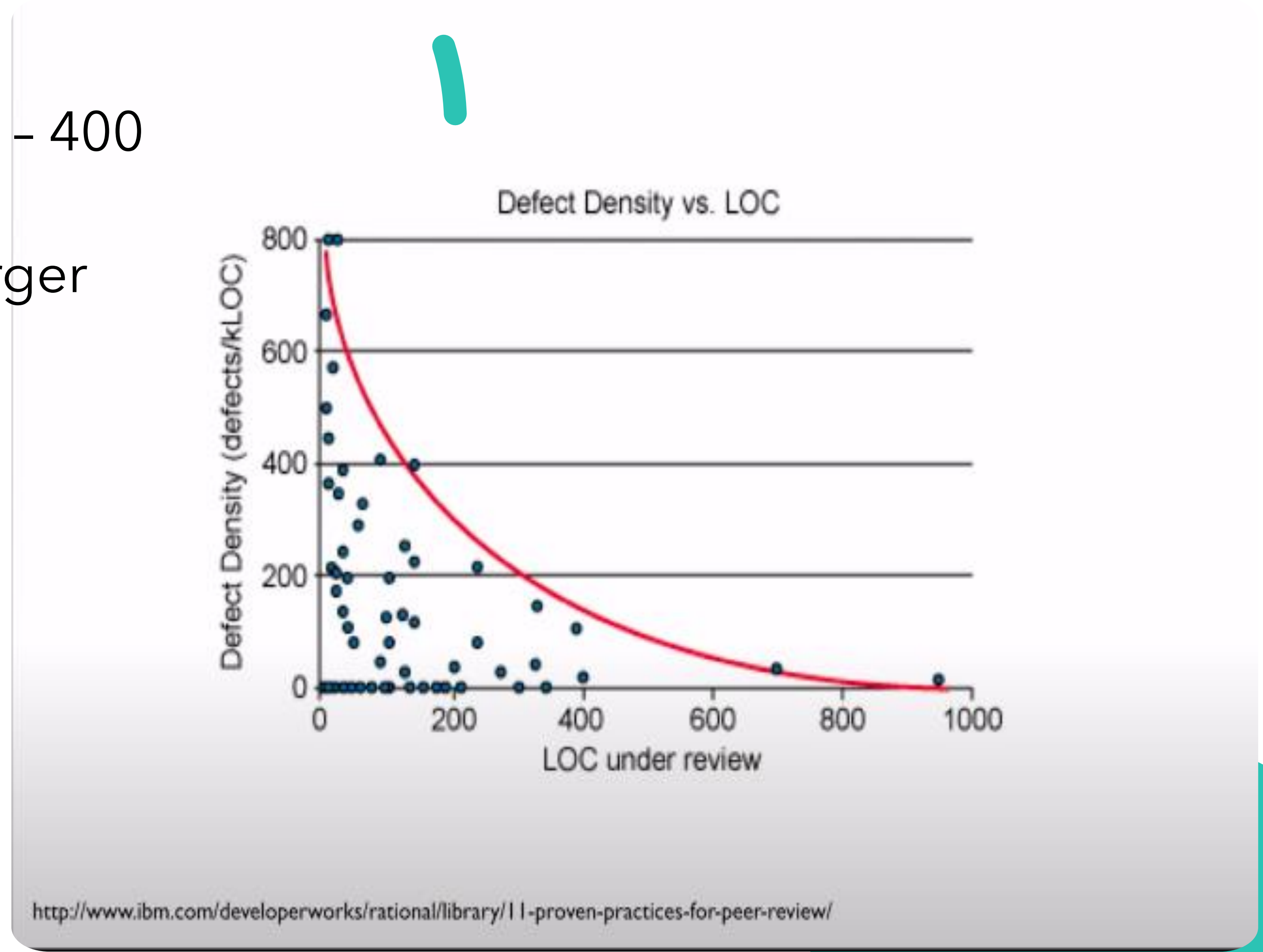
Code Quality on Github-

Code Reviews by Peers/Supervisors.

Automated Tools.

Code Reviews

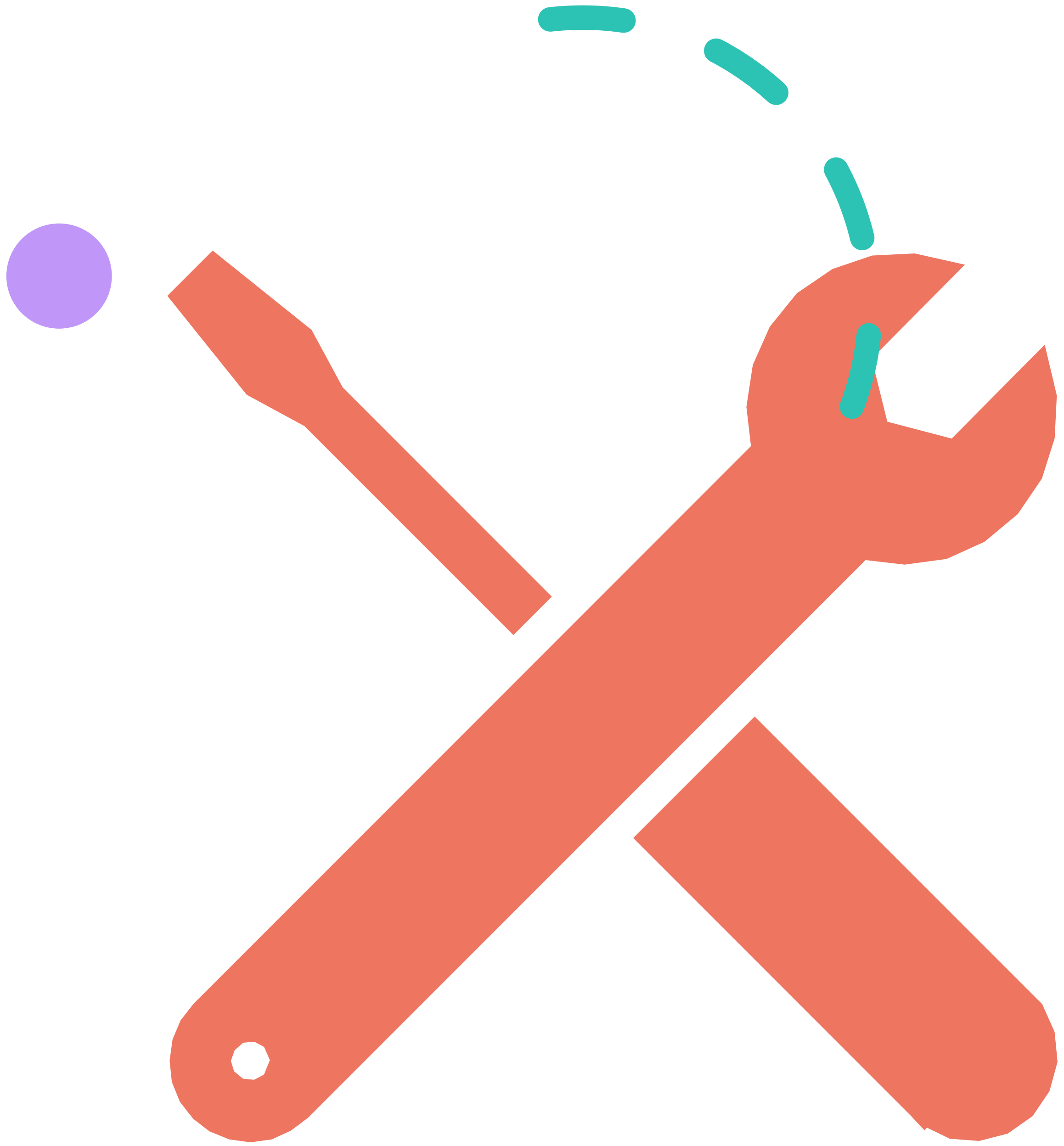
- Optimal Size of Pull Request – 400 LOC
- Fewer issues are found in larger pull requests .



http://www.ibm.com/developerworks/rational/library/11-proven-practices-for-peer-review/

Automated tools

- **Tortoise Git**.*(for windows)*
- **Codacy** - https://github.com/marketplace/codacy
- **Github Marketplace** - https://github.com/marketplace/category/code-review



IDEs

- IntelliJ IDEA
- Eclipse
- Others

IntelliJ IDEA

Code Inspection - highlight various problems, locate dead code, find probable bugs, spelling problems, and improve the overall code structure. (Ctrl+Alt+S)

Code Formatting – arranges code according to the requirements you've specified in the Code Style settings.(Ctrl+Alt+L)

For more - https://www.jetbrains.com/help/idea/reformat-and-rearrange-code.html

Eclipse

Code Inspection – Ctrl+Shift+I

Code formatting - Ctrl+I.

Other plugins - eclipse-pmd , Impact Tracer for Analyse JET

Suggestion or Questions ?

Task:

Even if it’s a temporary code , write that neatly. Use this presentation for better coding habits.

Thank You