“”

“”

“”

“”spaghetti code

treeif

```
if (...) {
  if (...) {
    ...
  } else {
    ...
  }
} else if (...) {
  ...
} else {
  ...
}
```

ifelseif

“”“module”VCS repo“”

“”

- 405040405040

  40

-

  inline

  macroC“”

- “”“”

  ```
  void foo() {
    if (getOS().equals("MacOS")) {
      a();
    } else {
      b();
    }
    c();
    if (getOS().equals("MacOS")) {
      d();
    } else {
      e();
    }
  }
  ```

“MacOS”`c()a(), b(), d(), e()`

“”

```
void fooMacOS() {
  a();
  c();
  d();
}



void fooOther() {
  b();
  c();
  e();
}



void foo() {
  a();
  b()
  c();
  if (getOS().equals("MacOS")) {
    d();
  } else {
    e();
  }
}

a()b()c()d()e()a()b()c()

void preFoo() {
  a();
  b()
  c();



void fooMacOS() {
  preFoo();
  d();
}



void fooOther() {
  preFoo();
  e();
}
```

- class member

```
 class A {
   String x;

   void findX() {
      ...
      x = ...;
   }

   void foo() {
     findX();
     ...
     print(x);
   }
 }

findX()xxxfindXprintxclass AXfindXfooclass A
```

class

```
String findX() {
   ...
   x = ...;
   return x;
}
void foo() {
  String x = findX();
  print(x);
}
```

……

1.

```
// put elephant1 into fridge2
put(elephant1, fridge2);
```

```
putelephant1fridge2
```

2.

```
void foo() {
  int index = ...;
  ...
  ...
  bar(index);
  ...
}
```

```
index
```

```
void foo() {
  ...
  ...
  int index = ...;
  bar(index);
  ...
}
```

`bar(index)index`“”indexindexindex

——

3. 2

```
boolean successInDeleteFile = deleteFile("foo.txt");
if (successInDeleteFile) {
  ...
} else {
  ...
}
```

`successInDeleteFiledeleteFilesuccess`”success in deleteFile”

```
boolean success = deleteFile("foo.txt");
if (success) {
  ...
} else {
  ...
}
```

`successInDeleteFile`“camelCase”

4. “”

```
String msg;
if (...) {
  msg = "succeed";
  log.info(msg);
} else {
  msg = "failed";
  log.info(msg);
}

msglog.info

if (...) {
  String msg = "succeed";
  log.info(msg);
} else {
  String msg = "failed";
  log.info(msg);
}

msgifmsg
```

5. “”

```
...
// put elephant1 into fridge2
openDoor(fridge2);
if (elephant1.alive()) {
  ...
} else {
   ...
}
closeDoor(fridge2);
...



void put(Elephant elephant, Fridge fridge) {
  openDoor(fridge);
  if (elephant.alive()) {
    ...
  } else {
     ...
  }
  closeDoor(fridge);
}



...
put(elephant1, fridge2);
...
```

6. “”

```
Pizza pizza = makePizza(crust(salt(), butter()),
   topping(onion(), tomato(), sausage()));



Crust crust = crust(salt(), butter());
Topping topping = topping(onion(), tomato(), sausage());
Pizza pizza = makePizza(crust, topping);

“”
```

7.

IDEIDE

```
 if (someLongCondition1() && someLongCondition2() && someLongCondition3() &&
```

```
    someLongCondition4()) {
    ...
  }
```

`someLongCondition4()`boolean`&&`

```
  if (someLongCondition1() &&
      someLongCondition2() &&
      someLongCondition3() &&
      someLongCondition4()) {
    ...
  }
```

```
  log.info("failed to find file {} for command {}, with exception {}", file, command,
    exception);
```

`filecommandexception`

```
  log.info("failed to find file {} for command {}, with exception {}",
    file, command, exception);
```

IDEIDEIntelliJ“”IDE

“”ChaiJavaScript

```
expect(foo).to.be.a('string');
expect(foo).to.equal('bar');
expect(foo).to.have.length(3);
expect(tea).to.have.property('flavors').with.length(3);
```

“”“”“”“”

“”“”“”

- i++++ii−−i

  ii`foo(i++)int t = i; i += 1; foo(t);foo(++i)i += 1; foo(i);`

  i++++iforupdate`for(int i = 0; i < 5; i++)i++;foo(i++)foo(++i) + foo(i)`.....

- CJavaif

  ```
  if (...)
    action1();
  ```

  `action2()`if

  ```
  if (...)
    action1();
    action2();
  ```

  `action1()`if`action2()`if“”optical illusion

  `action2()action2()`ifif-elseCJava

- `1 + 2 * 32 << 7 - 2 * 3`

  `<<x << 1x`2`(2 << 7) - (2 * 3)`250`<<+2 << (7 - 2 * 3)`4

  `2 << (7 - 2 * 3)<<`

- continuebreakforwhilereturncontinuebreak

  continuebreakcontinuebreakcontinuebreak

  1. continuecontinuecontinue
  2. breakbreakbreak
  3. breakreturnbreak
  4. continuebreak

  1continue

```
List<String> goodNames = new ArrayList<>();
for (String name: names) {
  if (name.contains("bad")) {
    continue;
  }
  goodNames.add(name);
  ...
}
```

  “name’bad’……” “”””””continuecontinuebreak“”””””””

  continuecontinue

```
List<String> goodNames = new ArrayList<>();
for (String name: names) {
  if (!name.contains("bad")) {
    goodNames.add(name);
    ...
  }
}
```

  `goodNames.add(name);`ifcontinue“””name’bad’goodNames……”

  2forwhile“”breakbreak

```
while (condition1) {
  ...
  if (condition2) {
    break;
  }
}
```

  conditionbreakcondition2whilebreak

```
while (condition1 && !condition2) {
  ...
}
```

  breakbreak

  3breakreturnbreakreturn

```
public boolean hasBadName(List<String> names) {
    boolean result = false;

    for (String name: names) {
        if (name.contains("bad")) {
            result = true;
            break;
        }
    }
    return result;
}
```

  names“bad”break

```
public boolean hasBadName(List<String> names) {
    for (String name: names) {
        if (name.contains("bad")) {
```

```
                return true;
            }
        }
        return false;
    }
```

name“bad”`return true`resultbreakreturnfalsereturnbreakbreakresult

continuebreak99%breakcontinuereturnif1%

Unix“”

```
command1 && command2 && command3
```

Shell `a && b`“”afalseb command1 command2 command2 command3

```
command1 || command2 || command3
```

||command1command2command3command1command2command3

if

```
if (action1() || action2() && action3()) {
  ...
}
```

action2action3“action1action2action2action3”“”“”“”||||“action1……”

&&||“”ifififif

```
if (!action1()) {
  if (action2()) {
    action3();
  }
}
```

action1()action2()action2()action3()`if=!=`……

ifif

```
if (...) {
  if (...) {
    ...
    return false;
  } else {
    return true;
  }
} else if (...) {
  ...
  return false;
} else {
  return true;
}
```

corner caseififififelse

ifelseelseelse`return true`else`return true`elseif“”`return true`

```
if (...) {
  if (...) {
    ...
    return false;
  }
} else if (...) {
  ...
  return false;
}
```

```
return true;
```

ifelse“”elseif&&||

“”“”

else

```
String s = "";
if (x < 5) {
  s = "ok";
}
```

“”snullx<5mutate“ok”x<5sss

```
String s;
if (x < 5) {
  s = "ok";
} else {
  s = "";
}
```

x<5s""“”ss“”

elseJava“s”s

```
String s = x < 5 ? "ok" : "";
```

if

ifif

“”“”

UnixAPILinuxread

```
RETURN VALUE
On success, the number of bytes read is returned...

On error, -1 is returned, and errno is set appropriately.

ERRORS

EAGAIN, EBADF, EFAULT, EINTR, EINVAL, ...
```

read-1read-1-1“”

JavaJavaexception“union”

```
String foo() throws MyException {
  ...
}
```

MyExceptionunion{String, MyException}fooMyExceptionUnionTyped Racketunion

JavaJava“catchfoo”

```
try {
  foo();
} catch (Exception e) {}
```

logthrows Exception

catchfoo“”

catchExceptioncatchAcatchBAcatchExceptionBAbugdebugger

`throws Exceptionthrows Exception`

try { ... } catch`foobar`A

```
try {
  foo();
} catch (A e) {...}

try {
  bar();
} catch (A e) {...}



try {
  foo();
  bar();
} catch (A e) {...}
```

catchlog

null

null

CC++JavaC#.....null[Tony Hoare](#)Hoare“[billion dollar mistake](#)”

nullnullStringIntegernullNULLnull

- nullnullnull“”””JavaJavaunionfindString

  ```
  public String find() throws NotFoundException {
    if (...) {
      return ...;
    } else {
      throw new NotFoundException();
    }
  }
  ```

  JavacatchNotFoundExceptionnullJava

  Javatry...catch`find`null“”try...catchnull“”null“”””””””””null

- catch NullPointerExceptionnice“”null

  ```
  void foo() {
    String found = find();
    int len = found.length();
    ...
  }
  ```

  foo

  ```
  try {
    foo();
  } catch (Exception e) {
    ...
  }
  ```

  foundnullNullPointerException`catch (Exception e)`NullPointerExceptiontryNullPointerException

  `catch (NullPointerException e)`foonullNullPointerExceptioncatchfoofoo

  ```
  void foo() {
    String found = find();
    if (found != null) {
      int len = found.length();
      ...
    } else {
      ...
    }
  }
  ```

nullnull

- null“”collectionnullArrayListSetMapkeyvaluenullnullnull

  “”ArrayListSetMapentry“”

  null

  ```
  class A {
    String name = null;
    ...
  }
  ```

  nullAnamenullnamenull

- nullnullnull“”“”“”“”“”..... null

  nullnull“”`find`null“”`find`null“”

  “”null“”

  ```
  public String foo() {
    String found = find();
    if (found == null) {
      return null;
    }
  }
  ```

  find()nullfoonullnullnull

  ```
  public void foo(A a, B b, C c) {
    if (a == null) { ... }
    if (b == null) { ... }
    if (c == null) { ... }
    ...
  }
  ```

- nullnullnull“”null

  null“”“”“”“”null”nullnonsensenullnullnullnull

  nullnullnull

  ```
  Objects.requireNonNull()
  ```

  ```
  public static <T> T requireNonNull(T obj) {
    if (obj == null) {
      throw new NullPointerException();
    } else {
      return obj;
    }
  }
  ```

  nullnull`NullPointerException`null

- @NotNull@NullableIntelliJ@NotNull@NullablenullIntelliJ`NullPointerException`null `IllegalArgumentException`nulldeferencenull

- OptionalJava 8SwiftOptionalnullnull“”null“”

  Optional“”“”“”“”MLHaskellpattern matching

  Swift

  ```
  let found = find()
  if let content = found {
    print("found: " + content)
  }
  ```

  `find()`Optional`foundString?`StringnilifnullififBool`let content = found`

  foundnilifnilcontentfoundunwrap`print("found: " + content)`

Java 8 `Optional<String>`found“”

```
Optional<String> found = find();
found.ifPresent(content -> System.out.println("found: " + content));
```

JavaSwift“”“”ifPresentfoundnull“”lambdacontentunwraplambdafoundifPresentlambda

Javanulllambdalambda“continuation”Java “Consumer”“foundnull”lambda`return`null

```
public static String foo() {
  String found = find();
  if (found != null) {
    return found;
  } else {
    return "";
  }
}
```

```
public static String foo() {
  Optional<String> found = find();
  found.ifPresent(content -> {
    return content;    // can't return from foo here
  });
  return "";
}
```

`return afoo`lambdalambdaConsumer.accept`void`StringJavaclosurelambda

```
public static String foo() {
  Optional<String> found = find();
  String result = "";
  found.ifPresent(content -> {
    result = content;    // can't assign to result
  });
  return result;
}
```

lambdafoundJavanullJava 8`map`, `flatMap`, `orElse`

```
public static String foo() {
  Optional<String> found = find();
  return found.orElse("");
}
```

Java 8Optionalfunctorcontinuationmonad..... OptionalJava

JavaOptionalSwift`if let content = found {...}`

Optional“”nullJava 8`found.get()`foundSwift`found!`

JavaOptional

```
Option<String> found = find();
if (found.isPresent()) {
  System.out.println("found: " + found.get());
}
```

`if (found.isPresent())`null`found.isPresent()found.get()NoSuchElementExceptionNullPointerException`null
Optional“”

over-engineering

“”“”“”“”.....

“”“”

“”“”“”

“”“”bug

“bug”“bug”“bug”coveragebug“bug”

1.
2.
3. bug