

ارائه پروژه

Object Oriented Mutation Testing

گردآوری:

یزدان افرا، نسترن سرخوش، محمدامین کیانی

استاد راهنما:

دکتر شعرفاف



فهرست مطالب

01
توضیح
راهکار

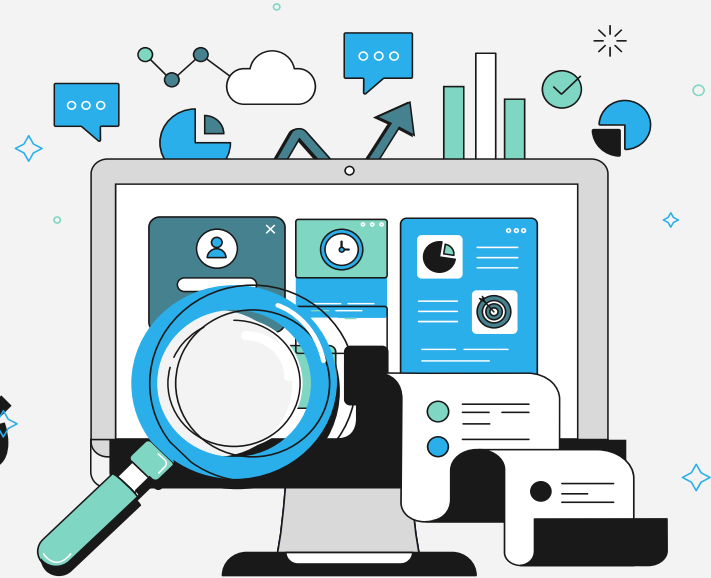
02
مثال‌های
اجرائی

03
Mutation
Score



01

توضیح راهکار



راهکار اصلی: Abstraction Syntax Tree



AST چیست؟

در واقع یک نمایش سلسله
مراتبی و ساختار یافته از کد منبع
است. این درخت، ساختار
دستوری کد را به صورت یک
ساختار درختی نشان می‌دهد. هر
گره در این درخت یک عنصر
ساختاری در کد را نشان می‌دهد،
مانند کلاس‌ها، متدها، عبارات
...

ویژگی‌های آن

درک بهتر کد
تحلیل کد
تبدیل کد
ایجاد ابزارهای تست

نقش آن در پروژه

در این پروژه، درخت نحوی
انتزاعی AST به عنوان یک ساختار
داده‌ای مرکزی استفاده می‌شود تا
کد منبع جاوا را به صورت سلسله
مراتبی و ساختاریافته نمایش
دهد.

JavaParser: کتابخانه برای AST



JavaParser

چیست؟

JavaParser یک کتابخانه قدرتمند و پرکاربرد در جاوا است که برای تجزیه کد منبع جاوا به یک ساختار درختی انتزاعی طراحی شده است.

فرآیند کار

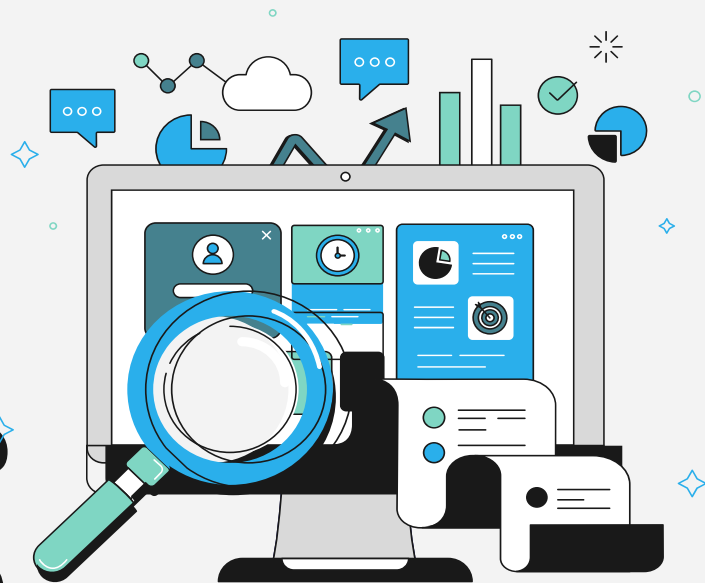
1. خواندن کد منبع
2. تجزیه لکسیگرافی
3. ساخت درخت نحوی انتزاعی
4. بازگشت ساختار درختی به صورت NodeList

اجزای اصلی

- CompilationUnit
- Node
- Visitor Pattern
- Parser

02

مثال‌های اجرای Mutation



AMC: Access Modifier Change

```
package mutants.AMC;

public class code {

    public static class Vehicle { 3 inheritors

        public String brand;

        public int speed;

        public int fuelCapacity;

        static public int vehicleCount = 0;

        public Vehicle() {
            this.brand = "brand";
            this.speed = 0;
            this.fuelCapacity = 0;
            vehicleCount++;
        }

        public Vehicle(String brand, int speed, int fuelCapacity) {
            this.brand = brand;
            this.speed = speed;
            this.fuelCapacity = fuelCapacity;
            vehicleCount++;
        }

        public int calculateRange() { return speed * fuelCapacity; }
```

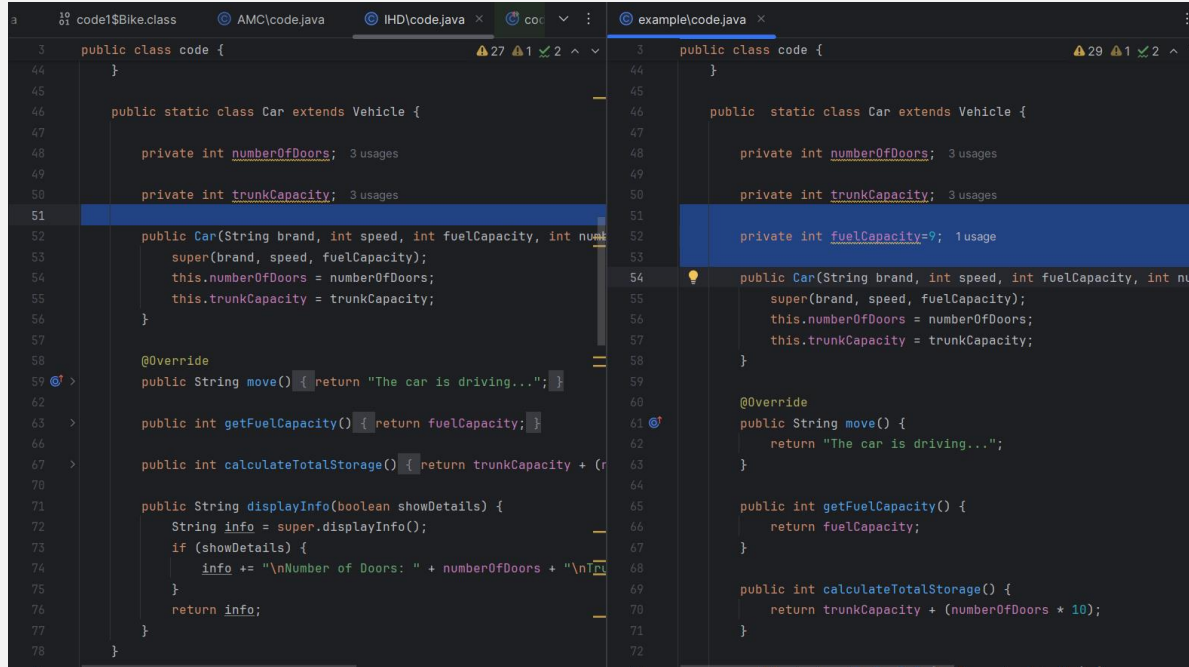
```
1 package org.example;
2
3 public class code {
4
5     public static class Vehicle { 3 inheritors
6
7         protected String brand;
8
9         private int speed; 4 usages
10
11         public int fuelCapacity;
12
13         public static int vehicleCount = 0;
14
15         public Vehicle() {
16             this.brand = "brand";
17             this.speed = 0;
18             this.fuelCapacity = 0;
19             vehicleCount++;
20         }
21
22         public Vehicle(String brand, int speed, int fuelCapacity) {
23             this.brand = brand;
24             this.speed = speed;
25             this.fuelCapacity = fuelCapacity;
26             vehicleCount++;
27         }
28
29         public int calculateRange() {
30             return speed * fuelCapacity;
31         }
32     }
```

IHI: Hiding Variable Insertion

```
public class code {  
    }  
  
    public static class Car extends Vehicle {  
        public static int vehicleCount = 0;  
  
        private int speed; no usages  
  
        protected String brand;  
  
        private int numberOfDoors; 3 usages  
  
        private int trunkCapacity; 3 usages  
  
        private int fuelCapacity = 9; 1 usage  
  
        public Car(String brand, int speed, int fuelCapacity, int num  
            super(brand, speed, fuelCapacity);  
            this.numberOfDoors = numberOfDoors;  
            this.trunkCapacity = trunkCapacity;  
        }  
  
        @Override  
        public String move() { return "The car is driving..."; }  
  
        public int getFuelCapacity() { return fuelCapacity; }  
  
        public int calculateTotalStorage() { return trunkCapacity + T  
  
        public String displayInfo(boolean showDetails) {  
    }  
}
```

```
public class code {  
    }  
  
    public static class Car extends Vehicle {  
        private int numberOfDoors; 3 usages  
  
        private int trunkCapacity; 3 usages  
  
        private int fuelCapacity=9; 1 usage  
  
        public Car(String brand, int speed, int fuelCapacity, int num  
            super(brand, speed, fuelCapacity);  
            this.numberOfDoors = numberOfDoors;  
            this.trunkCapacity = trunkCapacity;  
        }  
  
        @Override  
        public String move() {  
            return "The car is driving...";  
        }  
  
        public int getFuelCapacity() {  
            return fuelCapacity;  
        }  
  
        public int calculateTotalStorage() {  
            return trunkCapacity + (numberOfDoors * 10);  
        }  
  
        public String displayInfo(boolean showDetails) {  
    }  
}
```


IHD: Hiding Variable Deletion



```
code\SBike.class  AMC\code.java  IHD\code.java  example\code.java
3 public class code {
44 }
45
46 public static class Car extends Vehicle {
47
48     private int numberOfDoors; 3 usages
49
50     private int trunkCapacity; 3 usages
51
52     public Car(String brand, int speed, int fuelCapacity, int num
53         super(brand, speed, fuelCapacity);
54         this.numberOfDoors = numberOfDoors;
55         this.trunkCapacity = trunkCapacity;
56     }
57
58     @Override
59     public String move() { return "The car is driving..."; }
60
61
62
63     public int getFuelCapacity() { return fuelCapacity; }
64
65
66
67     public int calculateTotalStorage() { return trunkCapacity + (r
68
69
70
71     public String displayInfo(boolean showDetails) {
72         String info = super.displayInfo();
73         if (showDetails) {
74             info += "\nNumber of Doors: " + numberOfDoors + "\nIn
75         }
76         return info;
77     }
78 }

3 public class code {
44 }
45
46 public static class Car extends Vehicle {
47
48     private int numberOfDoors; 3 usages
49
50     private int trunkCapacity; 3 usages
51
52     private int fuelCapacity=9; 1 usage
53
54     public Car(String brand, int speed, int fuelCapacity, int num
55         super(brand, speed, fuelCapacity);
56         this.numberOfDoors = numberOfDoors;
57         this.trunkCapacity = trunkCapacity;
58     }
59
60
61     @Override
62     public String move() {
63         return "The car is driving...";
64     }
65
66     public int getFuelCapacity() {
67         return fuelCapacity;
68     }
69
70     public int calculateTotalStorage() {
71         return trunkCapacity + (numberOfDoors * 10);
72     }
73 }
```

IOD: Overriding Method Deletion

```
3 public class code {
44 }
45
46 public static class Car extends Vehicle {
47
48     private int numberOfDoors; 3 usages
49
50     private int trunkCapacity; 3 usages
51
52     private int fuelCapacity = 9; 1 usage
53
54     public Car(String brand, int speed, int fuelCapacity, int number
55         super(brand, speed, fuelCapacity);
56         this.numberOfDoors = numberOfDoors;
57         this.trunkCapacity = trunkCapacity;
58     }
59
60     public int getFuelCapacity() { return fuelCapacity; }
61
62     public int calculateTotalStorage() { return trunkCapacity + (num
63
64     public String displayInfo(boolean showDetails) {
65         String info = super.displayInfo();
66         if (showDetails) {
67             info += "\nNumber of Doors: " + numberOfDoors + "\nTrunk
68         }
69         return info;
70     }
71
72 }
73
74
75
76
77 public static class Bike extends Vehicle { 1 inheritor
```

```
3 public class code {
45 public static class Vehicle {
46
47     public static class Car extends Vehicle {
48
49         private int numberOfDoors; 3 usages
50
51         private int trunkCapacity; 3 usages
52
53         private int fuelCapacity=9; 1 usage
54
55         public Car(String brand, int speed, int fuelCapacity, int numbe
56             super(brand, speed, fuelCapacity);
57             this.numberOfDoors = numberOfDoors;
58             this.trunkCapacity = trunkCapacity;
59         }
60
61         @Override
62         public String move() {
63             return "The car is driving...";
64         }
65
66         public int getFuelCapacity() {
67             return fuelCapacity;
68         }
69
70         public int calculateTotalStorage() {
71             return trunkCapacity + (numberOfDoors * 10);
72         }
73
74         public String displayInfo(boolean showDetails) {
```

IOP: Overridden Method Calling Position Change

```
3 public class code {
46     public static class Car extends Vehicle {
73         public String displayInfo(boolean showDetails) {
79             }
80     }
81
82     public static class Bike extends Vehicle { 1 inheritor
83
84         private boolean hasBasket; 4 usages
85
86         private int basketCapacity; 3 usages
87
88         public Bike(String brand, int speed, int fuelCapacity, boolean hasBasket,
89             this.hasBasket = hasBasket;
90             this.basketCapacity = basketCapacity;
91             super(brand, speed, fuelCapacity);
92     }
93
94     @Override
95     public String move() { return "The bike is pedaling..."; }
96
97     public int calculateStorage() { return hasBasket ? basketCapacity : 0; }
98
99     public int getFuelCapacity() { return fuelCapacity; }
100
101     public int calculateStorage(int additionalItems) {
102         int totalCapacity = calculateStorage();
103         return totalCapacity + additionalItems * 2;
104     }
105 }
106
107 @Override
```

```
3 public class code {
46     public static class Car extends Vehicle {
73         public String displayInfo(boolean showDetails) {
76             info += "\nNumber of Doors: " + numberOfDoors;
77         }
78         return info;
79     }
80
81     public static class Bike extends Vehicle { 1 inheritor
82
83         private boolean hasBasket; 4 usages
84
85         private int basketCapacity; 3 usages
86
87         public Bike(String brand, int speed, int fuelCapacity, boolean hasBasket,
88             super(brand, speed, fuelCapacity);
89             this.hasBasket = hasBasket;
90             this.basketCapacity = basketCapacity;
91         }
92
93     @Override
94     public String move() {
95         return "The bike is pedaling...";
96     }
97
98     public int calculateStorage() {
99         return hasBasket ? basketCapacity : 0;
100     }
101
102     public int getFuelCapacity() {
```

IOR: Overridden Method Rename

```
public class code {  
    public static class Vehicle { 3 inheritors  
  
        public String pDisplayInfo() { 2 usages  
            return "Brand: " + brand + "\n" + "Speed: " + speed + " km/h\n";  
        }  
  
        public String pMove() { return "The vehicle is moving..."; }  
  
        public String displayVehicleCount() { return "Total Vehicles: " + vehicleCount; }  
    }  
  
    public static class Car extends Vehicle {  
  
        private int numberOfDoors; 3 usages  
  
        private int trunkCapacity; 3 usages  
  
        private int fuelCapacity = 9; 1 usage  
  
        public Car(String brand, int speed, int fuelCapacity, int numberOfDoors) {  
            super(brand, speed, fuelCapacity);  
            this.numberOfDoors = numberOfDoors;  
            this.trunkCapacity = trunkCapacity;  
        }  
  
        @Override no usages  
        public String pMove() { return "The car is driving..."; }  
  
        public int pGetFuelCapacity() { return fuelCapacity; }  
    }  
}
```

```
public class code {  
    public static class Vehicle { 3 inheritors  
  
        public String displayInfo() { 1 override  
            return "Brand: " + brand + "\n" + "Speed: " + speed + " km/h\n";  
        }  
  
        public String move() { 2 overrides  
            return "The vehicle is moving...";  
        }  
  
        public String displayVehicleCount() {  
            return "Total Vehicles: " + vehicleCount;  
        }  
    }  
  
    public static class Car extends Vehicle {  
  
        private int numberOfDoors; 3 usages  
  
        private int trunkCapacity; 3 usages  
  
        private int fuelCapacity=9; 1 usage  
  
        public Car(String brand, int speed, int fuelCapacity, int numberOfDoors) {  
            super(brand, speed, fuelCapacity);  
            this.numberOfDoors = numberOfDoors;  
            this.trunkCapacity = trunkCapacity;  
        }  
  
        @Override  
        public String move() { return "The car is driving..."; }  
    }  
}
```

ISI: Super Keyword Insertion

```
public class code {
    public static class Car extends Vehicle {
        private int fuelCapacity = 9; no usages

        public Car(String brand, int speed, int fuelCapacity, int numberOfDoors,
            super(brand, speed, fuelCapacity);
            this.numberOfDoors = numberOfDoors;
            this.trunkCapacity = trunkCapacity;
        }

        @Override
        public String move() { return "The car is driving..."; }

        public int getFuelCapacity() { return super.fuelCapacity; }

        public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }

        public String displayInfo(boolean showDetails) {
            String info = super.displayInfo();
            if (showDetails) {
                info += "\nNumber of Doors: " + numberOfDoors + "\nTrunk Capacity: " + trunkCapacity;
            }
            return info;
        }
    }

    public static class Bike extends Vehicle { 1 inheritor
        private boolean hasBasket; 4 usages
        private int hasBasketCapacity; 4 usages
    }
}
```

```
public class code {
    public static class Car extends Vehicle {
        public Car(String brand, int speed, int fuelCapacity, int numberOfDoors,
            super(brand, speed, fuelCapacity);
            this.numberOfDoors = numberOfDoors;
            this.trunkCapacity = trunkCapacity;
        }

        @Override
        public String move() {
            return "The car is driving...";
        }

        public int getFuelCapacity() {
            return fuelCapacity;
        }

        public int calculateTotalStorage() {
            return trunkCapacity + (numberOfDoors * 10);
        }

        public String displayInfo(boolean showDetails) {
            String info = super.displayInfo();
            if (showDetails) {
                info += "\nNumber of Doors: " + numberOfDoors + "\nTrunk Capacity: " + trunkCapacity;
            }
            return info;
        }
    }

    public static class Bike extends Vehicle { 1 inheritor
        private boolean hasBasket; 4 usages
        private int hasBasketCapacity; 4 usages
    }
}
```

ISD: Super Keyword Deletion

```
64 public int getFuelCapacity() { return fuelCapacity; }
65
66 public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }
67
68 public String displayInfo(boolean showDetails) {
69     String info = super.displayInfo();
70     if (showDetails) {
71         info += "\nNumber of Doors: " + numberOfDoors + "\nTrunk Capacity: " + trunkCapacity + " liters";
72     }
73     return info;
74 }
75
76 public static class Bike extends Vehicle {
77     private boolean hasBasket;
```

```
59 @Override
60 public String move() { return "The car is driving..."; }
61
62 public int getFuelCapacity() { return fuelCapacity; }
63
64 public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }
65
66 public String displayInfo(boolean showDetails) {
67     String info = displayInfo();
68     if (showDetails) {
69         info += "\nNumber of Doors: " + numberOfDoors + "\nTrunk Capacity: " + trunkCapacity + " liters";
70     }
71     return info;
72 }
73
74 public static class Bike extends Vehicle {
```

IPC : Explicit Parent Constructor Deletion

```
49 private int trunkCapacity;  
50  
51  
52 private int fuelCapacity=9;  
53  
54 public Car(String brand, int speed, int fuelCapacity, int numberOfDoors, int trunkCapacity) {  
55     Super(brand, speed, fuelCapacity);  
56     this.numberOfDoors = numberOfDoors;  
57     this.trunkCapacity = trunkCapacity;  
58 }  
59  
60 @Override  
61 public String move() { return "The car is driving..."; }  
62  
63  
64 public int getFuelCapacity() { return fuelCapacity; }  
65  
66  
67 public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }  
68  
69  
70
```

```
49 private int trunkCapacity;  
50  
51  
52 private int fuelCapacity = 9;  
53  
54 public Car(String brand, int speed, int fuelCapacity, int numberOfDoors, int trunkCapacity) {  
55     this.numberOfDoors = numberOfDoors;  
56     this.trunkCapacity = trunkCapacity;  
57 }  
58  
59 @Override  
60 public String move() { return "The car is driving..."; }  
61  
62  
63 public int getFuelCapacity() { return fuelCapacity; }  
64  
65 public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }  
66  
67  
68 public String displayInfo(boolean showDetails) {  
69  
70  
71  
72
```

PNC: new Method Call With Child Class Type

```
140
141
142 Vehicle vehicle = (Vehicle) new Vehicle( brand: "sd", speed: 12, fuelCapacity: 25);
143 Vehicle v2 = new Vehicle( brand: "fffff", speed: 20, fuelCapacity: 20);
144 Bike b = new Bike( brand: "lllll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);
145
146
147
148
149
150
151
152
153 Vehicle vehicle = (Vehicle) new Bike( brand: "sd", speed: 12, fuelCapacity: 25);
154 Vehicle v2 = new Bike( brand: "fffff", speed: 20, fuelCapacity: 20);
155 Bike b = new UniCycle( brand: "lllll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);
156
157
```


PMD: Member Variable Declaration with Parent Class Type

```
133 public class Main{  
134     public static void main(String[] args) {  
135         Vehicle v = new Car( brand: "ghl", speed: 2, fuelCapacity: 2, numberOfDoors: 5, trunkCapacity: 6);  
136         Vehicle v1 = new Bike( brand: "ll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);  
137  
137     public static void main(String[] args) {  
138         Car v = new Car( brand: "ghl", speed: 2, fuelCapacity: 2, numberOfDoors: 5, trunkCapacity: 6);  
139         Bike v1 = new Bike( brand: "ll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);  
140  
141     }
```

PPD: Parameter Variable Declaration with Child Class Type

```
167
168  public static boolean isNull(Vehicle v) { return v == null; }
171
172
173  public static boolean equals(Bike b) { return b==null; }
176  }
177
159
160  public static boolean isNull(Bike v) { return v == null; }
163
164  public static boolean equals(UniCycle b) { return b == null; }
167  }
168
```

PCI: Type Cast Operator Insertion

```
Vehicle v = new Car( brand: "ghl", speed: 2, fuelCapacity: 2, numberOfDoors: 5, trunkCapacity: 6);  
Vehicle v1 = new Bike( brand: "ll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);
```

```
String y = v.displayVehicleCount();  
Vehicle v4=v ;
```

```
Vehicle v = new Car( brand: "ghl", speed: 2, fuelCapacity: 2, numberOfDoors: 5, trunkCapacity: 6);  
Vehicle v1 = new Bike( brand: "ll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);  
String y = (Vehicle) v.displayVehicleCount();  
Vehicle v4 = (Vehicle) v;
```

PCD: Type Cast Operator Deletion

```
135 Vehicle v = new Car( brand: "ghl", speed: 2, fuelCapacity: 2, numberOfDoors: 5, trunkCapacity: 6);  
136 Vehicle v1 = new Bike( brand: "ll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);  
137  
138 Vehicle vehicle = (Vehicle) new Vehicle( brand: "sd", speed: 12, fuelCapacity: 25);  
139  
140  
141 Vehicle v4=(Vehicle) v ;
```

```
Vehicle v = new Car( brand: "ghl", speed: 2, fuelCapacity: 2, numberOfDoors: 5, trunkCapacity: 6);  
Vehicle v1 = new Bike( brand: "ll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);  
Vehicle vehicle = new Vehicle( brand: "sd", speed: 12, fuelCapacity: 25);  
Vehicle v4 = v;
```

• PCC: Cast Type Change •

```
159  
160 Vehicle v34 = (Bike) v8;  
161 Vehicle v2234566 = (Car) v2;  
162  
155  
156 Vehicle v34 = (Tank) v8;  
157 Vehicle v2234566 = (Bike) v2;  
158
```

PRV: Reference Assignment with Other Compatible Type

```
Bike b = new Bike( brand: "llllll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);  
Car myCar = new Car( brand: "ll", speed: 120, fuelCapacity: 24, numberOfDoors: 5, trunkCapacity: 6);
```

```
Vehicle vehicle1;  
vehicle1 = myCar;
```

```
Bike b = new Bike( brand: "llllll", speed: 20, fuelCapacity: 20, hasBasket: true, basketCapacity: 20);  
Car myCar = new Car( brand: "ll", speed: 120, fuelCapacity: 24, numberOfDoors: 5, trunkCapacity: 6);
```

```
Vehicle vehicle1;  
vehicle1 = b;
```

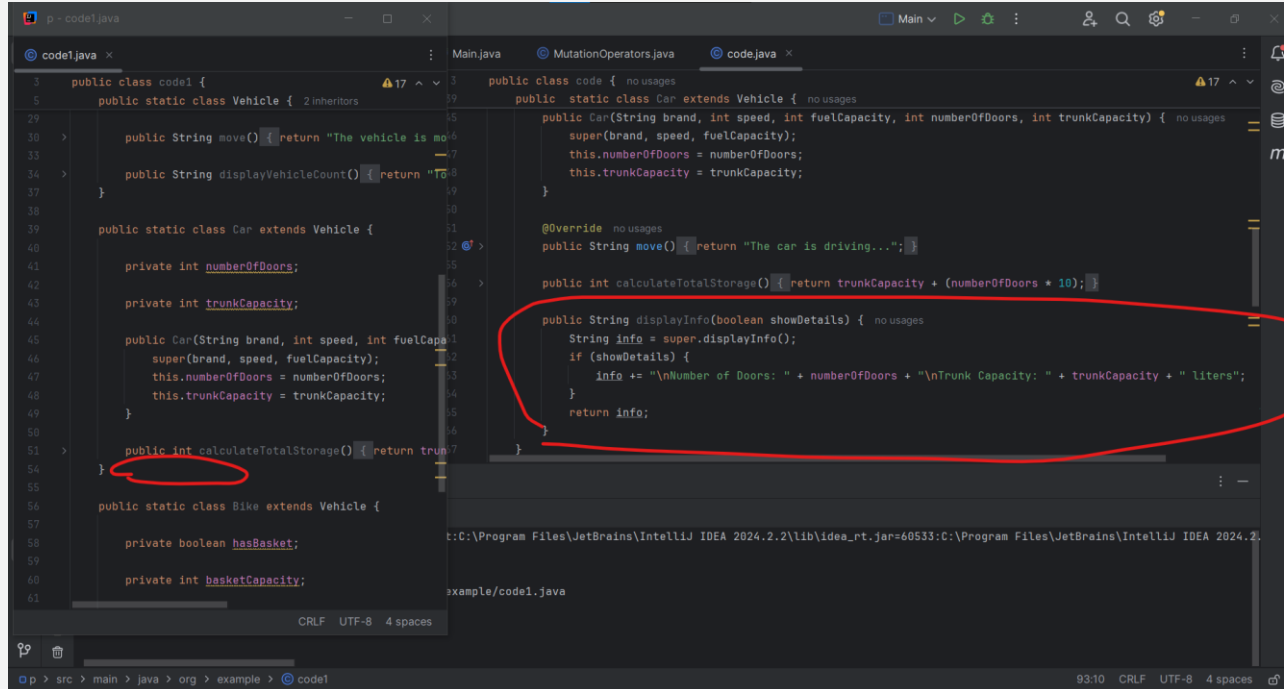
OMR: Overloading Method Contents Replace

The screenshot shows two Java files in an IDE. The left file, `code1.java`, contains a `Vehicle` class with a `displayInfo()` method. The right file, `MutationOperators.java`, contains a `Car` class that extends `Vehicle` and also has a `displayInfo()` method. Both methods are circled in red to show how the content of the overloaded method is replaced.

```
public class code1 {  
    public static class Bike extends Vehicle {  
        public Bike(String brand, int speed, int fuelCapacity) {  
            super(brand, speed, fuelCapacity);  
            this.hasBasket = hasBasket;  
            this.basketCapacity = basketCapacity;  
        }  
        @Override  
        public String move() {  
            return "The car is driving...";  
        }  
        public int calculateStorage() {  
            int totalCapacity = calculateStorage();  
            return totalCapacity + additionalItems * 2;  
        }  
        public int calculateStorage(int additionalItems) {  
            return hasBasket ? basketCapacity : 0;  
        }  
        public String displayInfo() {  
            String info = super.displayInfo();  
            if (showDetails) {  
                info += "\nNumber of Doors: " + numberOfDoors;  
            }  
            return info;  
        }  
    }  
}
```

```
public class code {  
    public static class Car extends Vehicle {  
        public Car(String brand, int speed, int fuelCapacity, int numberOfDoors, int trunkCapacity) {  
            super(brand, speed, fuelCapacity);  
            this.numberOfDoors = numberOfDoors;  
            this.trunkCapacity = trunkCapacity;  
        }  
        @Override  
        public String move() { return "The car is driving..."; }  
        public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }  
        public String displayInfo(boolean showDetails) {  
            String info = super.displayInfo();  
            if (showDetails) {  
                info += "\nNumber of Doors: " + numberOfDoors + "\nTrunk Capacity: " + trunkCapacity + " liters";  
            }  
            return info;  
        }  
    }  
}
```

OMD: Overloading Method Deletion



```
code1.java
3 public class code1 {
5     public static class Vehicle { 2 inheritors
29         public String move() { return "The vehicle is moving..."; }
33         public String displayVehicleCount() { return "Total vehicles: " + Vehicle.getCount(); }
37     }
38     public static class Car extends Vehicle {
40         private int numberOfDoors;
42         private int trunkCapacity;
44         public Car(String brand, int speed, int fuelCapacity, int numberOfDoors, int trunkCapacity) {
46             super(brand, speed, fuelCapacity);
47             this.numberOfDoors = numberOfDoors;
48             this.trunkCapacity = trunkCapacity;
49         }
51         public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }
54     }
55     public static class Bike extends Vehicle {
57         private boolean hasBasket;
59         private int basketCapacity;
61     }

```

```
Main.java
3 public class Main {
5     public static void main(String[] args) {
7         Car car = new Car("Toyota", 100, 50, 4, 10);
9         car.move();
11        car.displayVehicleCount();
13        car.calculateTotalStorage();
15        car.displayInfo(true);
17    }

```

```
MutationOperators.java
3 public class MutationOperators {
5     public static void mutate(Car car) {
7         car.numberOfDoors = car.numberOfDoors * 2;
9         car.trunkCapacity = car.trunkCapacity * 2;
11    }

```

```
code.java
3 public class code { no usages
5     public static class Car extends Vehicle { no usages
15         public Car(String brand, int speed, int fuelCapacity, int numberOfDoors, int trunkCapacity) { no usages
17             super(brand, speed, fuelCapacity);
18             this.numberOfDoors = numberOfDoors;
19             this.trunkCapacity = trunkCapacity;
20         }
21     }
22     @Override no usages
23     public String move() { return "The car is driving..."; }
24     public int calculateTotalStorage() { return trunkCapacity + (numberOfDoors * 10); }
25     public String displayInfo(boolean showDetails) { no usages
26         String info = super.displayInfo();
27         if (showDetails) {
28             info += "\nNumber of Doors: " + numberOfDoors + "\nTrunk Capacity: " + trunkCapacity + " liters";
29         }
30         return info;
31     }
32 }

```


OAC: Arguments of Overloading Method Call Change

The screenshot displays the IntelliJ IDEA IDE with two tabs open: `code1.java` and `code.java`. The `code1.java` tab shows the original code, and the `code.java` tab shows the code after applying the OAC mutation. The mutation changes the argument `additionalItems` to `null` in the `calculateStorage` method call.

```
public class code1 {  
    public static class Bike extends Vehicle {  
        @ate int basketCapacity;  
        lic Bike(String brand, int speed, int fuelCapacity, boolean hasBasket, int basketCapacity) {  
            super(brand, speed, fuelCapacity);  
            this.hasBasket = hasBasket;  
            this.basketCapacity = basketCapacity;  
        }  
        @Override  
        lic String move() { return "The bike is pedaling..."; }  
        lic int calculateStorage() { return hasBasket ? basketCapacity : 0; }  
        lic int calculateStorage(int additionalItems) {  
            int totalCapacity = calculateStorage(additionalItems, null);  
            return totalCapacity + additionalItems * 2;  
        }  
        lic String displayInfo() {  
            return "Bike: " + brand + ", Speed: " + speed + ", Fuel Capacity: " + fuelCapacity + ", Has Basket: " + hasBasket + ", Basket Capacity: " + basketCapacity;  
        }  
    }  
}
```

The `code.java` tab shows the mutated code, where the `calculateStorage` method call is changed to `calculateStorage()` with no arguments, as indicated by the red circle around the method call.

```
public class code {  
    public static class Bike extends Vehicle {  
        private int basketCapacity;  
        public Bike(String brand, int speed, int fuelCapacity, boolean hasBasket, int basketCapacity) {  
            super(brand, speed, fuelCapacity);  
            this.hasBasket = hasBasket;  
            this.basketCapacity = basketCapacity;  
        }  
        @Override  
        public String move() { return "The bike is pedaling..."; }  
        public int calculateStorage() { return hasBasket ? basketCapacity : 0; }  
        public int calculateStorage(int additionalItems) {  
            int totalCapacity = calculateStorage();  
            return totalCapacity + additionalItems * 2;  
        }  
        public String displayInfo() {  
            return "Bike: " + brand + ", Speed: " + speed + ", Fuel Capacity: " + fuelCapacity + ", Has Basket: " + hasBasket + ", Basket Capacity: " + basketCapacity;  
        }  
    }  
}
```

The bottom panel shows the OAC mutation applied and the code saved to `src/main/java/org/example/code1.java`. The process finished with exit code 0.

JTI: this Keyword Insertion

```
public class code1 {  
    public static class Bike extends Vehicle {  
        public String move() { return "The bike is pedaling..."; }  
  
        public int calculateStorage() { return this.hasBasket; }  
  
        public int calculateStorage(int additionalItems) {  
            int totalCapacity = this.calculateStorage();  
            return this.totalCapacity + this.additionalItems;  
        }  
  
        public String displayInfo() {  
            String info = super.displayInfo();  
            this.info += "\nHas Basket: " + (this.hasBasket ? "Yes" : "No");  
            if (this.hasBasket) {  
                this.info += "\nBasket Capacity: " + this.basketCapacity;  
            }  
            return this.info;  
        }  
    }  
}
```

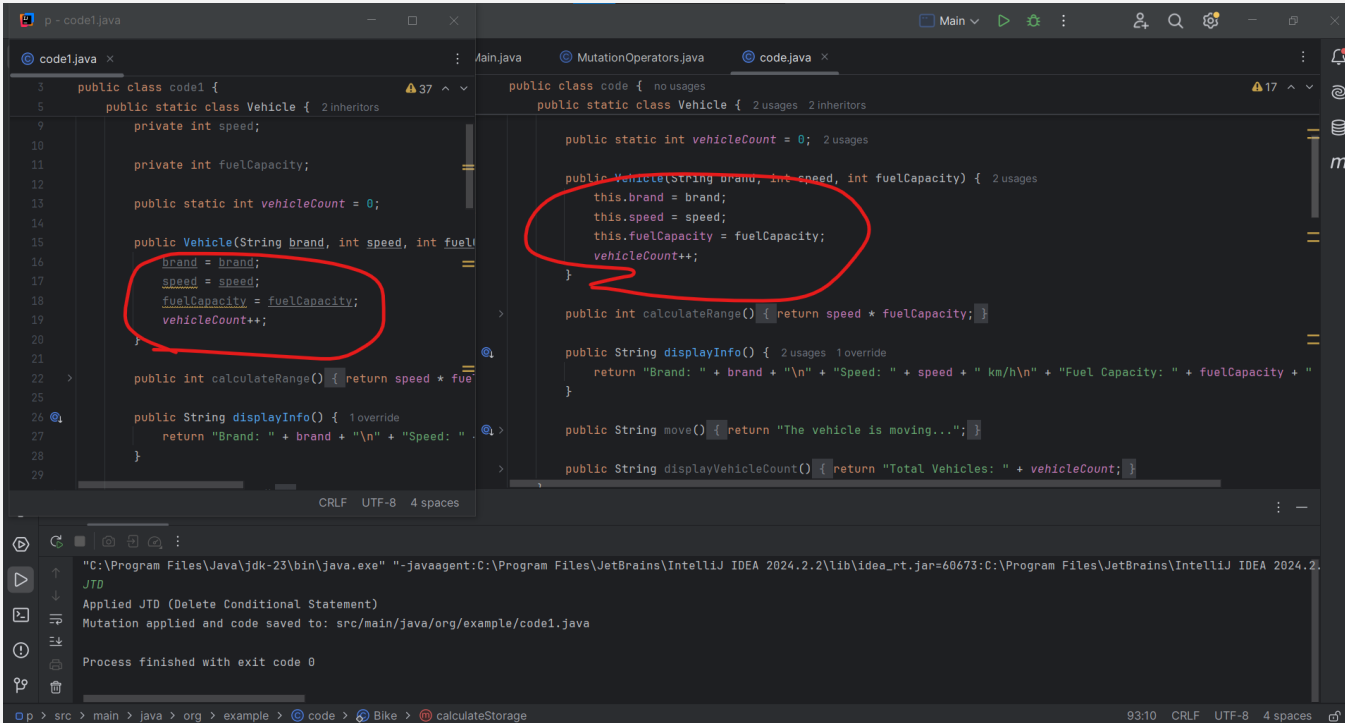
```
public class code {  
    public static class Bike extends Vehicle {  
        public int calculateStorage(int additionalItems) {  
            int totalCapacity = calculateStorage();  
            return totalCapacity + additionalItems * 2;  
        }  
  
        public String displayInfo() {  
            String info = super.displayInfo();  
            info += "\nHas Basket: " + (hasBasket ? "Yes" : "No");  
            if (hasBasket) {  
                info += "\nBasket Capacity: " + basketCapacity + " liters";  
            }  
            return info;  
        }  
    }  
}
```

CRLF UTF-8 4 spaces

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=60642:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin" 60642
JTI
Applied JTI (Change Conditional Statement)
Mutation applied and code saved to: src/main/java/org/example/code1.java
Process finished with exit code 0

p > src > main > java > org > example > code > Bike > calculateStorage 93:10 CRLF UTF-8 4 spaces

JTD: this Keyword Deletion



```
code1.java
3 public class code1 {
5     public static class Vehicle { 2 inheritors
9         private int speed;
10
11         private int fuelCapacity;
12
13         public static int vehicleCount = 0;
14
15         public Vehicle(String brand, int speed, int fuelCapacity) {
16             brand = brand;
17             speed = speed;
18             fuelCapacity = fuelCapacity;
19             vehicleCount++;
20         }
21
22         public int calculateRange() { return speed * fuelCapacity; }
23
24         public String displayInfo() { 1 override
25             return "Brand: " + brand + "\n" + "Speed: " + speed + " km/h\n" + "Fuel Capacity: " + fuelCapacity + "
26         }
27     }
28 }
29
```

```
code.java
3 public class code { no usages
5     public static class Vehicle { 2 usages 2 inheritors
9         public static int vehicleCount = 0; 2 usages
10
11         public Vehicle(String brand, int speed, int fuelCapacity) { 2 usages
12             brand = brand;
13             speed = speed;
14             fuelCapacity = fuelCapacity;
15             vehicleCount++;
16         }
17
18         public int calculateRange() { return speed * fuelCapacity; }
19
20         public String displayInfo() { 2 usages 1 override
21             return "Brand: " + brand + "\n" + "Speed: " + speed + " km/h\n" + "Fuel Capacity: " + fuelCapacity + "
22         }
23
24         public String move() { return "The vehicle is moving..."; }
25
26         public String displayVehicleCount() { return "Total Vehicles: " + vehicleCount; }
27     }
28 }
29
```

Applied JTD (Delete Conditional Statement)
Mutation applied and code saved to: src/main/java/org/example/code1.java
Process finished with exit code 0

JSL: Static Modifier Insertion

```
package org.example;

public class code1 {

    public static class Vehicle { 2 inheritors
        protected static String brand;
        private static int speed;

        private static int fuelCapacity;

        public static int vehicleCount = 0;

        public Vehicle(String brand, int speed, int fuelCapacity) {
            this.brand = brand;
            this.speed = speed;
            this.fuelCapacity = fuelCapacity;
            vehicleCount++;
        }

        public int calculateRange() { return speed + fuelCapacity; }

        public String displayInfo() { 1 override
            return "Brand: " + brand + "\n" + "Speed: " + speed;
        }

        public String move() { return "The vehicle is moving"; }

        public String getTotalVehicleCount() { return "Total Vehicle Count: " + vehicleCount; }
    }
}

package org.example;

public class code { no usages

    public static class Vehicle { 2 usages 2 inheritors
        protected String brand; 2 usages
        private int speed; 3 usages

        private int fuelCapacity; 3 usages

        public static int vehicleCount = 0; 2 usages

        public Vehicle(String brand, int speed, int fuelCapacity) { 2 usages
            this.brand = brand;
            this.speed = speed;
            this.fuelCapacity = fuelCapacity;
            vehicleCount++;
        }
    }
}
```

93:10 CRLF UTF-8 4 spaces

JSD: Static Modifier Deletion

The screenshot displays the IntelliJ IDEA IDE with two code editors side-by-side, illustrating the JSD (Java Static Deletion) mutation operator. The left editor shows the original code, and the right editor shows the code after applying the 'Delete static... Statement' mutation.

Original Code (Left Editor):

```
1 package org.example;
2
3 class code {
4
5     class Vehicle { 2 inheritors
6
7         String brand;
8
9         int speed;
10
11         int fuelCapacity;
12
13         int vehicleCount = 0;
14
15         public Vehicle(String brand, int speed, int fuelCapacity) {
16             this.brand = brand;
17             this.speed = speed;
18             this.fuelCapacity = fuelCapacity;
19             vehicleCount++;
20         }
21     }
22 }
```

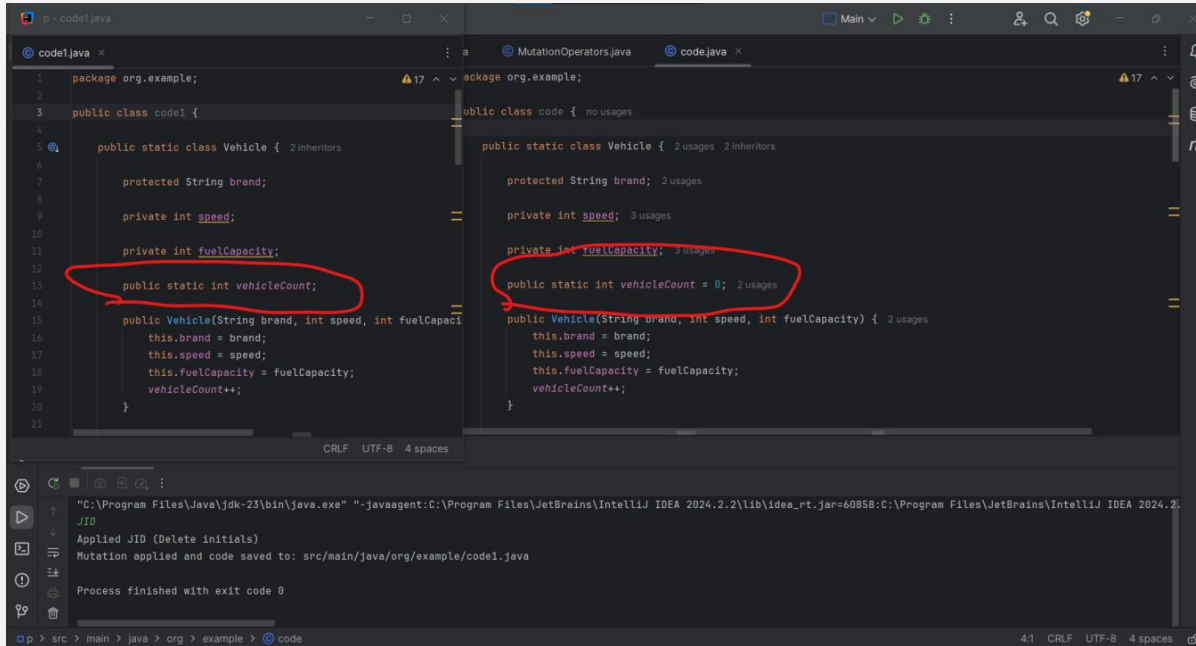
Mutated Code (Right Editor):

```
1 package org.example;
2
3 class code { no usages
4
5     public static class Vehicle { 2 usages 2 inheritors
6
7         protected String brand; 2 usages
8
9         private int speed; 3 usages
10
11         private int fuelCapacity; 3 usages
12
13         public static int vehicleCount = 0; 2 usages
14
15         public Vehicle(String brand, int speed, int fuelCapacity) { 2 usages
16             this.brand = brand;
17             this.speed = speed;
18             this.fuelCapacity = fuelCapacity;
19             vehicleCount++;
20         }
21     }
22 }
```

Console Output (Bottom Panel):

```
"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=60822:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin" -Didea.config.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\config -Didea.system.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin -Didea.vendor.name=IntelliJ -jar C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin\idea_rt.jar 60822
JSD
Applied JSD (Delete static... Statement)
Mutation applied and code saved to: src/main/java/org/example/code1.java
Process finished with exit code 0
```

JID: Member Variable Initialization Deletion



JDC: Java-supported Default Constructor Deletion

```
package org.example;

public class code1 {

    public static class Vehicle { 2 inheritors

        protected String brand;

        private int speed;

        private int fuelCapacity;

        public static int vehicleCount;

        public Vehicle(String brand, int speed, int fuelCapacity) {

            this.brand = brand;
            this.speed = speed;
            this.fuelCapacity = fuelCapacity;
            vehicleCount++;
        }
    }
}

package org.example;

public class code { no usages

    public static class Vehicle { 2 usages 2 inheritors

        protected String brand; 2 usages

        private int speed; 3 usages

        private int fuelCapacity; 3 usages

        public static int vehicleCount = 0; 2 usages

        public Vehicle(String brand, int speed, int fuelCapacity) { 2 usages

            this.brand = brand;
            this.speed = speed;
            this.fuelCapacity = fuelCapacity;
            vehicleCount++;
        }
    }
}
```

"C:\Program Files\Java\jdk-23\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\lib\idea_rt.jar=60858:C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin" -Didea.config.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\conf -Didea.copyright.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\copyright -Didea.home.path=C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin -Didea.platform.prefix=JDK -Didea.vendor.id=JetBrains -Didea.version=2024.2.2 -jar C:\Program Files\JetBrains\IntelliJ IDEA 2024.2.2\bin\idea_rt.jar 60858

JDC

Applied JDC (Delete initials)

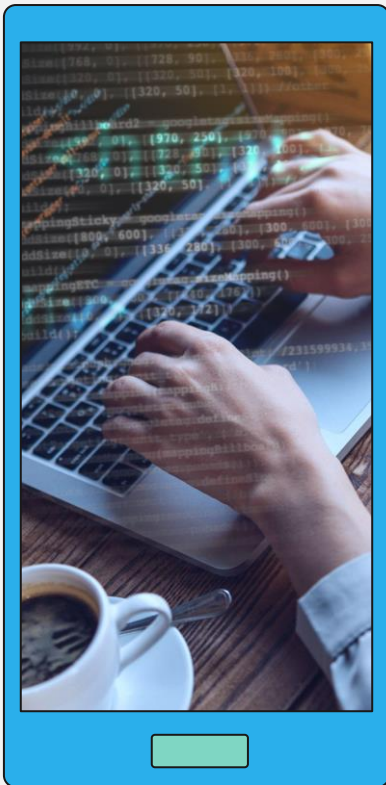
Mutation applied and code saved to: src/main/java/org/example/code1.java

Process finished with exit code 0

p > src > main > java > org > example > code

03 Mutation Score





انجام تست کلی:



- تعداد کل Mutant های تست: 25
- تعداد Mutant های kill شده در تست: 16
(IOP, IOR, IPC, JDC, JTI, OAC, OMR, PCI,
PNC, IHD, IOD, ISD, ISI, JID, JSI, JTD)
- تعداد Equivalent ها: 0



Result:

$$\text{Mutation Score} = \frac{16}{25} \times 100 = 64\%$$