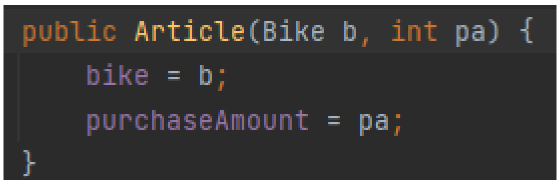
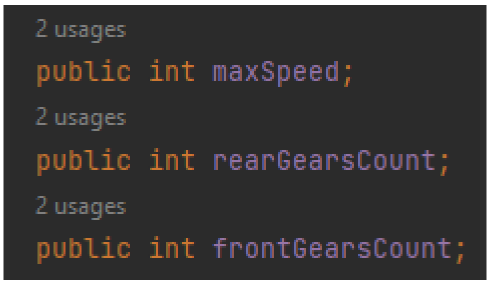
```
Zweifelhafte Namen -> Refactoring Rename
```



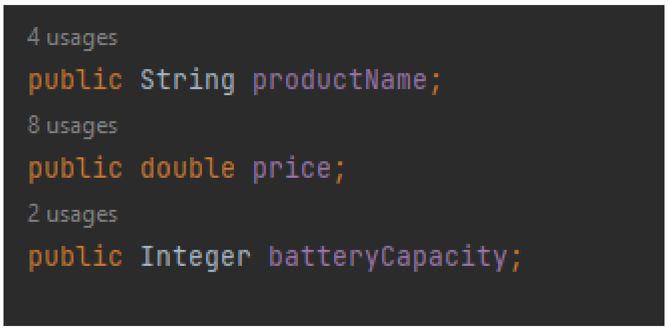
```
public Article(Bike b, int pa) {
    bike = b;
    purchaseAmount = pa;
}
```

Duplizierter Code in allen erbenden Klassen -> Refactoring Pull Up



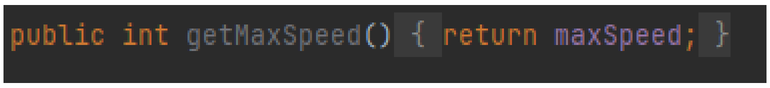
```
2 usages
public int maxSpeed;
2 usages
public int rearGearsCount;
2 usages
public int frontGearsCount;
```

Fehlende Datenkapselung -> Refactoring Encapsulate Field



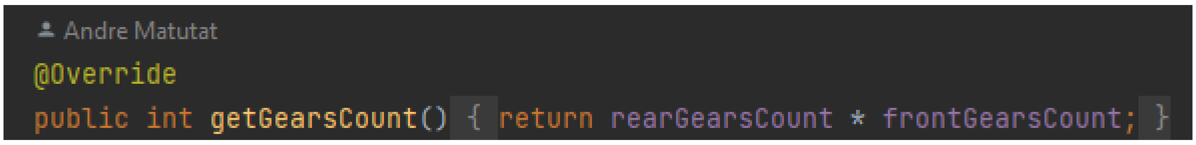
```
4 usages
public String productName;
8 usages
public double price;
2 usages
public Integer batteryCapacity;
```

duplizierter Code -> Refactoring Pull Up



```
public int getMaxSpeed() { return maxSpeed; }
```

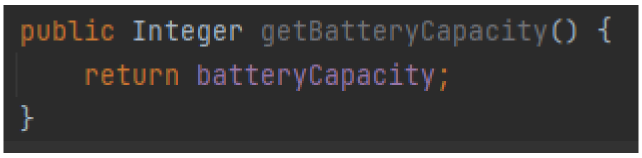
Doppelter Code -> Refactoring Extract Method



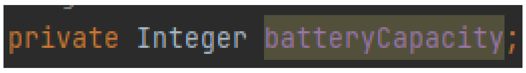
```
Andre Matutat
@Override
public int getGearsCount() { return rearGearsCount * frontGearsCount; }
```

Integer zu int und final

Methode und Attribute werden nur in EBike Klasse benutzt -> Refactoring Pull Down

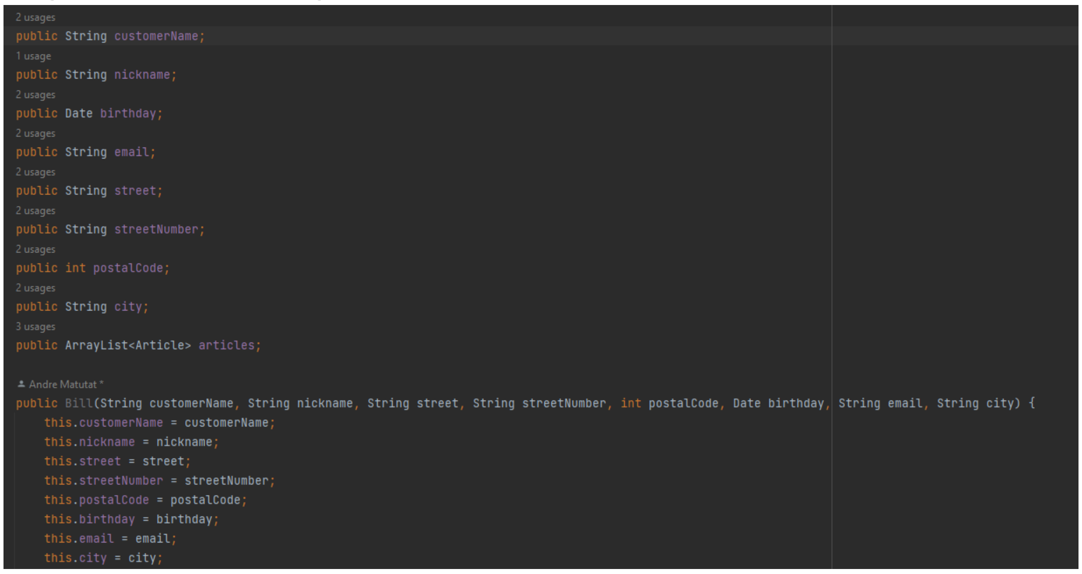


```java
public Integer getBatteryCapacity() {
    return batteryCapacity;
}
```



```java
private Integer batteryCapacity;
```

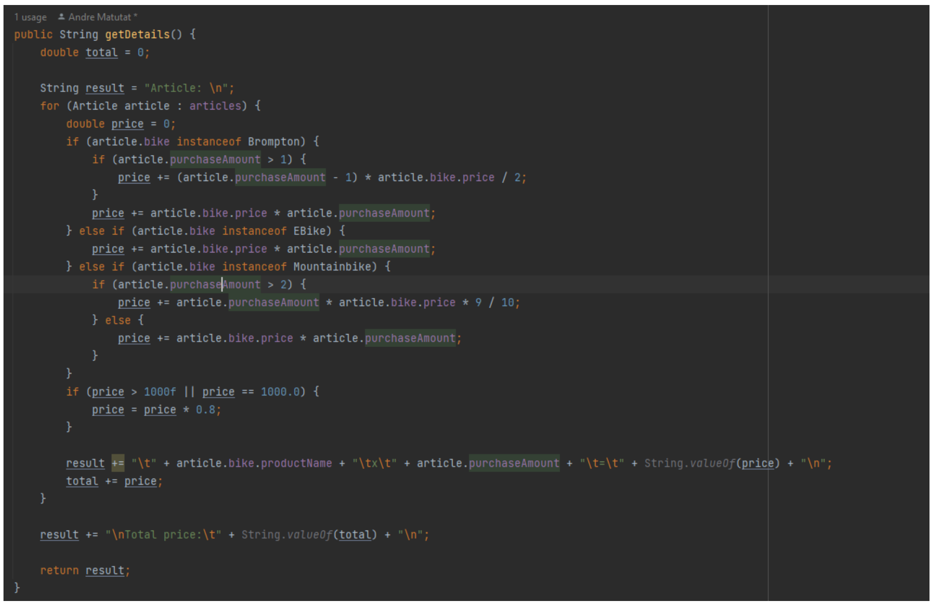
Langer Code -> Refactoring Extract Class



```java
2 usages
public String customerName;
1 usage
public String nickname;
2 usages
public Date birthday;
2 usages
public String email;
2 usages
public String street;
2 usages
public String streetNumber;
2 usages
public int postalCode;
2 usages
public String city;
3 usages
public ArrayList<Article> articles;

Andre Matutat *
public Bill(String customerName, String nickname, String street, String streetNumber, int postalCode, Date birthday, String email, String city) {
    this.customerName = customerName;
    this.nickname = nickname;
    this.street = street;
    this.streetNumber = streetNumber;
    this.postalCode = postalCode;
    this.birthday = birthday;
    this.email = email;
    this.city = city;
```

Methode zu groß, zu langer Code, tief verschachtelt -> Refactoring Extract Method



```java
1 usage   Andre Matutat *
public String getDetails() {
    double total = 0;

    String result = "Article: \n";
    for (Article article : articles) {
        double price = 0;
        if (article.bike instanceof Brompton) {
            if (article.purchaseAmount > 1) {
                price += (article.purchaseAmount - 1) * article.bike.price / 2;
            }
            price += article.bike.price * article.purchaseAmount;
        } else if (article.bike instanceof EBike) {
            price += article.bike.price * article.purchaseAmount;
        } else if (article.bike instanceof Mountainbike) {
            if (article.purchaseAmount > 2) {
                price += article.purchaseAmount * article.bike.price * 9 / 10;
            } else {
                price += article.bike.price * article.purchaseAmount;
            }
        }
        if (price > 1000f || price == 1000.0) {
            price = price * 0.8;
        }

        result += "\t" + article.bike.productName + "\tx\t" + article.purchaseAmount + "\t=\t" + String.valueOf(price) + "\n";
        total += price;
    }

    result += "\nTotal price:\t" + String.valueOf(total) + "\n";

    return result;
}
```