

Code Reengineering Document Project

Dosen : Kornelius Irfandhi S.Kom., M.Tl.

Kelas : LF01

Anggota kelompok :

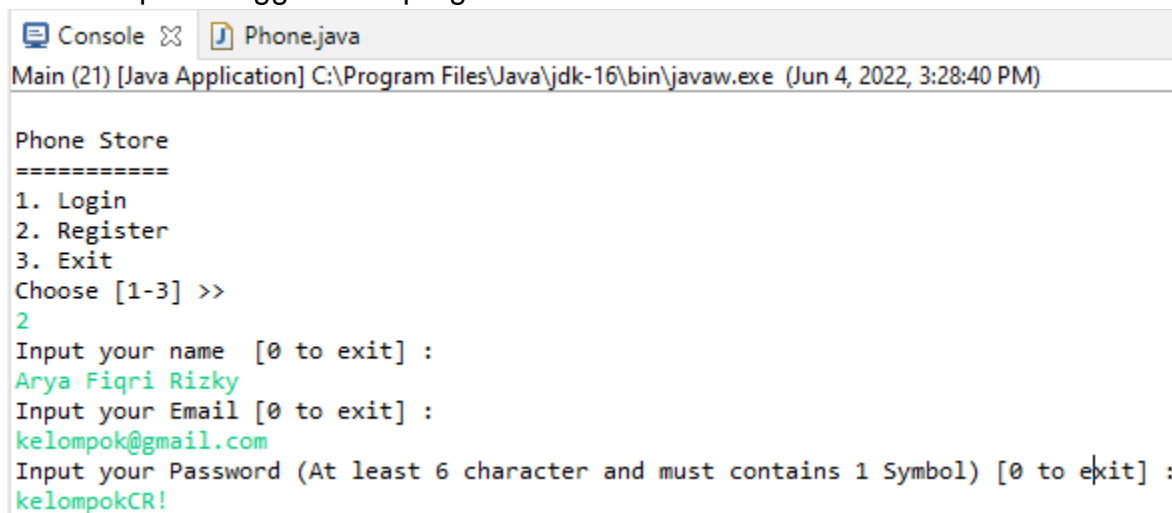
- Arya Putra Kartiwa - 2440040343
- Muhammad Fiqri Febriansyah - 2440084995
- Rizky Hertama - 2440062483

A. Introduction

Judul Project : Phone Store App

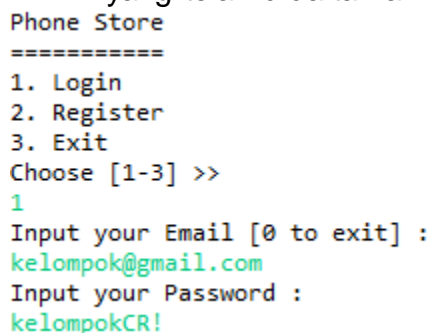
Phone store app merupakan sebuah program yang dapat memiliki berbagai fitur seperti :

1. Register : ketika menggunakan program tersebut kita harus register untuk dapat menggunakan program.



```
Phone Store
=====
1. Login
2. Register
3. Exit
Choose [1-3] >>
2
Input your name [0 to exit] :
Arya Fiqri Rizky
Input your Email [0 to exit] :
kelompok@gmail.com
Input your Password (At least 6 character and must contains 1 Symbol) [0 to exit] :
kelompokCR!
```

2. Login : setelah register maka dapat login menggunakan email dan password yang telah didaftarkan



```
Phone Store
=====
1. Login
2. Register
3. Exit
Choose [1-3] >>
1
Input your Email [0 to exit] :
kelompok@gmail.com
Input your Password :
kelompokCR!
```

Setelah login akan muncul menu untuk melakukan

```
Hello, Arya Fiqri Rizky
Logged at : 2022-06-04
```

```
=====
```

```
Chose your menu
```

1. Browse & Buy Phone
2. View Checkout
3. Input Ratings
4. Create Shop & Sell
5. Log out !

```
Choose one >>
```

3. Browse & buy phone : pengguna dapat membeli phone yang tersedia pada list penjualan.

```
Console Phone.java
Main (21) [Java Application] C:\Program Files\Java\jdk-16\bin\javaw.e
=====
Phone ID      : 4
Phone shop    : Mi store
Phone Brand   : Xiaomi
Phone Type    : Xiaomi note 8
Processor     : Snapdragon 665
RAM           : 4 Gb
Storage       : 64 Gb
Year          : 2019
Price         : 4000000
BATTERY       : 3800
Ratings       : 7.0/10
=====
Phone ID      : 5
Phone shop    : Harapan cellular
Phone Brand   : Oppo
Phone Type    : Oppo F1 s
Processor     : Snapdragon 616
RAM           : 3 Gb
Storage       : 32 Gb
Year          : 2018
Price         : 1300000
BATTERY       : 2500
Ratings       : 7.0/10
=====
Choose Your Phone (Use Phone ID / Type 0 To Exit) :
```

4. View Checkout : setelah melakukan pembelian pengguna dapat melihat barang yang sudah dibeli pada menu checkout

```
Checkout
```

```
=====
```

1. Xiaomi note 8
2. Oppo F1 s

```
Press Enter When You Are Done...
```

5. Input ratings : pengguna dapat memasukkan input 1-10 setelah membeli produk.

You Bought :

1. Xiaomi note 8

2. Oppo F1 s

Type The Phone Name To Rate : **Xiaomi note 8**

Rate this phone from 1 - 10 scale :

10

6. Create shop & sell phone : pengguna dapat membuka toko dan menjual handphone.

Name Of Your Shop :

binus store

Insert Phone's Brand [Iphone/Oppo/Xiaomi/Vivo/Samsung] :

Vivo

Insert Phone's Type :

vivo m4

Insert Phone's Processor :

android

Insert Phone's Ram :

3

Insert Phone's Storage :

32

Insert Phone's Year :

2018

Insert Phone's Price :

3000000

Insert Phone's Battery :

2000

Insert Phone's Rating :

8

Your store is created !

7. Logout : pengguna akan kembali ke menu login dan register apabila memilih menu logout.

B. Object-oriented Design Principles

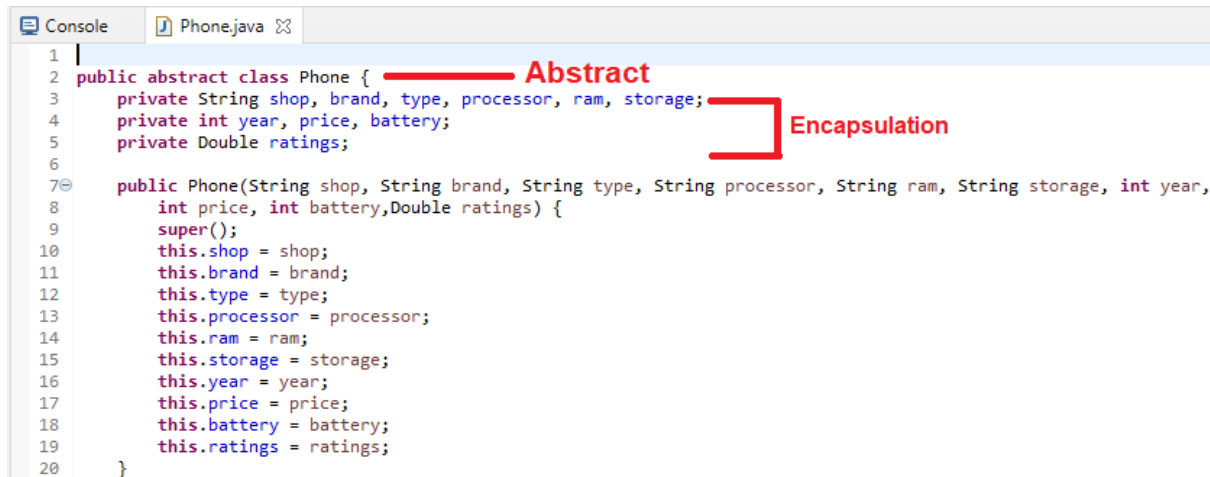
- Encapsulation implementation

```
1 |
2 | public abstract class Phone { Abstract
3 |     private String shop, brand, type, processor, ram, storage;
4 |     private int year, price, battery;
5 |     private Double ratings;
6 |
7 |     public Phone(String shop, String brand, String type, String processor, String ram, String storage, int year,
8 |         int price, int battery, Double ratings) {
9 |         super();
10 |         this.shop = shop;
11 |         this.brand = brand;
12 |         this.type = type;
13 |         this.processor = processor;
14 |         this.ram = ram;
15 |         this.storage = storage;
16 |         this.year = year;
17 |         this.price = price;
18 |         this.battery = battery;
19 |         this.ratings = ratings;
20 |     }
```

The screenshot shows a Java IDE with a file named Phone.java. The code defines an abstract class Phone with private attributes (shop, brand, type, processor, ram, storage, year, price, battery, ratings) and a constructor. Annotations highlight 'Abstract' and 'Encapsulation'.

- Abstraction implementation

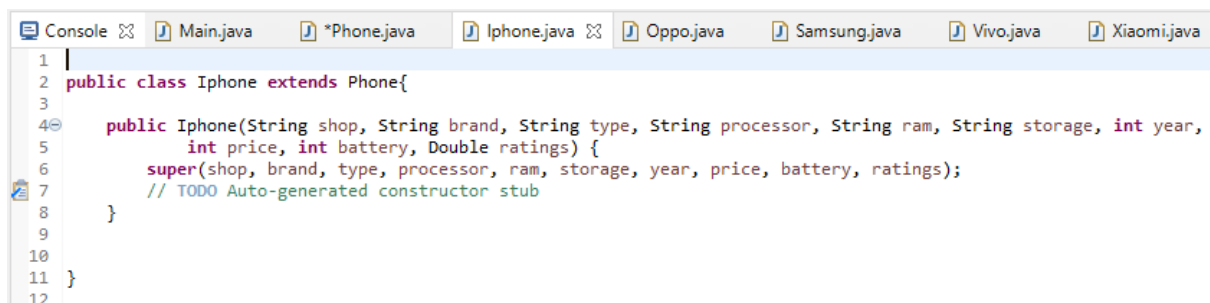
Terdapat class phone pada project kami yang berfungsi sebagai parent class dari berbagai macam merk handphone.



```
1 |
2 | public abstract class Phone { Abstract
3 |     private String shop, brand, type, processor, ram, storage;
4 |     private int year, price, battery;
5 |     private Double ratings;
6 |
7 |     public Phone(String shop, String brand, String type, String processor, String ram, String storage, int year,
8 |         int price, int battery, Double ratings) {
9 |         super();
10 |         this.shop = shop;
11 |         this.brand = brand;
12 |         this.type = type;
13 |         this.processor = processor;
14 |         this.ram = ram;
15 |         this.storage = storage;
16 |         this.year = year;
17 |         this.price = price;
18 |         this.battery = battery;
19 |         this.ratings = ratings;
20 |     }
```

- **Inheritance implementation**

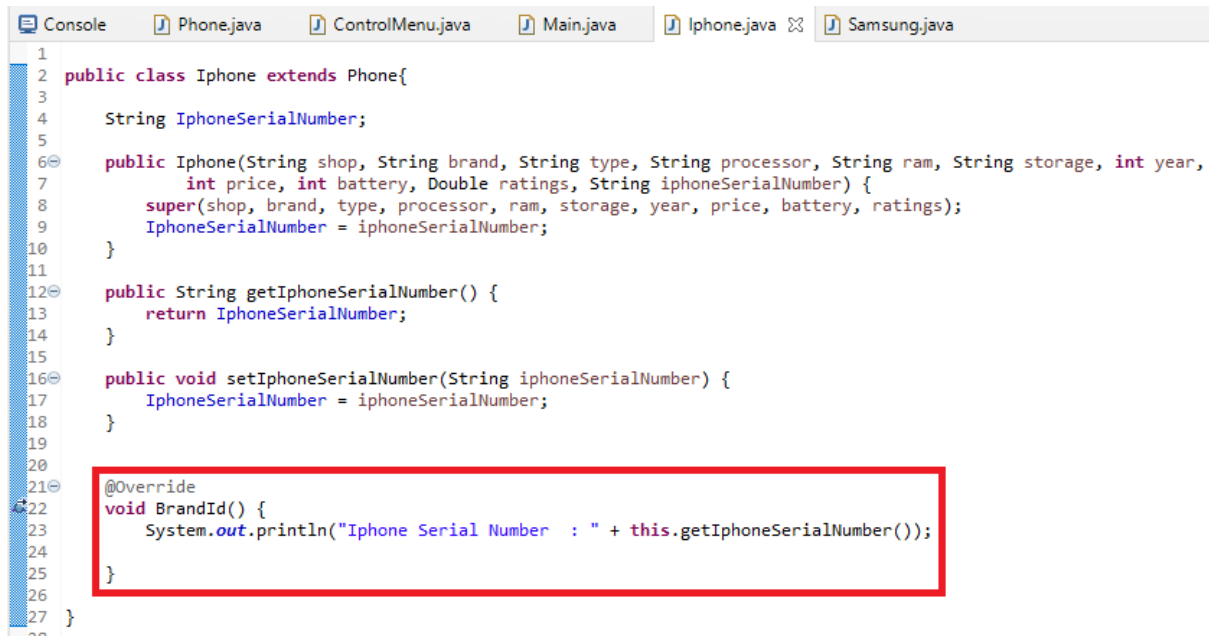
Kemudian, salah satu contohnya class iphone melakukan extends ke class phone.class iphone merupakan child class dari class phone.



```
1 |
2 | public class Iphone extends Phone{
3 |
4 |     public Iphone(String shop, String brand, String type, String processor, String ram, String storage, int year,
5 |         int price, int battery, Double ratings) {
6 |         super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
7 |         // TODO Auto-generated constructor stub
8 |     }
9 |
10 |
11 | }
12 |
```

- **Polymorphism implementation**

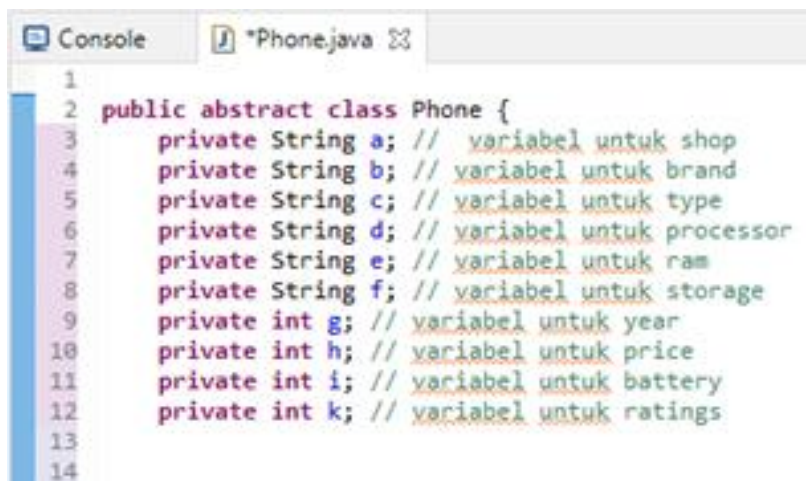
Implementasi polymorphism pada project kami terletak pada setiap class pada merk handphone yang memiliki atribut variabel “serial number”.



```
1 public class Iphone extends Phone{
2
3
4     String IphoneSerialNumber;
5
6     public Iphone(String shop, String brand, String type, String processor, String ram, String storage, int year,
7         int price, int battery, Double ratings, String iphoneSerialNumber) {
8         super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
9         IphoneSerialNumber = iphoneSerialNumber;
10    }
11
12    public String getIphoneSerialNumber() {
13        return IphoneSerialNumber;
14    }
15
16    public void setIphoneSerialNumber(String iphoneSerialNumber) {
17        IphoneSerialNumber = iphoneSerialNumber;
18    }
19
20
21    @Override
22    void BrandId() {
23        System.out.println("Iphone Serial Number : " + this.getIphoneSerialNumber());
24    }
25
26
27 }
```

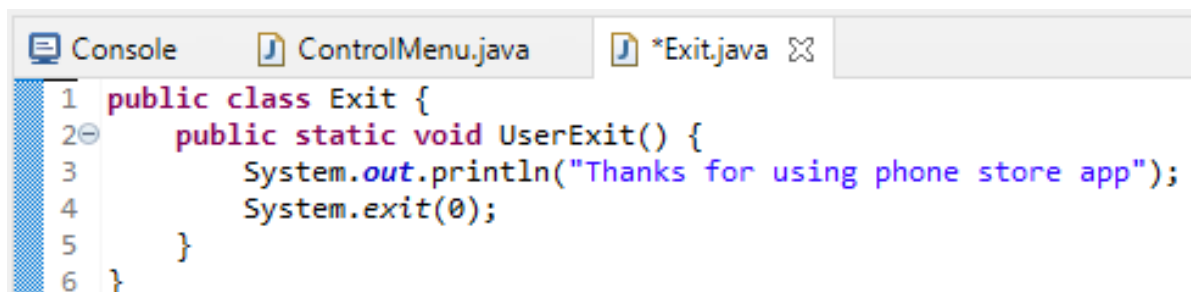
C. Object Smell

1. Dispensable smell : The comments



```
1
2 public abstract class Phone {
3     private String a; // variabel untuk shop
4     private String b; // variabel untuk brand
5     private String c; // variabel untuk type
6     private String d; // variabel untuk processor
7     private String e; // variabel untuk ram
8     private String f; // variabel untuk storage
9     private int g; // variabel untuk year
10    private int h; // variabel untuk price
11    private int i; // variabel untuk battery
12    private int k; // variabel untuk ratings
13
14 }
```

2. Dispensable smell : Lazy Class



```
1 public class Exit {
2     public static void UserExit() {
3         System.out.println("Thanks for using phone store app");
4         System.exit(0);
5     }
6 }
```

3. Dispensable smell : Dead Code

```

public void BuyPhone(ArrayList<Phone> list) {
    printPhoneList(list);
    int index;

    // String input = "";
    // int index = -1;
    // do {
    //     System.out.println(list.size());
    //     System.out.println("Choose your Phone (Use Phone ID or Brand to Choose) : ");
    //     input = scan.nextLine();
    //     index = valid.checkPhone(list, input);
    // } while (index == -1);
    // user.AddPhoneList(list.get(index));

    System.out.println("Choose Your Phone (Use Phone ID / Type 0 To Exit) : ");
    index = scan.nextInt();
    if (index == 0) {}
    else
    {
        String Type;
        Type = list.get(index-1).getC();
        System.out.printf("%s ", Type);
        checkout.add(new UserCheckout(Type));

        System.out.println("Phone added to chart!");
        scan.nextLine();
    }
}

```

4. Couplers smell : Inappropriate Intimacy

Console ControlMenu.java Phone.java

```

53
54 public void printPhoneList(ArrayList<Phone> list) {
55     int index = 1;
56     System.out.println("\tAll Phone");
57     System.out.println("=====");
58     for (Phone phone : list) {
59         System.out.println("Phone ID      : " + index++);
60         System.out.println("Phone shop  : " + phone.getA());
61         System.out.println("Phone Brand : " + phone.getB());
62         System.out.println("Phone Type  : " + phone.getC());
63         System.out.println("Processor   : " + phone.getD());
64         System.out.println("RAM         : " + phone.getE());
65         System.out.println("Storage     : " + phone.getF());
66         System.out.println("Year        : " + phone.getG());
67         System.out.println("Price       : " + phone.getH());
68         System.out.println("Baterry    : " + phone.getI());
69         System.out.println("Ratings    : " + phone.getj() + "/10");
70         System.out.println("=====");
71     }

```

5. Dispensable smell : Duplicate Code

```

167 System.out.println("Insert Phone's Brand [Iphone/Oppo/Xiaomi/Vivo/Samsung] : ");
168 PhoneBrand = scan.nextLine();
169 if (PhoneBrand.equalsIgnoreCase("Iphone"))
170 {
171     System.out.println("Insert Phone's Type : ");
172     PhoneType = scan.nextLine();
173     System.out.println("Insert Phone's Processor : ");
174     PhoneProcessor = scan.nextLine();
175     System.out.println("Insert Phone's Ram : ");
176     PhoneRam = scan.nextLine();
177     System.out.println("Insert Phone's Storage : ");
178     PhoneStorage = scan.nextLine();
179     System.out.println("Insert Phone's Year : ");
180     PhoneYear = scan.nextInt();
181     System.out.println("Insert Phone's Price : ");
182     PhonePrice = scan.nextInt();
183     System.out.println("Insert Phone's Battery : ");
184     PhoneBattery = scan.nextInt();
185     System.out.println("Insert Phone's Rating : ");
186     PhoneRatings = scan.nextDouble();
187     list.add(new Iphone(ShopName,PhoneBrand,PhoneType,PhoneProcessor,PhoneRam,PhoneStorage,PhoneYear,PhonePrice,PhoneBattery,PhoneRati
188 }
189 else if(PhoneBrand.equalsIgnoreCase("Oppo"))
190 {
191     System.out.println("Insert Phone's Type : ");
192     PhoneType = scan.nextLine();
193     System.out.println("Insert Phone's Processor : ");
194     PhoneProcessor = scan.nextLine();
195     System.out.println("Insert Phone's Ram : ");
196     PhoneRam = scan.nextLine();
197     System.out.println("Insert Phone's Storage : ");
198     PhoneStorage = scan.nextLine();
199     System.out.println("Insert Phone's Year : ");
200     PhoneYear = scan.nextInt();
201     System.out.println("Insert Phone's Price : ");
202     PhonePrice = scan.nextInt();
203     System.out.println("Insert Phone's Battery : ");
204     PhoneBattery = scan.nextInt();
205     System.out.println("Insert Phone's Rating : ");
206     PhoneRatings = scan.nextDouble();
207     list.add(new Oppo(ShopName,PhoneBrand,PhoneType,PhoneProcessor,PhoneRam,PhoneStorage,PhoneYear,PhonePrice,PhoneBattery,PhoneRating
208 }

```

6. Bloaters Smell: Long Method

```

17
18 public Main(boolean onApp) {
19     String shop = "Ibox";
20     String brand = "Iphone";
21     String type = "Iphone 8 Plus";
22     String processor = "A11 Bionic";
23     String ram = "4 Gb";
24     String storage = "32 gb";
25     int year = 2017;
26     int price = 2000000;
27     int battery = 3500;
28     double rating = 7.5;
29     Iphone iphone = new Iphone(shop, brand, type, processor, ram, storage, year, price, battery, rating);
30     IP.add(iphone);
31 }

```

7. Bloaters Smell: Large Class

```

2
3 public class Phone {
4     //Iphone
5     private String iphoneShop;
6     private String iphoneBrand;
7     private String iphoneType;
8     private String iphoneProcessor;
9     private String iphoneRam;
10    private String iphoneStorage;
11    private int iphoneYear;
12    private int iphonePrice;
13    private int iphoneBattery;
14    private double iphoneRatings;
15
16    public String getIphoneShop() {
17        return iphoneShop;
18    }
19    public void setIphoneShop(String iphoneShop) {
20        this.iphoneShop = iphoneShop;
21    }
22    public String getIphoneBrand() {
23        return iphoneBrand;
24    }
25    public void setIphoneBrand(String iphoneBrand) {
26        this.iphoneBrand = iphoneBrand;
27    }
28    public String getIphoneType() {
29        return iphoneType;
30    }
31    public void setIphoneType(String iphoneType) {
32        this.iphoneType = iphoneType;
33    }
34    public String getIphoneProcessor() {
35        return iphoneProcessor;
36    }
37    public void setIphoneProcessor(String iphoneProcessor) {
38        this.iphoneProcessor = iphoneProcessor;
39    }
40    public String getIphoneRam() {
41        return iphoneRam;
42    }
43    public void setIphoneRam(String iphoneRam) {
44        this.iphoneRam = iphoneRam;
45    }
46    public String getIphoneStorage() {
47        return iphoneStorage;
48    }
49    public void setIphoneStorage(String iphoneStorage) {
50        this.iphoneStorage = iphoneStorage;
51    }
52    public int getIphoneYear() {
53        return iphoneYear;
54    }
55    public void setIphoneYear(int iphoneYear) {
56        this.iphoneYear = iphoneYear;
57    }
58    public int getIphonePrice() {
59        return iphonePrice;
60    }
61    public void setIphonePrice(int iphonePrice) {
62        this.iphonePrice = iphonePrice;
63    }
64    public int getIphoneBattery() {
65        return iphoneBattery;
66    }
67    public void setIphoneBattery(int iphoneBattery) {
68        this.iphoneBattery = iphoneBattery;
69    }
70    public double getIphoneRatings() {
71        return iphoneRatings;
72    }
73    public void setIphoneRatings(double iphoneRatings) {
74        this.iphoneRatings = iphoneRatings;
75    }

```

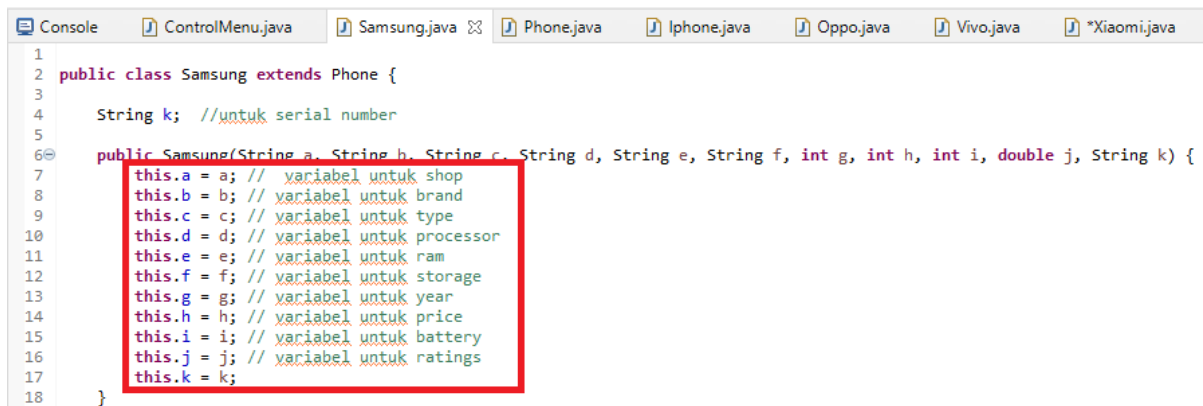


```

76= public Phone(String iphoneShop, String iphoneBrand, String iphoneType, String iphoneProcessor, String iphoneRam,
77     String iphoneStorage, int iphoneYear, int iphonePrice, int iphoneBattery, double iphoneRatings) {
78     super();
79     this.iphoneShop = iphoneShop;
80     this.iphoneBrand = iphoneBrand;
81     this.iphoneType = iphoneType;
82     this.iphoneProcessor = iphoneProcessor;
83     this.iphoneRam = iphoneRam;
84     this.iphoneStorage = iphoneStorage;
85     this.iphoneYear = iphoneYear;
86     this.iphonePrice = iphonePrice;
87     this.iphoneBattery = iphoneBattery;
88     this.iphoneRatings = iphoneRatings;
89 }
90
91 //Samsung
92 private String samsungShop;
93 private String samsungBrand;
94 private String samsungType;
95 private String samsungProcessor;
96 private String samsung;
97 private String samsungStorage;
98 private int samsungYear;
99 private int samsungPrice;
100 private int samsungBattery;
101 private double samsungRatings;
102
103= public String getSamsungShop() {
104     return samsungShop;
105 }
106= public void setSamsungShop(String samsungShop) {
107     this.samsungShop = samsungShop;
108 }
109= public String getSamsungBrand() {
110     return samsungBrand;
111 }
112= public void setSamsungBrand(String samsungBrand) {
113     this.samsungBrand = samsungBrand;
114 }
115= public String getSamsungType() {
116     return samsungType;
117 }
118= public void setSamsungType(String samsungType) {
119     this.samsungType = samsungType;
120 }
121= public String getSamsungProcessor() {
122     return samsungProcessor;
123 }
124= public void setSamsungProcessor(String samsungProcessor) {
125     this.samsungProcessor = samsungProcessor;
126 }
127= public String getSamsung() {
128     return samsung;
129 }
130= public void setSamsung(String samsung) {
131     this.samsung = samsung;
132 }
133= public String getSamsungStorage() {
134     return samsungStorage;
135 }
136= public void setSamsungStorage(String samsungStorage) {
137     this.samsungStorage = samsungStorage;
138 }
139= public int getSamsungYear() {
140     return samsungYear;
141 }
142= public void setSamsungYear(int samsungYear) {
143     this.samsungYear = samsungYear;
144 }
145= public int getSamsungPrice() {
146     return samsungPrice;
147 }
148= public void setSamsungPrice(int samsungPrice) {
149     this.samsungPrice = samsungPrice;
150 }
151= public int getSamsungBattery() {
152     return samsungBattery;
153 }
154= public void setSamsungBattery(int samsungBattery) {
155     this.samsungBattery = samsungBattery;
156 }
157= public double getSamsungRatings() {
158     return samsungRatings;
159 }
160= public void setSamsungRatings(double samsungRatings) {
161     this.samsungRatings = samsungRatings;
162 }
163= public Phone(String iphoneShop, String iphoneBrand, String iphoneType, String iphoneProcessor, String iphoneRam,
164     String iphoneStorage, int iphoneYear, int iphonePrice, int iphoneBattery, double iphoneRatings,
165     String samsungShop, String samsungBrand, String samsungType, String samsungProcessor, String samsung,
166     String samsungStorage, int samsungYear, int samsungPrice, int samsungBattery, double samsungRatings) {
167     super();
168     this.iphoneShop = iphoneShop;
169     this.iphoneBrand = iphoneBrand;
170     this.iphoneType = iphoneType;
171     this.iphoneProcessor = iphoneProcessor;
172     this.iphoneRam = iphoneRam;
173     this.iphoneStorage = iphoneStorage;
174     this.iphoneYear = iphoneYear;
175     this.iphonePrice = iphonePrice;
176     this.iphoneBattery = iphoneBattery;
177     this.iphoneRatings = iphoneRatings;
178     this.samsungShop = samsungShop;
179     this.samsungBrand = samsungBrand;
180     this.samsungType = samsungType;
181     this.samsungProcessor = samsungProcessor;
182     this.samsung = samsung;
183     this.samsungStorage = samsungStorage;
184     this.samsungYear = samsungYear;
185     this.samsungPrice = samsungPrice;
186     this.samsungBattery = samsungBattery;
187     this.samsungRatings = samsungRatings;
188 }
189
190 }

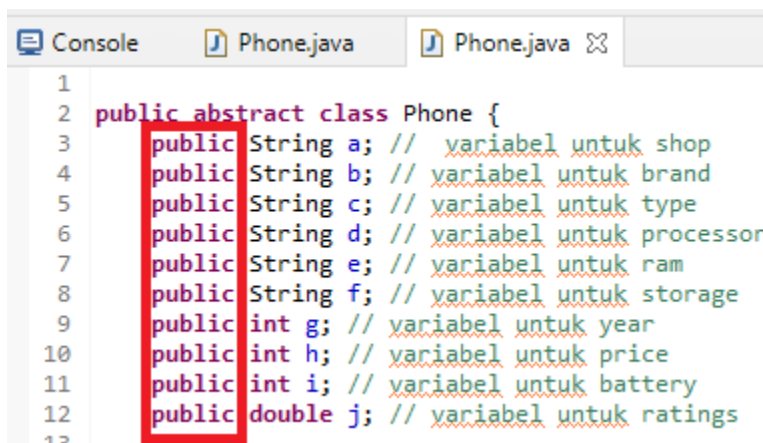
```

8. Object oriented design smell : Pull up constructor body



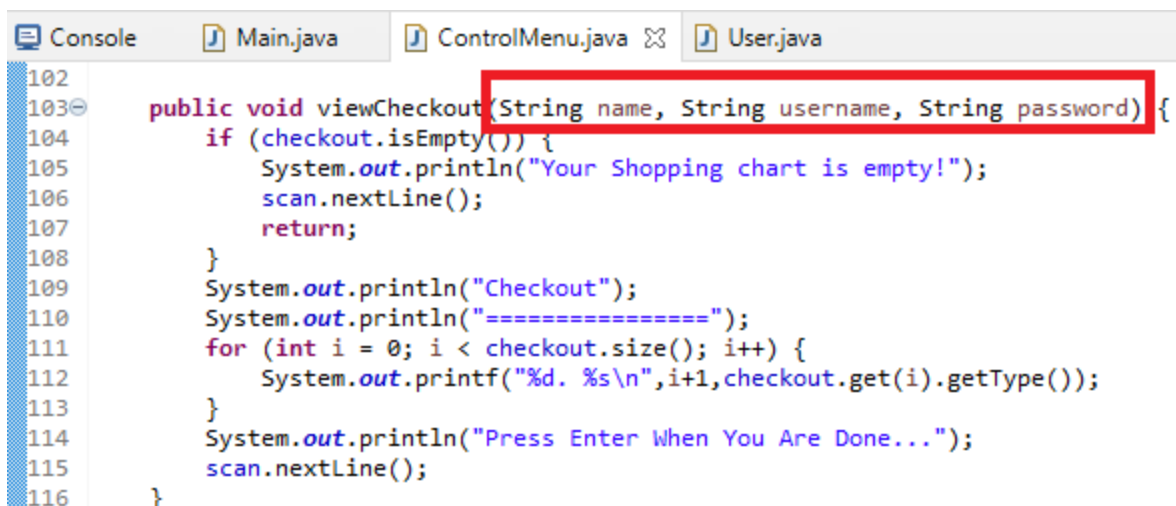
```
1 public class Samsung extends Phone {
2
3     String k; //untuk serial number
4
5     public Samsung(String a, String b, String c, String d, String e, String f, int g, int h, int i, double j, String k) {
6
7         this.a = a; // variabel untuk shop
8         this.b = b; // variabel untuk brand
9         this.c = c; // variabel untuk type
10        this.d = d; // variabel untuk processor
11        this.e = e; // variabel untuk ram
12        this.f = f; // variabel untuk storage
13        this.g = g; // variabel untuk year
14        this.h = h; // variabel untuk price
15        this.i = i; // variabel untuk battery
16        this.j = j; // variabel untuk ratings
17        this.k = k;
18    }
```

9. Encapsulation Smell



```
1 public abstract class Phone {
2
3     public String a; // variabel untuk shop
4     public String b; // variabel untuk brand
5     public String c; // variabel untuk type
6     public String d; // variabel untuk processor
7     public String e; // variabel untuk ram
8     public String f; // variabel untuk storage
9     public int g; // variabel untuk year
10    public int h; // variabel untuk price
11    public int i; // variabel untuk battery
12    public double j; // variabel untuk ratings
13 }
```

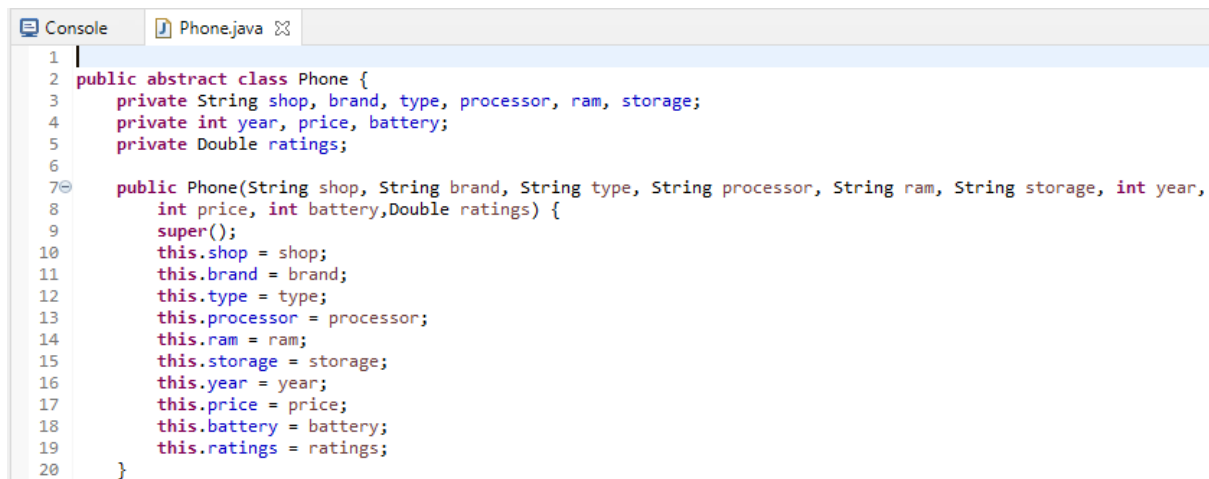
10. Bloaters Smell : Long Parameter List



```
102
103 public void viewCheckout(String name, String username, String password) {
104     if (checkout.isEmpty()) {
105         System.out.println("Your Shopping chart is empty!");
106         scan.nextLine();
107         return;
108     }
109     System.out.println("Checkout");
110     System.out.println("=====");
111     for (int i = 0; i < checkout.size(); i++) {
112         System.out.printf("%d. %s\n", i+1, checkout.get(i).getType());
113     }
114     System.out.println("Press Enter When You Are Done...");
115     scan.nextLine();
116 }
```

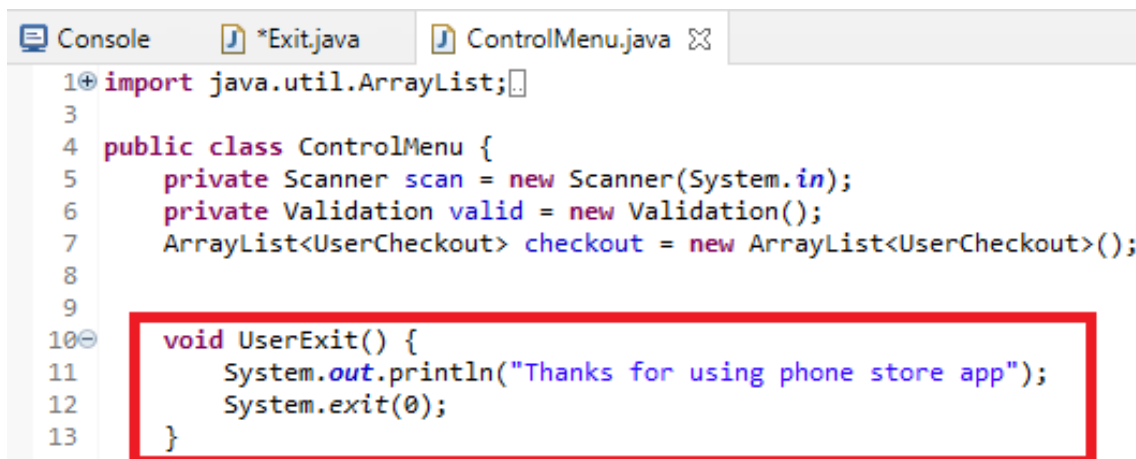
D. Refactoring Process

1. **Solusi refactoring Dispensable smell The comments** : dengan cara mengubah nama variabel yang mudah dipahami sehingga tidak membutuhkan comment yang berlebihan.



```
1 |
2 public abstract class Phone {
3     private String shop, brand, type, processor, ram, storage;
4     private int year, price, battery;
5     private Double ratings;
6
7     public Phone(String shop, String brand, String type, String processor, String ram, String storage, int year,
8         int price, int battery, Double ratings) {
9         super();
10        this.shop = shop;
11        this.brand = brand;
12        this.type = type;
13        this.processor = processor;
14        this.ram = ram;
15        this.storage = storage;
16        this.year = year;
17        this.price = price;
18        this.battery = battery;
19        this.ratings = ratings;
20    }
```

2. **Solusi refactoring Dispensable smell Lazy Class** : dengan cara menggabungkan code tersebut kedalam class yang digunakan menggunakan method baru.



```
1 import java.util.ArrayList;
2
3
4 public class ControlMenu {
5     private Scanner scan = new Scanner(System.in);
6     private Validation valid = new Validation();
7     ArrayList<UserCheckout> checkout = new ArrayList<UserCheckout>();
8
9
10    void UserExit() {
11        System.out.println("Thanks for using phone store app");
12        System.exit(0);
13    }
```

3. **Solusi refactoring Dispensable smell Dead Code** : dengan cara menghapus code yang sudah tidak digunakan.
4. **Solusi refactoring Couplers smell** : Inappropriate Intimacy dengan cara hide data phone pada class "ControlMenu". Karena class "ControlMenu" telah mengakses internal field dan method dari class "Phone".

```
Console Phone.java ControlMenu.java Main.java Iphone.java
58
59
60 public void printPhoneList(ArrayList<Phone> list) {
61     int index = 1;
62     System.out.println("\tAll Phone");
63     System.out.println("=====");
64     for (Phone phone : list) {
65         System.out.println("Phone ID           : " + index++);
66         phone.PhoneForSale(list);
67     }
68
69 }
```

5. **Solusi refactoring Dispensable smell Duplicate Code** : dengan cara menggabungkan code yang memiliki fungsional yang sama agar meminimalisir baris code.

```
Console Phone.java *ControlMenu.java Main.java Iphone.java Samsung.java
146 }
147
148
149 public static void createShop(ArrayList<Phone> list){
150     String ShopName,PhoneBrand,PhoneType,PhoneProcessor,PhoneRam,PhoneStorage;
151     Integer PhoneYear,PhonePrice,PhoneBattery;
152     Double PhoneRatings;
153
154     System.out.println("Name Of Your Shop : ");
155     ShopName = scan.nextLine();
156
157     do {
158         System.out.println("Insert Phone's Brand [Iphone/Oppo/Xiaomi/Vivo/Samsung] : ");
159         PhoneBrand = scan.nextLine();
160
161         System.out.println("Insert Phone's Type : ");
162         PhoneType = scan.nextLine();
163         System.out.println("Insert Phone's Processor : ");
164         PhoneProcessor = scan.nextLine();
165         System.out.println("Insert Phone's Ram : ");
166         PhoneRam = scan.nextLine();
167         System.out.println("Insert Phone's Storage : ");
168         PhoneStorage = scan.nextLine();
169         System.out.println("Insert Phone's Year : ");
170         PhoneYear = scan.nextInt();
171         System.out.println("Insert Phone's Price : ");
172         PhonePrice = scan.nextInt();
173         System.out.println("Insert Phone's Battery : ");
174         PhoneBattery = scan.nextInt();
175         System.out.println("Insert Phone's Rating : ");
176         PhoneRatings = scan.nextDouble();
177     }
```

6. **Solusi refactoring Bloaters smell Long Method**: dengan cara menggunakan “.add new()” pada ArrayList untuk menambahkan data kedalam child class phone sehingga tidak perlu menulis ulang nama variabel

```

1 import java.util.*;
2
3 public class Main {
4     private Scanner scan = new Scanner(System.in);
5     private ArrayList<User> UserList = new ArrayList<User>();
6     private Controller<User> ctrl = new Controller<User>();
7     public ArrayList<Phone> PhoneList = new ArrayList<Phone>();
8
9     void menu() {
10         System.out.println("\nPhone Store");
11         System.out.println("=====");
12         System.out.println("1. Login");
13         System.out.println("2. Register");
14         System.out.println("3. Exit");
15     }
16
17     public Main(boolean onApp) {
18         PhoneList.add(new Iphone("Ibox", "Iphone", "Iphone 8 Plus", "A11 Bionic", "4 Gb", "32 Gb", 2017, 2000000, 3500, 7.5));
19         PhoneList.add(new Samsung("Samsung official store", "Samsung", "Samsung A52", "Octa Core", "8 Gb", "128 Gb", 2021, 4300000, 5000, 7.0));
20         PhoneList.add(new Vivo("Ds cell", "Vivo", "Vivo T1 Pro", "Snapdragon 778G", "8 Gb", "128 Gb", 2021, 4300000, 5000, 7.0));
21         PhoneList.add(new Xiaomi("Mi store", "Xiaomi", "Xiaomi note 8", "Snapdragon 665", "4 Gb", "64 Gb", 2019, 4000000, 3800, 7.0));
22         PhoneList.add(new Oppo("Harapan cellular", "Oppo", "Oppo F1 s", "Snapdragon 616", "3 Gb", "32 Gb", 2018, 1300000, 2500, 7.0));
23     }

```

7. **Solusi refactoring Bloaters smell Large Class:** dengan mengimplementasikan Inheritance. Pisahkan brand handphone ke dalam class yang berbeda-beda dengan cara membuat class untuk setiap brand handphone sebagai child class dan buat class baru sebagai parent class.

```

2 public abstract class Phone {
3     String shop, brand, type, processor, ram, storage;
4     int year, price, battery;
5     Double ratings;
6
7
8
9
10    public Phone(String shop, String brand, String type, String processor, String ram, String storage, int year,
11        int price, int battery, Double ratings) {
12        super();
13        this.shop = shop;
14        this.brand = brand;
15        this.type = type;
16        this.processor = processor;
17        this.ram = ram;
18        this.storage = storage;
19        this.year = year;
20        this.price = price;
21        this.battery = battery;
22        this.ratings = ratings;
23    }

```

Parent Class

```

2 public class Iphone extends Phone{
3
4    public Iphone(String shop, String brand, String type, String processor, String ram, String storage, int year,
5        int price, int battery, Double ratings) {
6        super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
7        // TODO Auto-generated constructor stub
8    }
9
10 }

```

```

2 public class Samsung extends Phone {
3
4    public Samsung(String shop, String brand, String type, String processor, String ram, String storage, int year,
5        int price, int battery, Double ratings) {
6        super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
7        // TODO Auto-generated constructor stub
8    }
9
10 }

```

```

2 public class Vivo extends Phone {
3
4     public Vivo(String shop, String brand, String type, String processor, String ram, String storage, int year,
5                 int price, int battery, Double ratings) {
6         super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
7         // TODO Auto-generated constructor stub
8     }
9
10
11
12 }

```

```

2 public class Xiaomi extends Phone {
3
4     public Xiaomi(String shop, String brand, String type, String processor, String ram, String storage, int year,
5                  int price, int battery, Double ratings) {
6         super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
7         // TODO Auto-generated constructor stub
8     }
9
10
11
12 }

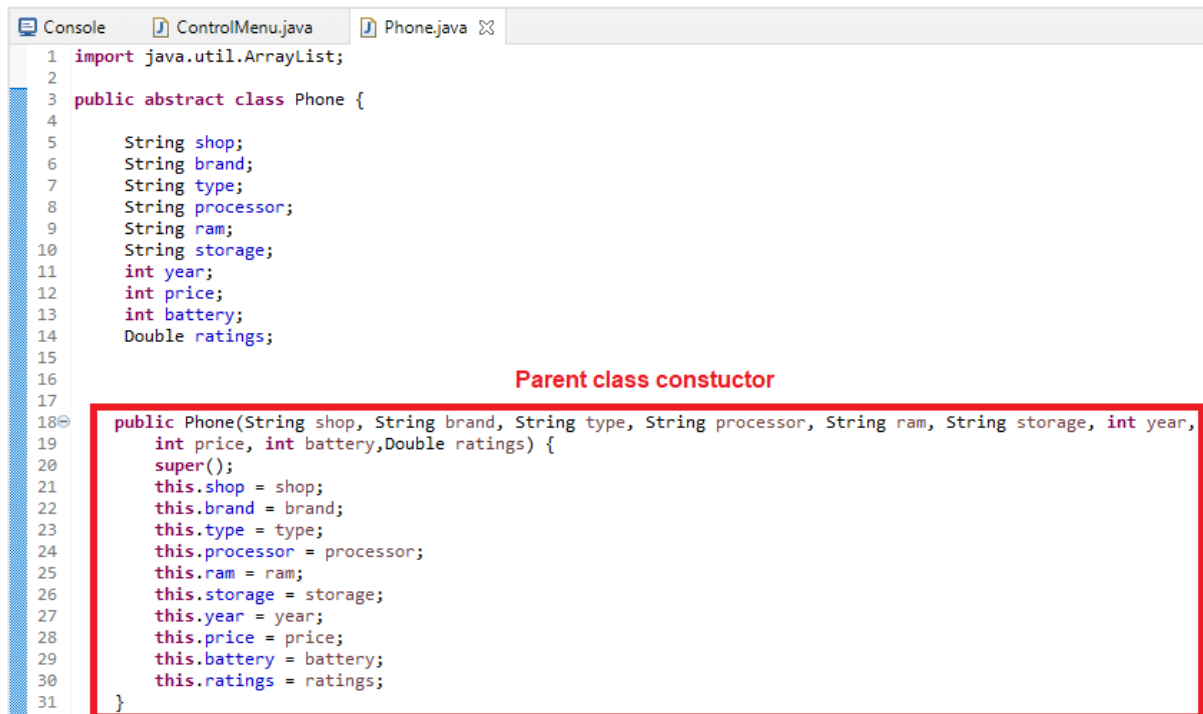
```

```

2 public class Oppo extends Phone {
3
4     public Oppo(String shop, String brand, String type, String processor, String ram, String storage, int year,
5                int price, int battery, Double ratings) {
6         super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
7         // TODO Auto-generated constructor stub
8     }
9
10
11

```

8. **Solusi Object oriented design smell : Pull up constructor body** dengan membuat constructor pada parent class lalu menggunakan kata kunci “super” pada child class untuk mengambil constructor dari parent class.



```

1 import java.util.ArrayList;
2
3 public abstract class Phone {
4
5     String shop;
6     String brand;
7     String type;
8     String processor;
9     String ram;
10    String storage;
11    int year;
12    int price;
13    int battery;
14    Double ratings;
15
16
17
18    public Phone(String shop, String brand, String type, String processor, String ram, String storage, int year,
19                int price, int battery, Double ratings) {
20        super();
21        this.shop = shop;
22        this.brand = brand;
23        this.type = type;
24        this.processor = processor;
25        this.ram = ram;
26        this.storage = storage;
27        this.year = year;
28        this.price = price;
29        this.battery = battery;
30        this.ratings = ratings;
31    }
32

```

Parent class constructor

```

1 public class Samsung extends Phone {
2
3     String SamsungSerialNumber;
4
5
6
7     public Samsung(String shop, String brand, String type, String processor, String ram, String storage, int year,
8         int price, int battery, Double ratings, String samsungSerialNumber) {
9         super(shop, brand, type, processor, ram, storage, year, price, battery, ratings);
10        SamsungSerialNumber = samsungSerialNumber;
11    }

```

9. **Solusi Encapsulation smell** dengan cara membuat private pada setiap atribut yang terdapat pada class tersebut

```

1 import java.util.ArrayList;
2
3 public abstract class Phone {
4
5     private String shop;
6     private String brand;
7     private String type;
8     private String processor;
9     private String ram;
10    private String storage;
11    private int year;
12    private int price;
13    private int battery;
14    private Double ratings;

```

10. **Solusi Bloaters Smell : Long Parameter List** dengan cara melakukan Preserve Whole Object pada method viewCheckout.

```

69
90 public void viewCheckout(User user) {
91     if (checkout.isEmpty()) {
92         System.out.println("Your Shopping chart is empty!");
93         scan.nextLine();
94         return;
95     }
96     System.out.println("Checkout");
97     System.out.println("=====");
98     for (int i = 0; i < checkout.size(); i++) {
99         System.out.printf("%d. %s\n", i+1, checkout.get(i).getType());
100    }
101    System.out.println("Press Enter When You Are Done...");
102    scan.nextLine();
103 }

```

E. Implementation

Hasil implementasi kami buat dalam bentuk 2 file kode yang berbeda yaitu file : code smell dan hasil refactoring. Untuk link backup code dapat diakses melalui link berikut ini :

https://drive.google.com/drive/folders/1eFhohCF_Pfgk0p4W408maBxa5LfZXfbn?usp=sharing