Projektowanie Obiektowe
Radosław Kopeć 305333
Grupa pon 9:35A

2.1

Zmiany hronologicznie:

Dodanie interfejsu ICategory

Interface ma za zadanie zdefiniować klasy będące kategoriami

```
package pl.edu.agh.dronka.shop.model;

import java.util.Map;

public interface ICategory {


    String getDisplayName();
    Map<String, Object> getPropertiesMap();

}
```

Klasa Item:

Dodanie pola categoryField: (kazdy iteam ma kategorie)

```
private ICategory categoryFiled;
```

Zmiana konstruktora:

```
public Item(String name, Category category, int price, int quantity,ICategory categoryFiled) {
    this.name = name;
    this.category = category;
    this.price = price;
    this.quantity = quantity;
    this.categoryFiled = categoryFiled;
}
```

Dodanie metody zwracającej Mapę parametrów Katergori danego przedmiotu

```
public Map<String, Object> getCategoryPropertiesMap() { return this.categoryFiled.getPropertiesMap(); }
```

Klasa ItemDetailsPanel:

Zmiana metody setItem, dodanie wyświetlania parametrów kategorii danego przedmiotu.

```
public void setItem(Item item) {
    infoPanel.removeAll();
    this.currentItem = item;

    Map<String, Object> propertiesMap = PropertiesHelper
            .getPropertiesMap(item);

    for (String displayName : propertiesMap.keySet()) {
        createInfoLabel(displayName, propertiesMap.get(displayName)
                .toString());
    }

    Map<String, Object> categoryPropertiesMap = item.getCategoryPropertiesMap();

    for (String displayName : categoryPropertiesMap.keySet()) {
        createInfoLabel(displayName, categoryPropertiesMap.get(displayName)
                .toString());
    }

    addToCartButton.setEnabled(item.getQuantity() > 0);
}
```

Stworzenie Klas reprezentujących Kategorie dziedziczących z ICategory

Book

```java
package pl.edu.agh.dronka.shop.model.categories;
import pl.edu.agh.dronka.shop.model.ICategory;

import java.util.LinkedHashMap;
import java.util.Map;

public class Book implements ICategory {
    private String displayedName;
    private int liczbaStron;
    private boolean twardaOprawa;

    public Book(int liczbaStron,boolean twardaOprawa){
        displayedName = "Książki";
        this.liczbaStron = liczbaStron;
        this.twardaOprawa = twardaOprawa;
    }
    @Override
    public Map<String, Object> getPropertiesMap() {
        Map<String, Object> propertiesMap = new LinkedHashMap<>();

        propertiesMap.put("Liczba stron", liczbaStron);
        propertiesMap.put("Oprawa", twardaOprawa);

        return propertiesMap;
    }

    @Override
    public String getDisplayName() {

        return displayedName;
    }
}
```

Elecrtronics

```java
import java.util.LinkedHashMap;
import java.util.Map;

public class Elecrtronics implements ICategory {

    private String displayedName;
    private boolean mobilny;
    private boolean gwarancja;
    private Category category;

    public Elecrtronics(boolean mobilny,boolean gwarancja){
        displayedName = "Elektronika";
        this.mobilny = mobilny;
        this.gwarancja = gwarancja;
    }
    @Override
    public Map<String, Object> getPropertiesMap() {
        Map<String, Object> propertiesMap = new LinkedHashMap<>();

        propertiesMap.put("Mobilny", mobilny);
        propertiesMap.put("Gwarancja", gwarancja);

        return propertiesMap;
    }

    @Override
    public String getDisplayName() {

        return displayedName;
    }

}
```

Food

```java
public class Food implements ICategory {
    private String displayedName;
    private String dataPrzydatnosci;

    public Food(String dataPrzydatnosci){
        displayedName = "Żywność";
        this.dataPrzydatnosci = dataPrzydatnosci;
    }
    @Override
    public Map<String, Object> getPropertiesMap() {
        Map<String, Object> propertiesMap = new LinkedHashMap<>();

        propertiesMap.put("data przydatnosci", dataPrzydatnosci);

        return propertiesMap;
    }

    @Override
    public String getDisplayName() {

        return displayedName;
    }
}
```

Sport

```java
public class Sport implements ICategory {
    private String displayedName;

    public Sport(){
        displayedName = "Sport";

    }
    @Override
    public Map<String, Object> getPropertiesMap() { return null; }

    @Override
    public String getDisplayName() { return displayedName; }
}
```

Muzyka

```
public class Muzyka implements ICategory {

    private String displayedName;
    private GatunekMuzyki gatunekMuzyki;
    private boolean video;
    private Category category;

    public Muzyka(boolean video,GatunekMuzyki gatunekMuzyki){
        displayedName = "Muzyka";
        this.video = video;
        this.gatunekMuzyki = gatunekMuzyki;
    }
    @Override
    public Map<String, Object> getPropertiesMap() {
        Map<String, Object> propertiesMap = new LinkedHashMap<>();

        propertiesMap.put("Doalczone Video", video);
        propertiesMap.put("Gatunek Muzyki", gatunekMuzyki);

        return propertiesMap;
    }

    @Override
    public String getDisplayName() {

        return displayedName;
    }

}
```

Enum klasa reprezentująca Gatunek muzyki

```
public enum GatunekMuzyki {
    HIPHOP("HIPHOP"), ROCK("ROCK");

    private String displayName;

    public String getDisplayName() { return displayName; }

    GatunekMuzyki(String displayName) { this.displayName = displayName; }

    public static GatunekMuzyki parseGatunekMuzyki(String gatunek){
        if(stringCompare(gatunek,"HIPHOP"))return HIPHOP;
        if(stringCompare(gatunek,"ROCK"))return ROCK;
        return null;
    }
```

Modyfikacja metody **readItems** w klasie **ShopProvider** tak aby generowała odpowiednie kategorie do przedmiotu:

```
private static List<Item> readItems(CSVReader reader, Category category) {
    List<Item> items = new ArrayList<>();

    try {
        reader.parse();
        List<String[]> data = reader.getData();

        for (String[] dataLine : data) {

            String name = reader.getValue(dataLine, name: "Nazwa");
            int price = Integer.parseInt(reader.getValue(dataLine, name: "Cena"));
            int quantity = Integer.parseInt(reader.getValue(dataLine,
                    name: "Ilość"));

            boolean isPolish = Boolean.parseBoolean(reader.getValue(
                    dataLine, name: "Tanie bo polskie"));
            boolean isSecondhand = Boolean.parseBoolean(reader.getValue(
                    dataLine, name: "Używany"));
```

```
ICategory categoryFileds = null;
switch (category){
    case FOOD:
        String dataPrzydatnosci = reader.getValue(dataLine, name: "Data");
        categoryFileds = new Food(dataPrzydatnosci);
        break;
    case BOOKS:
        int liczbaStron = Integer.parseInt(reader.getValue(dataLine,
                name: "Liczba stron"));
        boolean twardaOprawa = Boolean.parseBoolean(reader.getValue(
                dataLine, name: "Twarda oprawa"));
        categoryFileds = new Book(liczbaStron,twardaOprawa);
        break;
    case MUSIC:
        boolean viedo = Boolean.parseBoolean(reader.getValue(
                dataLine, name: "Video"));
        GatunekMuzyki gatunek = GatunekMuzyki.parseGatunekMuzyki(reader.getValue(
                dataLine, name: "Gatunek"
        ));
        categoryFileds = new Muzyka(viedo,gatunek);
        break;
    case SPORT:
        categoryFileds = new Sport();
        break;
```

```
        case ELECTRONICS:
            boolean mobilny = Boolean.parseBoolean(reader.getValue(
                    dataLine,  name: "Mobilny"));
            boolean gwarancja = Boolean.parseBoolean(reader.getValue(
                    dataLine,  name: "Gwarancja"));
            categoryFileds = new Elecrtronics(mobilny,gwarancja);
            break;
    }
    Item item = new Item(name, category, price, quantity,categoryFileds);
    item.setPolish(isPolish);
    item.setSecondhand(isSecondhand);

    items.add(item);
```

Dopisanie danych do plików csv

```
Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Liczba stron,Twarda oprawa
Nowe Przygody Gangu Czworga,50,2,TRUE,TRUE,500,TRUE
Zamodeluj swoje życie. Technologie obiektowe for dummies,120,15,FALSE,FALSE,200,FALSE
When The Smoke is Going Down : Wprowadzenie do testów wydajnościowych,90,3,TRUE,FALSE,2000,TRUE
```

```
Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Mobilny,Gwarancja
Telewizor LCD El Dżi,2000,10,FALSE,TRUE,FALSE,TRUE
Legendarny Bulbulator,99999999,1,TRUE,FALSE,TRUE,FALSE
```

```
Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Data
Zupa Studencka Instant,2,100,TRUE,FALSE,12.01.2021
Sałycel mrożony,6,30,TRUE,FALSE,01.02.2020
```

```
Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Video,Gatunek
Cliley Myrus - Big Ball of Mud,60,20,TRUE,FALSE,FALSE,HIPHOP
przyklad 2,60,20,TRUE,FALSE,FALSE,ROCK
```

Widoki po modyfikacji:

Dronka Shop

Nazwa: Zupa Studencka Instant
Cena: 2
Kategoria: Żywność
Ilość: 100
Tanie bo polskie: true
Używany: false
data przydatnosci: 12.01.2021

Dronka Shop

Nazwa: Ciley Myrus - Big Ball of Mud
Cena: 60
Kategoria: Muzyka
Ilość: 20
Tanie bo polskie: true
Używany: false
Doalczone Video: false
Gatunek Muzyki: HIPHOP

Dronka Shop

Nazwa: When The Smoke is Going Down : Wprowadzenie do testów wydajnościowych
Cena: 90
Kategoria: Książki
Ilość: 3
Tanie bo polskie: true
Używany: false
Liczba stron: 2000
Oprawa: true

Dronka Shop

Nazwa: Legendarny Bulbulator
Cena: 99999999
Kategoria: Elektronika
Ilość: 1
Tanie bo polskie: true
Używany: false
Mobilny: false
Gwarancja: false

2.2 Rozszerzenie panelu

Rozszerzenie interfacu ICategory o funkcje

Potrzebujemy miec dostep do pól booleanowych wewnątrz każdej z kategori aby w ItemFilter modyfikować “falgi” (po których polach filtrujemy)

```
void setBoolianPropertie(Boolean b,int propertieNumber);
boolean getBoolianProperie(int propertieNumber);
boolean equals(ICategory c);
```

Dodanie do klasy ItemFilter pola:

```
ICategory categorySpec;
```

Dodanie do funkcji appliesTo linijek:

```
public boolean appliesTo(Item item) {
    if (itemSpec.getName() != null
            && !itemSpec.getName().equals(item.getName())) {
        return false;
    }
    if (itemSpec.getCategory() != null
            && !itemSpec.getCategory().equals(item.getCategory())) {
        return false;
    }

    if (categorySpec != null
            && !categorySpec.equals(item.getCategoryFiled())) {
        return false;
    }

    // applies filter only if the flag (secondHand) is true)
    if (itemSpec.isSecondhand() && !item.isSecondhand()) {
        return false;
    }

    // applies filter only if the flag (polish) is true)
    if (itemSpec.isPolish() && !item.isPolish()) {
        return false;
    }

    return true;
}
```

Klasa Book dodanie funkcji:

```
public boolean equals(ICategory book) {
    if(twardaOprawa == false)return true;
    return (this.twardaOprawa &&  book.getBoolianProperie( propertieNumber: 0));
}
@Override
public void setBoolianPropertie(Boolean b, int propertieNumber) { if(propertieNumber == 0)this.twardaOprawa =b; }

@Override
public boolean getBoolianProperie(int propertieNumber) {
    if(propertieNumber == 0)return this.twardaOprawa;
    return false;
}
```

Klasa Food:

```
@Override
public void setBoolianPropertie(Boolean b, int propertieNumber) {
}

@Override
public boolean getBoolianProperie(int propertieNumber) {
    return false;
}

@Override
public boolean equals(ICategory c) {
    return true;
}
```

Muzyka

```
@Override
public void setBoolianPropertie(Boolean b, int propertieNumber) {
    if(propertieNumber == 0)this.video = b;
}

@Override
public boolean getBoolianProperie(int propertieNumber) {
    return video;
}

@Override
public boolean equals(ICategory c) {
    if(!video)return true;
    return c.getBoolianProperie( propertieNumber: 0) == this.video;
}
```

Sport:

```
@Override
public void setBoolianPropertie(Boolean b, int propertieNumber) {

}

@Override
public boolean getBoolianProperie(int propertieNumber) {
    return false;
}

@Override
public boolean equals(ICategory c) {
    return true;
}
```

Electronics:

```
public void setBoolianPropertie(Boolean b, int propertieNumber) {
    if(propertieNumber == 0)this.mobilny = b;
    if(propertieNumber == 1)this.gwarancja = b;
}

@Override
public boolean getBoolianProperie(int propertieNumber) {
    if(propertieNumber == 0)return this.mobilny;
    if(propertieNumber == 1)return this.gwarancja;
    return false;
}

@Override
public boolean equals(ICategory c) {
    if(!(mobilny || gwarancja))return true;
    if(mobilny && gwarancja)return mobilny == c.getBoolianProperie( propertieNumber: 0) && gwarancja == c.getBoolianProperie( propertieNumber: 1);
    if(!mobilny)return c.getBoolianProperie( propertieNumber: 1);
    if(!gwarancja)return c.getBoolianProperie( propertieNumber: 0);
    return  false;

}
```

Dodanie do funkcji **fillProperties** w klasie **PropertiesPanel** czesci odpowiedzialej za generowanie checkboxów dla każdej z klas.

```
switch (shopController.getCurrentCategory()) {
    case FOOD:
        filter.setICategory(new Food( dataPrzydatnosci: "-"));
        break;
    case BOOKS:
        filter.setICategory( new Book( liczbaStron: 0, twardaOprawa: false));
        add(createPropertyCheckbox( propertyName: "twarda Oprawa", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                filter.getICategory().setBoolianPropertie(
                        ((JCheckBox) event.getSource()).isSelected(), propertieNumber: 0);
                shopController.filterItems(filter);
            }
        }));
        break;
    case MUSIC:
        filter.setICategory(new Muzyka( video: false,GatunekMuzyki.HIPHOP));
        add(createPropertyCheckbox( propertyName: "film video", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                filter.getICategory().setBoolianPropertie(
                        ((JCheckBox) event.getSource()).isSelected(), propertieNumber: 0);
                shopController.filterItems(filter);
            }
        }));
        break;
```

```
    case SPORT:
        filter.setICategory(new Sport());
        break;
    case ELECTRONICS:
        filter.setICategory(new Elecrtronics( mobilny: false, gwarancja: false));
        add(createPropertyCheckbox( propertyName: "mobilny", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                filter.getICategory().setBoolianPropertie(
                        ((JCheckBox) event.getSource()).isSelected(), propertieNumber: 0);
                shopController.filterItems(filter);
            }
        }));
        add(createPropertyCheckbox( propertyName: "gwarancja", new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent event) {
                filter.getICategory().setBoolianPropertie(
                        ((JCheckBox) event.getSource()).isSelected(), propertieNumber: 1);
                shopController.filterItems(filter);
            }
        }));
        break;

}
```

Widok po wprowadzeniu zmian:

Dronka Shop

- [ ] Tanie bo polskie
- [ ] Używany
- [x] mobilny
- [ ] gwarancja

Legendarny Bulbulator

Dronka Shop

- [ ] Tanie bo polskie
- [ ] Używany
- [x] twarda Oprawa

Nowe Przygody Gangu Czworga

When The Smoke is Going Down : Wprowadzenie do testów wydajnościowych

Dronka Shop

- [ ] Tanie bo polskie
- [ ] Używany
- [x] film video

Ciley Myrus - Big Ball of Mud