Wojciech Kosztyła
nr. indeksu: 305 338

Projektowanie obiektowe

Laboratorium 2

Wprowadzanie zmian w istniejącej aplikacji

2.1 Dodatkowe właściwości dla produktów w poszczególnych kategoriach

1. Rozpocząłem od stworzenia klas dziedziczących od klasy „Item”:

    a. klasę „Book”

1.
```java
package pl.edu.agh.dronka.shop.model;

public class Book extends Item {

    private int howManyPages;
    private boolean hardCover;

    // Nazwa, Cena, Ilość, Tanie bo polskie, Używany, Liczba stron, Twarda oprawa
    public Book(String name, Category category, int price, int quantity, int howManyPages, boolean hardCover){
        super(name, category, price, quantity);
        this.howManyPages = howManyPages;
        this.hardCover = hardCover;
    }

    public int getHowManyPages(){ return this.howManyPages; }

    public boolean hasHardCover(){ return this.hardCover; }
}
```

    b. klasę „Electronic”

1.
```java
package pl.edu.agh.dronka.shop.model;

public class Electronic extends Item {

    private boolean isMobile;
    private boolean hasGuarantee;

    // Nazwa, Cena, Ilość, Tanie bo polskie, Używany, Mobilny, Gwarancja
    public Electronic(String name, Category category, int price, int quantity, boolean isMobile, boolean hasGuarantee){
        super(name, category, price, quantity);
        this.isMobile = isMobile;
        this.hasGuarantee = hasGuarantee;
    }

    public boolean isMobile() { return this.isMobile; }

    public boolean hasGuarantee() { return this.hasGuarantee; }
}
```

c. klasę „Food”

1.
```java
package pl.edu.agh.dronka.shop.model;

import java.util.Date;

public class Food extends Item {

    private Date expireDate;

    // Nazwa, Cena, Ilość, Tanie bo polskie, Używany, DataPrzydatności
    public Food(String name, Category category, int price, int quantity, Date expireDate){
        super(name, category, price, quantity);
        this.expireDate = expireDate;
    }

    public Date getExpireDate(){ return this.expireDate; }
}
```

ii. Dodałem w pliku „resource/food.csv” kategorię „DataPrzydatności”

d. klasę „Music”

1.
```java
package pl.edu.agh.dronka.shop.model;

public class Music extends Item {

    private GatunekMuzyczny genre;
    private boolean hasVideo;

    // Nazwa, Cena, Ilość, Tanie bo polskie, Używany, Gatunek muzyczny, zTeledyskiem
    public Music(String name, Category category, int price, int quantity, GatunekMuzyczny genre, boolean hasVideo){
        super(name, category, price, quantity);
        this.genre = genre;
        this.hasVideo = hasVideo;
    }

    public GatunekMuzyczny getGenre() { return this.genre; }

    public boolean hasVideo() { return this.hasVideo; }
}
```

ii. Dodałem „enum” „Gatunek muzyczny”, a w nim parę przykładowych gatunków muzycznych.

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

public enum GatunekMuzyczny {
    POP, ELECTRONIC, CLASSICAL, RAP, ROCK, METAL
}
```

iii. Dodałem w pliku „resource/music.csv” kategorie „Gatunek muzyczny” i „zTeledyskiem”.

e. klasę „Sport”

1.
```java
package pl.edu.agh.dronka.shop.model;

public class Sport extends Item {

    public Sport(String name, Category category, int price, int quantity){
        super(name, category, price, quantity);
    }
}
```

2. Aby produkty były parsowane jako obiekty tych klas, zmieniłem metodę „readItems” w „ShopProvider”, tak aby w zależności od kategorii w której znajduje się produkt, wczytało dodatkowe wartości z plików w „resource”, a następnie użyło odpowiedniego konstruktora.

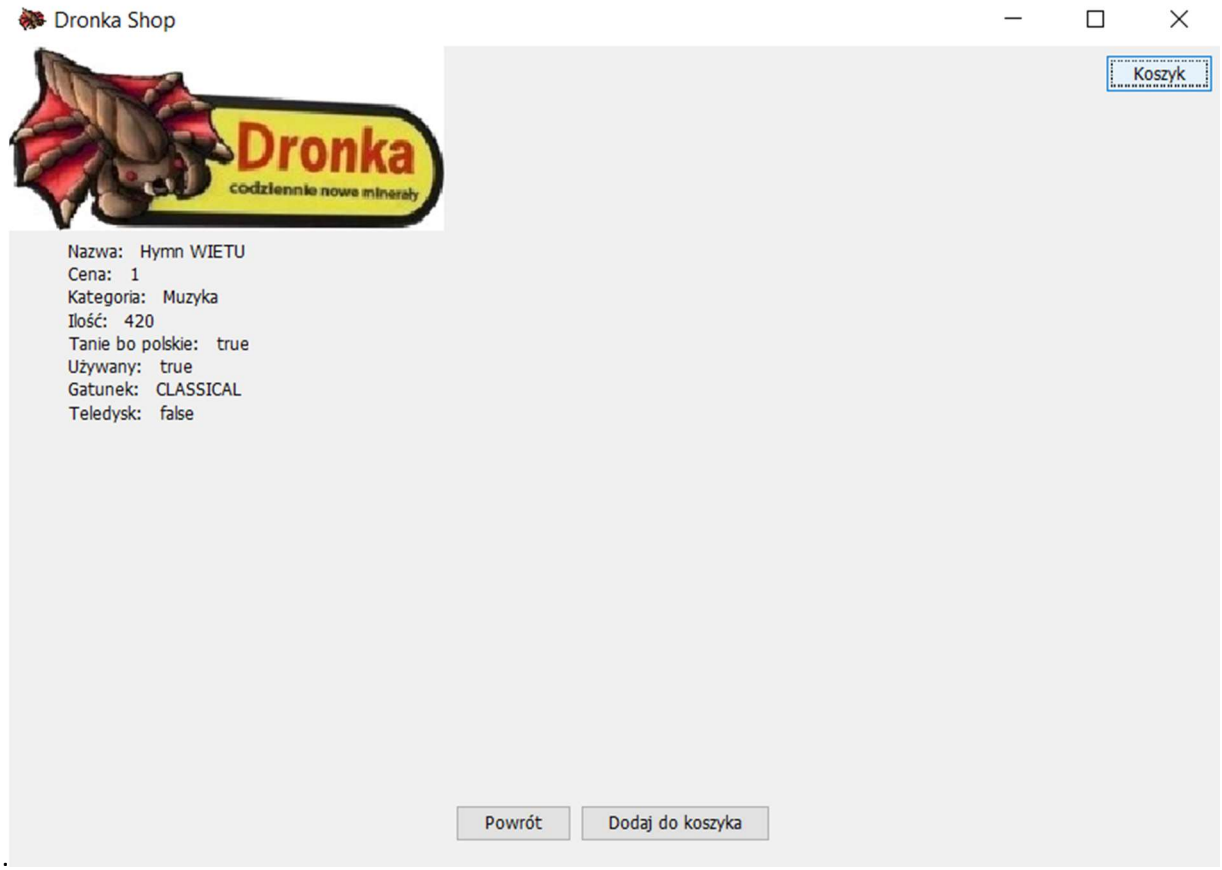
1.
```
boolean isPolish = Boolean.parseBoolean(reader.getValue(
        dataLine,  name: "Tanie bo polskie"));
boolean isSecondhand = Boolean.parseBoolean(reader.getValue(
        dataLine,  name: "Używany"));

/* My part */
Item item;
switch(category){
    case BOOKS: // Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Liczba stron,Twarda oprawa
        int howManyPages = Integer.parseInt(reader.getValue(dataLine,  name: "Liczba stron"));
        boolean hasHardCover = Boolean.parseBoolean(reader.getValue(dataLine,  name: "Twarda oprawa"));
        item = new Book(name, category, price, quantity, howManyPages, hasHardCover);
        break;

    case ELECTRONICS: // Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Mobilny,Gwarancja
        boolean isMobile = Boolean.parseBoolean(reader.getValue(dataLine,  name: "Mobilny"));
        boolean hasGuarantee = Boolean.parseBoolean(reader.getValue(dataLine,  name: "Gwarancja"));
        item = new Electronic(name, category, price, quantity, isMobile, hasGuarantee);
        break;

    case FOOD: // Nazwa,Cena,Ilość,Tanie bo polskie,Używany,DataPrzydatności
        String dateInString = reader.getValue(dataLine,  name: "DataPrzydatności");
        SimpleDateFormat tempParser = new SimpleDateFormat( pattern: "dd-MM-yyyy");
        Date expireDate = tempParser.parse(dateInString);
        item = new Food(name, category, price, quantity, expireDate);
        break;
```

2.
```
    case MUSIC: // Nazwa,Cena,Ilość,Tanie bo polskie,Używany,Gatunek muzyczny,zTeledyskiem
        GatunekMuzyczny genre = GatunekMuzyczny.valueOf(reader.getValue(dataLine,  name: "Gatunek muzyczny"));
        boolean hasVideo = Boolean.parseBoolean(reader.getValue(dataLine,  name: "zTeledyskiem"));
        item = new Music(name, category, price, quantity, genre, hasVideo);
        break;

    case SPORT: // Nazwa,Cena,Ilość,Tanie bo polskie,Używany
        item = new Sport(name, category, price, quantity);
        break;

    default: // Nazwa,Cena,Ilość,Tanie bo polskie,Używany
        item = new Item(name, category, price, quantity);
        break;
}
/* End of my part */


item.setPolish(isPolish);
item.setSecondhand(isSecondhand);

items.add(item);
```

3. Aby dodatkowe parametry produktów wyświetlić, dodałem kod w „PropertiesHelper”.

1.
```
    propertiesMap.put( k: "Ilość", Integer.toString(item.getQuantity()));
    propertiesMap.put( k: "Tanie bo polskie", item.isPolish());
    propertiesMap.put( k: "Używany", item.isSecondhand());

    /* Moja część */
    if (item instanceof Book){
        propertiesMap.put( k: "Liczba stron", ((Book) item).getHowManyPages());
        propertiesMap.put( k: "Twarda oprawa", ((Book) item).hasHardCover());
    }
    else if(item instanceof Electronic){
        propertiesMap.put( k: "Mobilne", ((Electronic) item).isMobile());
        propertiesMap.put( k: "Gwarancja", ((Electronic) item).hasGuarantee());
    }
    else if(item instanceof Food){
        propertiesMap.put( k: "Data ważności", ((Food) item).getExpireDate().toString());
    }
    else if(item instanceof Music){
        propertiesMap.put( k: "Gatunek", ((Music) item).getGenre());
        propertiesMap.put( k: "Teledysk", ((Music) item).hasVideo());
    }
    /*else if(item instanceof Sport){
    }*/
    /* Koniec mojej części */

    return propertiesMap;
}
```

4. Screenshot, który udowadnia poprawne wyświetlanie dodatkowej kategorii:
    i. (widać nowe pola „Gatunek” i „Teledysk”)

1. 

2.2 Rozszerzenie panelu

1. Należy zależnie od kategorii produktu wyświetlić dodatkowe opcje filtrowania. Rozpocząłem od modyfikacji „appliesTo” w klasie „ItemFilter”:

1.

```
54
55          // applies filter only if the flag (polish) is true)
56          if (itemSpec.isPolish() && !item.isPolish()) {
57              return false;
58          }
59
60
61          /* Moja część */
62          if(itemSpec instanceof Book){
63              if( ((Book) itemSpec).hasHardCover() && !((Book) item).hasHardCover() ){
64                  return false;
65              }
66          }
67
68          if(itemSpec instanceof Electronic){
69              if( ((Electronic) itemSpec).isMobile() && !((Electronic) item).isMobile() ){
70                  return false;
71              }
72              if( ((Electronic) itemSpec).hasGuarantee() && !((Electronic) item).hasGuarantee() ){
73                  return false;
74              }
75          }
76
77          if(itemSpec instanceof Music){
78              if( ((Music) itemSpec).hasVideo() && !((Music) item).hasVideo() ){
79                  return false;
80              }
81          }
82
83          /* Koniec mojej części */
84
```

2. Zauważyłem, że aby odwołać się do odpowiednich metod, potrzebuję do wszystkich klas pochodnych od „Item” dodać „pusty” konstruktor `public Book(){}`.

    a. Zatem w zależności od kategorii, „itemSpec” będzie instancją odpowiedniej klasy.

```
private Item itemSpec;
private Category category;

public ItemFilter(Category category){
    this.category = category;
    switch(category){
        case BOOKS:
            itemSpec = new Book();
            break;
        case ELECTRONICS:
            itemSpec = new Electronic();
            break;
        case FOOD:
            itemSpec = new Food();
            break;
        case MUSIC:
            itemSpec = new Music();
            break;
        case SPORT:
            itemSpec = new Sport();
            break;
        default:
            itemSpec = new Item();
            break;
    }
}
```

3. W „PropertiesPanel” zmieniłem „filter” tak, by został tworzony przy każdym użyciu „fillProperties”.

1.
```
private ItemFilter filter;
```

2.
```
29      public void fillProperties() {
30          removeAll();
31
32          this.filter = new ItemFilter(shopController.getCurrentCategory());
33          filter.getItemSpec().setCategory(shopController.getCurrentCategory());
```

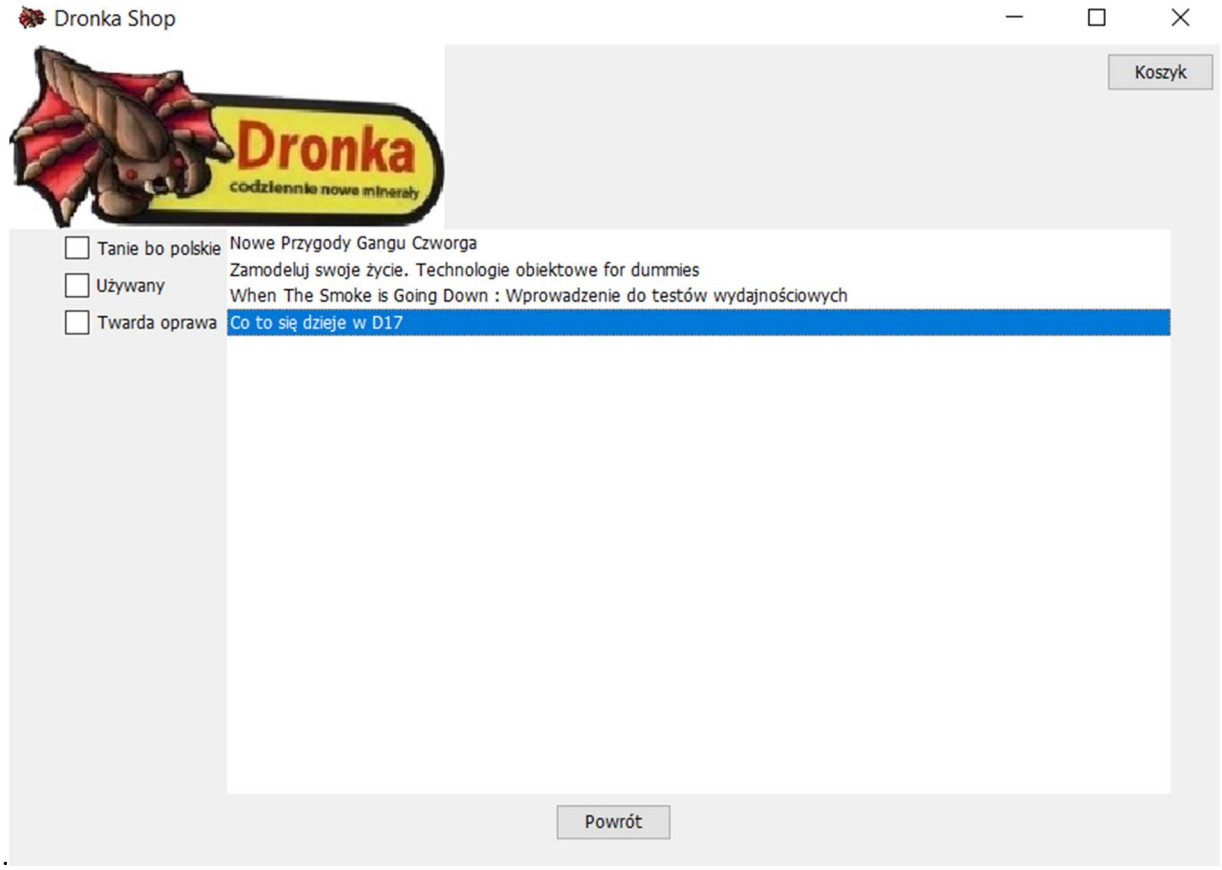
4. W „PropertiesPanel” zmieniłem „fillProperties”, tak aby w zależności od kategorii tworzył dodatkowe okienka wyboru (tzw. „ptaszki”).

1.
```
            @Override
            public void actionPerformed(ActionEvent event) {
                filter.getItemSpec().setSecondhand(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }
        }));

        /* Moja część */
        if(shopController.getCurrentCategory() == Category.BOOKS){
            add(createPropertyCheckbox( propertyName: "Twarda oprawa", new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent event) {
                    ((Book) filter.getItemSpec()).setHardCover(
                            ((JCheckBox) event.getSource()).isSelected());
                    shopController.filterItems(filter);
                }
            }));
        }

        if(shopController.getCurrentCategory() == Category.ELECTRONICS){
            add(createPropertyCheckbox( propertyName: "Mobilne", new ActionListener() {

                @Override
                public void actionPerformed(ActionEvent event) {
                    ((Electronic) filter.getItemSpec()).setIsMobile(
                            ((JCheckBox) event.getSource()).isSelected());
                    shopController.filterItems(filter);
                }
            }));
```

2.

```
78              add(createPropertyCheckbox( propertyName: "Gwarancja", new ActionListener() {
79
80                  @Override
81                  public void actionPerformed(ActionEvent event) {
82                      ((Electronic) filter.getItemSpec()).setHasGuarantee(
83                              ((JCheckBox) event.getSource()).isSelected());
84                      shopController.filterItems(filter);
85                  }
86              }));
87
88
89          }
90
91          if(shopController.getCurrentCategory() == Category.MUSIC){
92              add(createPropertyCheckbox( propertyName: "Teledysk", new ActionListener() {
93
94                  @Override
95                  public void actionPerformed(ActionEvent event) {
96                      ((Music) filter.getItemSpec()).setHasVideo(
97                              ((JCheckBox) event.getSource()).isSelected());
98                      shopController.filterItems(filter);
99                  }
100             }));
101         }
102         /* Koniec mojej części */
103
104     }
```

5. Zamieszczam zdjęcia, jak dowód, że wprowadzone przeze mnie modyfikacje działają.

1. 

2.