Nazar Kordiumov

Laboratorium 2

Wprowadzanie zmian w istniejącej aplikacji

Zadanie 2.1

Dodatkowe właściwości dla produktów w poszczególnych kategoriach

1. Najpierw zmieniłem model danych. Klasę *Item* zmieniłem na klasę abstrakcyjną i stworzyłem 5 nowych klas dla każdej kategorii dziedziczących po klasie Item. Przykładowa klasa *BookItem*.

```java
public class BookItem extends Item {

    private int numberOfPages;
    private boolean isHardCover;

    public BookItem(String name, Category category, int price, int quantity, int numberOfPages, boolean isHardCover) {
        super(name, category, price, quantity);
        this.numberOfPages = numberOfPages;
        this.isHardCover = isHardCover;
    }

    public BookItem() {
    }

    public int getNumberOfPages() { return numberOfPages; }

    public void setNumberOfPages(int numberOfPages) { this.numberOfPages = numberOfPages; }

    public boolean isHardCover() { return isHardCover; }

    public void setHardCover(boolean hardCover) { isHardCover = hardCover; }

    @Override
    public Map<String, Object> getAdditionalPropertiesMap() {
        return new HashMap<String, Object>() {{
            put("Liczba stron", numberOfPages);
            put("Twarda oprawa", isHardCover);
        }};
    }

    @Override
    public boolean isAdditionalFieldsAppliedTo(FilterSpec filter) { return !filter.isHardCover() || isHardCover; }
}
```

2. Wczytywanie danych
    2.1. Następnie stworzyłem interfejs *Provider*, który ma jedną metodę.

```java
public interface Provider {

    Item createItem(String name, Category category, int price, int quantity, String[] dataLine, CSVReader reader);

}
```

2.2. Stworzyłem 5 klas implementujących ten interfejs zwracających konkretny rodzaj *Item* oraz fabrykę *ProviderFactory*, która zwraca konkretny provider w zależności od kategorii. Przykładowa klasa *BookProvider*.

```
public class BookProvider implements Provider {

    @Override
    public Item createItem(String name, Category category, int price, int quantity, String[] dataLine, CSVReader reader) {
        int numberOfPages  = Integer.parseInt(reader.getValue(dataLine,  name: "Liczba stron"));
        boolean isHardCover = Boolean.parseBoolean(reader.getValue(
                dataLine,  name: "Twarda oprawa"));
        return new BookItem(name, category, price, quantity, numberOfPages, isHardCover);
    }
}
```

2.3. Dodane 3 linijki do klasy *ShopProvider* gwarantują poprawne wczytanie nowych danych

```
                Provider provider = ProviderFactory.getProvider(category);

                Item item = provider.createItem(name, category, price, quantity, dataLine, reader);
                item.setPolish(isPolish);
                item.setSecondHand(isSecondhand);
```

3. Wyświetlanie szczegółów produktów. Metodę *getPropertiesMap(Item item)* w klasie *PropertiesHelper* zmieniłem tak, aby korzystała z metody *getAdditionalPropertiesMap()*, która zwraca specyficzne pola dla każdego rodzaju Item'a (pierwszy screen).

```
public class PropertiesHelper {

    public static Map<String, Object> getPropertiesMap(Item item) {
        Map<String, Object> propertiesMap = new LinkedHashMap<>();

        propertiesMap.put("Nazwa", item.getName());
        propertiesMap.put("Cena", item.getPrice());
        propertiesMap.put("Kategoria", item.getCategory().getDisplayName());
        propertiesMap.put("Ilość", Integer.toString(item.getQuantity()));
        propertiesMap.put("Tanie bo polskie", item.isPolish());
        propertiesMap.put("Używany", item.isSecondHand());
        Map<String, Object> additionalPropertiesMap = item.getAdditionalPropertiesMap();
        if(additionalPropertiesMap != null) {
            propertiesMap.putAll(additionalPropertiesMap);
        }

        return propertiesMap;
    }
}
```

Zadanie 2.2

Rozszerzenie panelu

1. Dodałem klasę *FilterSpec*, która ma wszystkie atrybuty typu *boolean*, czyli atrybuty według, których możemy sortować Item'y.

```
public class FilterSpec {

    private Category category;
    private boolean isPolish;
    private boolean isSecondHand;
    private boolean isHardCover;
    private boolean isMobile;
    private boolean isUnderWarranty;
    private boolean isWithVideo;

    public Category getCategory() {
        return category;
    }

    public void setCategory(Category category) {
        this.category = category;
    }

    public boolean isPolish() {
```

2. Klasa *ItemFilter* ma teraz *FilterSpec*, jako atrybut zamiast *Item*. Metodę *appliesTo(Item item)* zmieniłem w taki sposób, aby wywoływana metodę *isAdditionalFieldsAppliesTo(FilterSpec filter)* w klasie *Item* (pierwszy screen), która zwraca wartość *boolean* w zależności od tego czy dodatkowe pola konkretnego *Item'a* pasują do filtrów.

```
public class ItemFilter {

    private FilterSpec filter = new FilterSpec();

    public FilterSpec getFilter() {
        return filter;
    }

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

        if (filter.getCategory() != null
                && !filter.getCategory().equals(item.getCategory())) {
            return false;
        }

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

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

        return item.isAdditionalFieldsAppliedTo(filter);
    }
}
```

3. Ostatecznie dodałem specyficzne pola *boolean* jako checkbox'y w klasie *PropertiesPanel* za pomocą switch'a według kategorii.

```
public void fillProperties(Category category) {
    removeAll();

    filter.getFilter().setCategory(shopController.getCurrentCategory());

    add(createPropertyCheckbox( propertyName: "Tanie bo polskie", event -> {
        filter.getFilter().setPolish(
                ((JCheckBox) event.getSource()).isSelected());
        shopController.filterItems(filter);
    }));

    add(createPropertyCheckbox( propertyName: "Używany", event -> {
        filter.getFilter().setSecondHand(
                ((JCheckBox) event.getSource()).isSelected());
        shopController.filterItems(filter);
    }));

    switch (category) {
        case BOOKS: {
            add(createPropertyCheckbox( propertyName: "Twarda oprawa", event -> {
                filter.getFilter().setHardCover(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }));
            break;
        }
        case ELECTRONICS: {
            add(createPropertyCheckbox( propertyName: "Mobilny", event -> {
                filter.getFilter().setMobile(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }));

            add(createPropertyCheckbox( propertyName: "Gwarancja", event -> {
                filter.getFilter().setUnderWarranty(
                        ((JCheckBox) event.getSource()).isSelected());
                shopController.filterItems(filter);
            }));
            break;
        }
        case FOOD:
```