INF.04 – Zadanie praktyczne (projekt): Aplikacja desktopowa "Wypożyczalnia Narzędzi" w Java Swing

Niniejszy dokument opisuje pełne wymagania zadania praktycznego przeznaczonego dla klasy 4 technik programista. Zadanie zostało przygotowane w stylu zadań egzaminacyjnych INF.04 i dotyczy stworzenia aplikacji desktopowej w bibliotece Java Swing. Aplikacja służy do zarządzania katalogiem narzędzi, ich filtrowania, sortowania oraz zapisu/odczytu danych w formacie CSV.

1. Temat i cel zadania

Aplikacja desktopowa "Wypożyczalnia Narzędzi" – zarządzanie katalogiem narzędzi oraz zapis/odczyt danych z pliku CSV.

Cel: przygotowanie kompletnej aplikacji (logika oraz interfejs) z poprawną walidacją danych, obsługą błędów i przejrzystym UI.

2. Środowisko i technologie

- Język programowania: Java 17+
- Interfejs użytkownika: Java Swing
- Budowanie: Maven lub Gradle (opcjonalnie możliwa kompilacja ręczna)
- Format danych: CSV (obowiązkowo), JSON (opcjonalnie, za punkty dodatkowe)
- Rekomendowana architektura: podział na warstwy (model, service, ui) lub wzorzec MVC

3. Struktura danych i format pliku

Rekord narzędzia (pola):

- id – liczba całkowita, unikalna (np. rosnąca)
- nazwa – tekst (min. 2 znaki)
- kategoria – tekst (np. Elektryczne, Ręczne, Pomiarowe)
- cenaZaDzien – liczba zmiennoprzecinkowa (PLN)
- iloscDostepna – liczba całkowita (≥ 0)
- status – tekst: AKTYWNE lub WYCOFANE

Format CSV (separator: średnik ';', kropka w liczbach zmiennoprzecinkowych):

Nagłówki: id;nazwa;kategoria;cenaZaDzien;iloscDostepna;status
Przykładowe rekordy (dołącz osobno jako plik wejściowy):
1;Wiertarka udarowa;Elektryczne;29.99;3;AKTYWNE
2;Młotek 500g;Ręczne;9.50;10;AKTYWNE
3;Suwmiarka 150mm;Pomiarowe;12.00;0;WYCOFANE

4. Wymagania funkcjonalne (obowiązkowe)

- Główne okno (JFrame) zawierające tabelę (JTable) z danymi oraz menu/pasek narzędzi z akcjami: Wczytaj CSV, Zapisz CSV, Dodaj, Edytuj, Usuń, Filtruj, Wyczyść filtr, Sortuj.
- Operacje CRUD: dodawanie, edycja, usuwanie. Formularz w postaci okna dialogowego (JDialog/JOptionPane) z walidacją pól.
- Wczytywanie i zapis CSV: wczytanie czyści aktualne dane i ładuje poprawne wiersze, błędne są pomijane z komunikatem. Zapis nadpisuje lub tworzy wskazany plik.

- Filtrowanie po frazie w nazwie (niewrażliwe na wielkość liter) oraz po kategorii; sortowanie po cenie i ilości.
- Obsługa błędów: aplikacja nie kończy działania przy błędach danych, lecz prezentuje komunikaty użytkownikowi.
- Dokumentacja użytkownika (PDF/DOCX): krótki opis uruchomienia, funkcji, formatu CSV i znanych ograniczeń.

5. Wymagania dodatkowe (do +10% punktów)

- Eksport/Import JSON.
- Filtrowanie wielokryterialne (np. zakres ceny od–do).
- Eksport raportu (np. lista aktywnych narzędzi z dostępną ilością > 0) do CSV.
- Zapis ustawień (ostatnia ścieżka pliku, rozmiar okna) z wykorzystaniem Preferences.

6. Wymagania niefunkcjonalne

- Czytelny interfejs: spójne etykiety, układ komponentów (np. BorderLayout, GridBagLayout).
- Rozdział odpowiedzialności: warstwy/model, brak logiki biznesowej w warstwie UI.
- Komentarze i czytelne nazwy klas/metod.
- README.md (jak uruchomić, wersja JDK).

7. Interfejs użytkownika – układ i funkcje

Główne okno powinno zawierać: (1) menu aplikacji z grupami 'Plik' (Wczytaj CSV, Zapisz CSV) oraz 'Akcje' (Dodaj, Edytuj, Usuń); (2) pasek narzędzi z szybkimi przyciskami do tych samych akcji; (3) panel filtrów z polem "Fraza" oraz listą "Kategoria" i przyciskami "Filtruj"/"Wyczyść filtr"; (4) tabelę prezentującą dane (kolumny: ID, Nazwa, Kategoria, Cena/dzień, Ilość, Status) z możliwością sortowania; (5) panel sortowania (pole sortowania + kierunek); (6) pasek statusu informujący o bieżących operacjach.

Makieta (schemat) widoku:

```
+-------------------------------------------------------------------------------+
| Menu: Plik [Wczytaj CSV, Zapisz CSV] Akcje [Dodaj, Edytuj, Usuń] |
+-------------------------------------------------------------------------------+
| [Fraza: _____] [Kategoria: v Wszystkie] [Filtruj] [Wyczyść filtr] |
+-------------------------------------------------------------------------------+
| ID | Nazwa | Kategoria | Cena/dzień | Ilość | Status |
|----+----------------------+------------+-----------+-------+-----------------|
| 1 | Wiertarka udarowa | Elektryczne | 29.99 | 3 | AKTYWNE |
| 2 | Młotek 500g | Ręczne | 9.50 | 10 | AKTYWNE |
| 3 | Suwmiarka 150mm | Pomiarowe | 12.00 | 0 | WYCOFANE |
| ... |
+-------------------------------------------------------------------------------+
| Sortuj wg: [Nazwa|Cena|Ilość] Kierunek: (• Rosnąco ○ Malejąco) [Zastosuj] |
+-------------------------------------------------------------------------------+
| Status: gotowy |
+-------------------------------------------------------------------------------+
```

8. Walidacja danych

- nazwa: co najmniej 2 znaki (niepuste)
- cenaZaDzien: > 0 (dopuszczalne wartości z kropką jako separatorem dziesiętnym)
- iloscDostepna: liczba całkowita ≥ 0

- status: tylko wartości AKTYWNE lub WYCOFANE

9. Scenariusze testowe (egzaminacyjne)

- Wczytanie przykładowego pliku CSV – oczekiwane 3 rekordy; 'Suwmiarka 150mm' ma status WYCOFANE i ilość 0.
- Dodanie nowego narzędzia (np. Szlifierka kątowa) – poprawna walidacja i pojawienie się rekordu w tabeli.
- Walidacja – próba dodania nazwy 'X' i ceny -1 kończy się komunikatem o błędzie i brakiem zapisu.
- Edycja – zmiana ilości dla 'Młotek 500g' na 7; wynik widoczny w tabeli.
- Filtrowanie – fraza 'wier' oraz kategoria 'Elektryczne' pozostawia 'Wiertarka udarowa'.
- Zapis i ponowne wczytanie – zawartość spójna z tabelą przed zapisem.

10. Kryteria oceniania (100 pkt)

I. Projekt i model danych – 15 pkt

- Klasa modelu z polami i walidacją (7)
- Unikalne ID i zarządzanie nim (4)
- Czytelna struktura projektu (4)

II. Interfejs użytkownika – 25 pkt

- Tabela i formularze działające poprawnie (10)
- Menu/pasek narzędzi i akcje (8)
- Komunikaty błędów/confirm (7)

III. Logika biznesowa – 20 pkt

- Dodawanie/edycja/usuwanie z walidacją (10)
- Filtrowanie + sortowanie (10)

IV. Import/Eksport CSV – 20 pkt

- Poprawne wczytanie z obsługą błędnych wierszy (10)
- Poprawny zapis i spójność danych (10)

V. Jakość kodu i dokumentacja – 20 pkt

- Czytelność, komentarze, nazewnictwo (10)
- README + instrukcja użytkownika (10)

Bonus – do +10 pkt

- JSON, raporty, preferences

Minimalny próg zaliczenia: 60 pkt.

11. Wymagane rezultaty od ucznia

- Spakowany projekt źródłowy (.zip) wraz z README.md.
- Dokument "instrukcja_uzytkownika.pdf" (lub .docx).
- Działający plik .jar (opcjonalnie, jeśli użyto Maven/Gradle).
- Zrzuty ekranu: tabela, formularz dodawania, filtr, zapis/odczyt CSV.

12. Wskazówki techniczne (bez kodu)

- Do prezentacji danych użyj JTable z własnym modelem tabeli (rozszerzenie AbstractTableModel) – ułatwia kontrolę walidacji i formatowania.
- Formularze realizuj w JDialogach; waliduj dane po stronie formularza i serwisu (walidacja dwuetapowa).
- Przy CSV pamiętaj o pomijaniu pustych linii, nagłówku oraz komunikatach dla błędnych wierszy.
- Sortowanie i filtrowanie możesz zrealizować przez TableRowSorter oraz RowFilter, aby nie modyfikować danych źródłowych.
- Generator ID powinien być resetowany do wartości maksymalnej po imporcie (np. po wczytaniu pliku CSV).
- Obsługuj wyjątki i przekazuj czytelne komunikaty do użytkownika (np. przez JOptionPane).

13. Materiały do wydania uczniowi (pakiet egzaminacyjny)

- Plik wejściowy narzedzia.csv (z podanymi nagłówkami i przykładowymi danymi).
- Treść zadania (niniejszy dokument PDF).
- Ewentualny szkielet projektu – bez gotowych rozwiązań kluczowych fragmentów.
- Kryteria ocen – jawne.

14. Uwagi organizacyjne i zalecenia

- Zadanie powinno być wykonane samodzielnie, z zachowaniem dobrych praktyk programistycznych.
- Dostarcz wszystkie wymagane elementy w postaci archiwum .zip.
- Upewnij się, że ścieżki do plików CSV nie zawierają polskich znaków w nazwach katalogów (rekomendacja dla uniknięcia problemów środowiskowych).
- Nazwy interfejsu (etykiety, kolumny) powinny być w języku polskim i spójne z treścią zadania.

Uwaga: Dokument nie zawiera kodu źródłowego Java – uczniowie implementują rozwiązanie samodzielnie na podstawie wymagań.