ZADANIA DOMOWE – INF.04

Termin oddania: 7 stycznia 2026 r.

Ważne informacje

- **Termin wykonania:** 7 stycznia 2026 roku
- **Forma oddania:** Przyniesienie rozwiązań na nośniku USB/pendrive albo z chmury do oceny
- **Dozwolone języki:**
  - Aplikacje konsolowe: C# lub Python
  - Aplikacje mobilne: Java (Android)
  - Aplikacje desktopowe: C# WPF
- **Struktura folderów:** `konsolowa1/`, `konsolowa2/`, `mobilna1/`, `mobilna2/`, `desktop1/`, `desktop2/`
- **Zawartość folderów:** Tylko pliki źródłowe `.cs`, `.py`, `.java`, `.xml`.
- **Całość spakowana** Proszę wszystkie powyższe foldery spakować do archiwum o nazwie numer zdającego (może być hasło: egzamin)

Wymagany raport do każdego zadania

Do każdego z 6 zadań należy dołączyć plik `RAPORT.txt` odpowiadający na następujące pytania:

1. Ile czasu zajęło Ci zrobienie tego zadania?
2. Jaki procent pomocy otrzymałeś z zewnątrz (internet, AI, kolega)?
3. Czego nauczyłeś się wykonując to zadanie?
   Konkretnie podać: np pętli for, class w c sharp, int w java itp.
4. Jak oceniasz swój wynik procentowy z tego zadania na egzaminie (od 0 do 100), gdyby dokładnie takie pojawiło się na egzaminie:
   - z perspektywy dnia dzisiejszego jak zacząłeś to rozwiązywać (np 16 grudnia 2025)?
   - z perspektywy 7 stycznia 2026 (po zrobieniu zadania)?

Zadanie 1. Aplikacja konsolowa – Sortowanie według sumy cyfr

Działanie programu:

1. Program pobiera od użytkownika liczbę elementów n (zakres 3–20).
2. Program wczytuje n liczb całkowitych dodatnich (zakres 1–9999).
3. Program oblicza sumę cyfr dla każdej liczby.
4. Program sortuje liczby **metodą bąbelkową** według sumy cyfr (rosnąco).
5. Jeśli sumy cyfr są równe, kolejność tych liczb pozostaje niezmieniona.
6. Program wyświetla listę liczb przed sortowaniem, listę z sumami cyfr i listę po sortowaniu.

Przykład działania:

```
=== SORTOWANIE WEDŁUG SUMY CYFR ===

Podaj liczbę elementów (3-20): 2
Błąd! Liczba musi być z zakresu 3-20.
Podaj liczbę elementów (3-20): 5

Podaj liczbę 1: 123
Podaj liczbę 2: 45
Podaj liczbę 3: 789
Podaj liczbę 4: 12
Podaj liczbę 5: 100

Tablica przed sortowaniem: 123, 45, 789, 12, 100

Liczby z sumami cyfr:
123 (suma: 6), 45 (suma: 9), 789 (suma: 24), 12 (suma: 3), 100 (suma: 1)

Tablica po sortowaniu: 100, 12, 123, 45, 789

Czy chcesz posortować kolejną tablicę? (t/n): n
```

Wymagania:

Kategoria	Wymaganie	Punkty
Walidacja	Zakres liczby elementów (3-20) z pętlą	2
Walidacja	Zakres każdej liczby (1-9999)	2
Funkcje	`ObliczSumeCyfr()` – z komentarzem	4
Funkcje	`SortowanieBabelkowe()` – z komentarzem	4
Funkcje	`WyswietlTablice()` – z komentarzem	2
Działanie	Poprawne sortowanie i wyświetlanie	2

Zadanie 2. Aplikacja konsolowa – Ciąg Fibonacciego i liczby pierwsze

Działanie programu:

1. Program pobiera od użytkownika liczbę n – ile liczb Fibonacciego wygenerować (zakres 5–30).
2. Program generuje pierwsze n liczb ciągu Fibonacciego (0, 1, 1, 2, 3, 5, 8, 13...).
3. Program sprawdza dla każdej liczby, czy jest liczbą pierwszą.
4. Program wyświetla listę wszystkich liczb z oznaczeniem, które są pierwsze.
5. Program wyświetla statystyki: ile liczb pierwszych znaleziono.

Przykład działania:

```
=== FIBONACCI I LICZBY PIERWSZE ===

Podaj liczbę wyrazów ciągu (5-30): 3
Błąd! Liczba musi być z zakresu 5-30.
Podaj liczbę wyrazów ciągu (5-30): 10

Ciąg Fibonacciego (10 wyrazów):
F(0) = 0
F(1) = 1
F(2) = 1 [PIERWSZA]
F(3) = 2 [PIERWSZA]
F(4) = 3 [PIERWSZA]
F(5) = 5 [PIERWSZA]
F(6) = 8
F(7) = 13 [PIERWSZA]
F(8) = 21
F(9) = 34

=== STATYSTYKI ===
Liczb pierwszych: 5 / 10 (50.00%)

Czy chcesz wygenerować kolejny ciąg? (t/n): n
```

Wymagania:

Kategoria	Wymaganie	Punkty
Walidacja	Zakres liczby wyrazów (5-30) z pętlą	2
Funkcje	`GenerujFibonacci()` – z komentarzem	4
Funkcje	`CzyPierwsza()` – z komentarzem	5
Funkcje	`WyswietlWyniki()` – z komentarzem	2
Algorytm	Poprawne generowanie ciągu Fibonacciego	2
Działanie	Prawidłowe sprawdzanie pierwszości i statystyki	2

Zadanie 3. Aplikacja mobilna – Kalkulator Rabatu

Elementy interfejsu (LinearLayout wertykalny):

1. **EditText** – pole do wpisania ceny produktu (hint: „Cena produktu (zł)")
2. **TextView** – etykieta: „Rabat: X%"
3. **SeekBar** – zakres 0–50 (procent rabatu), wartość domyślna: 10
4. **RadioGroup** z trzema **RadioButton**: Standard (brak dodatkowego rabatu); Premium (+5% rabatu); VIP (+10% rabatu)
5. **CheckBox** – „Karta stałego klienta" (dodatkowe 5%)
6. **Switch** – „Darmowa wysyłka (od 200 zł)"
7. **Button** – „OBLICZ"
8. **ImageView** – obrazek poziomu oszczędności (zmienia się dynamicznie)
9. **TextView** – wynik obliczeń (cena przed, rabat, cena końcowa, info o wysyłce)

Działanie aplikacji:

Walidacja:

- Cena musi być liczbą dodatnią (10-10000 zł).Jeśli cena jest pusta lub nieprawidłowa - wyświetl Toast z komunikatem

Obliczenia:

- Rabat bazowy z SeekBar (0–50%)
- Dodatkowy rabat z RadioButton (0%, 5%, 10%)
- Dodatkowy rabat z CheckBox (+5% jeśli zaznaczony)
- Łączny rabat = suma wszystkich rabatów (max 75%)
- Cena końcowa = cena × (1 - łączny_rabat / 100)

ImageView:

- Rabat 0–20%: obrazek „oszczednosci_male"
- Rabat 21–40%: obrazek „oszczednosci_srednie"
- Rabat 41–75%: obrazek „oszczednosci_duze"

Tło: `#E8F5E9`, Przycisk: `#4CAF50`, Tekst: `#1B5E20`

Wymagania:

Kategoria	Wymaganie	Punkty
UI	Wszystkie kontrolki w LinearLayout	2
Walidacja	Sprawdzanie zakresu ceny + Toast	3
Logika	SeekBar z aktualizacją TextView	2
Logika	RadioGroup z obsługą wyboru	2
Logika	CheckBox i Switch	2
Algorytm	Prawidłowe obliczanie rabatu i ceny	3
ImageView	Dynamiczna zmiana obrazka	2
Kolory	Zdefiniowane w colors.xml	1

Zadanie 4. Aplikacja mobilna – Generator Haseł

Elementy interfejsu (LinearLayout wertykalny):

1. **TextView** – „Długość hasła: X znaków”
2. **SeekBar** – zakres 6–20 znaków, wartość domyślna: 12
3. **CheckBox** (4 elementy w LinearLayout poziomym):
   - Małe litery (a-z)
   - Wielkie litery (A-Z)
   - Cyfry (0-9)
   - Znaki specjalne (!@#$%^&*)
4. **Spinner** – liczba haseł do wygenerowania (1, 2, 3, 4, 5)
5. **RadioGroup** z trzema **RadioButton**:
   - Tryb podstawowy
   - Tryb mieszany
   - Tryb zaawansowany (min. 1 znak każdego typu)
6. **Switch** – „Zapamiętaj ustawienia”
7. **Button** – „GENERUJ”
8. **ImageView** – obrazek siły hasła
9. **TextView** – wygenerowane hasło/hasła (scrollable)

Działanie:

Walidacja:

- Minimum 1 CheckBox zaznaczony – w przeciwnym razie Toast
- Tryb zaawansowany wymaga wszystkich 4 CheckBox

Algorytm oceny siły:

- Punkty = długość hasła + 10 (małe) + 15 (wielkie) + 10 (cyfry) + 20 (spec)
- Suma < 30: słabe, 30–50: średnie, > 50: silne

SharedPreferences: Jeśli Switch włączony – zapisz ustawienia.

Kolory (colors.xml):

- Tło: `#E3F2FD`, Przycisk: `#2196F3`, Tekst: `#0D47A1`

Wymagania:

Kategoria	Wymaganie	Punkty
UI	Wszystkie kontrolki (2 LinearLayout)	2
Walidacja	Sprawdzanie CheckBox + Toast	2
Logika	SeekBar, Spinner, RadioGroup	3
Algorytm	Generowanie losowych haseł	4
Algorytm	Ocena siły hasła (punktacja)	3
ImageView	Dynamiczna zmiana obrazka	2
SharedPreferences	Zapis i odczyt ustawień	1

Zadanie 5. Aplikacja WPF – Formularz Rejestracji

Elementy interfejsu:

GroupBox 1 – „Dane osobowe”:

- TextBox – Imię (wymagane, min. 2 znaki)
- TextBox – Nazwisko (wymagane, min. 2 znaki)
- DatePicker – Data urodzenia (użytkownik musi mieć min. 13 lat)
- Image – obrazek statusu walidacji (zielony OK / czerwony błąd)

GroupBox 2 – „Dane kontaktowe”:

- TextBox – Email (walidacja formatu: xxx@xxx.xxx)
- TextBox – Telefon (9 cyfr, format: XXX-XXX-XXX)
- ComboBox – Województwo (lista 5 województw do wyboru)
- Image – obrazek statusu walidacji

GroupBox 3 – „Hasło i preferencje”:

- PasswordBox – Hasło (min. 8 znaków, musi zawierać cyfrę i wielką literę)
- PasswordBox – Powtórz hasło (musi być identyczne)
- CheckBox – „Akceptuję regulamin” (wymagane)
- CheckBox – „Chcę otrzymywać newsletter”
- Image – obrazek statusu walidacji

Panel dolny:

- Button – „ZAREJESTRUJ” (aktywny tylko gdy wszystkie walidacje OK)
- TextBlock – komunikat o błędach / sukcesie

Działanie aplikacji:

1. Walidacja działa na bieżąco (podczas wpisywania).
2. Każdy GroupBox ma obrazek pokazujący status walidacji swojej sekcji (OK = zielony, błąd = czerwony).
3. Przycisk ZAREJESTRUJ jest aktywny tylko gdy wszystkie pola są poprawne.
4. Po kliknięciu przycisku – wyświetl MessageBox z podsumowaniem danych.

Wymagania:

Kategoria	Wymaganie	Punkty
UI	3 GroupBoxy z kontrolkami	2
Walidacja	Imię i nazwisko (min. 2 znaki)	2
Walidacja	Data urodzenia (min. 13 lat)	2
Walidacja	Email (regex) i telefon (format)	3
Walidacja	Hasło (min. 8, cyfra, wielka) i powtórz	3
Logika	CheckBox regulaminu (wymagany)	2
Obrazki	3 obrazki statusu w GroupBoxach	2
Działanie	Przycisk aktywny tylko gdy wszystko OK	1

Zadanie 6. Aplikacja WPF – Kalkulator Kredytowy

Elementy interfejsu:

GroupBox 1 – „Parametry kredytu”:

- TextBox – Kwota kredytu (10 000 - 1 000 000 zł)
- Slider – Oprocentowanie (0.5% - 20%), wyświetlane w Label
- Slider – Okres kredytowania (12 - 360 miesięcy), wyświetlane w Label
- ComboBox – Typ rat (Malejące / Równe)
- Image – obrazek typu kredytu (dom/samochód/gotówka)

GroupBox 2 – „Wyniki”:

- TextBlock – Pierwsza rata
- TextBlock – Ostatnia rata
- TextBlock – Łączna kwota do spłaty
- TextBlock – Łączne odsetki
- ProgressBar – Procent odsetek w stosunku do kwoty

GroupBox 3 – „Harmonogram spłaty”:

- ListBox – lista pierwszych 12 rat (np. „Rata 1: 5234,50 zł”)

Panel dolny:

- Button – „OBLICZ”
- Button – „WYCZYŚĆ”

Działanie aplikacji:

1. Walidacja kwoty kredytu (zakres 10 000 - 1 000 000).
2. Algorytm rat malejących:
   - Część kapitałowa = Kwota / Liczba_miesięcy
   - Część odsetkowa = Pozostała_kwota × Oprocentowanie / 12
   - Rata = Część_kapitałowa + Część_odsetkowa
3. ListBox wyświetla pierwszych 12 rat w formacie: „Rata 1: 5234,50 zł”.
4. ProgressBar pokazuje procent odsetek (odsetki/kwota × 100%).
5. Image zmienia się w zależności od wybranego typu kredytu w ComboBox.

Wymagania:

Kategoria	Wymaganie	Punkty
UI	3 GroupBoxy z kontrolkami	2
Walidacja	Zakres kwoty kredytu (10k-1mln)	1
Logika	2 Slidery z Label (oprocentowanie, okres)	2
Logika	ComboBox typu rat	1
Algorytm	Obliczanie rat malejących	5
ListBox	Harmonogram pierwszych 12 rat	2
Image	Obrazek typu kredytu (zmienia się)	2