Programowanie / GUI w Javie

Zakres tematów — co umieć na test i egzamin

PJATK · opracowano na podstawie projektów z wykładów (rozdziały 1–20)

Jak korzystać

Dla każdego rozdziału masz **zagadnienia** i **na co uważać** (pułapki, które lubi skrupulatny wykładowca). Do tego dołączone: **fiszki** (powtórka pojęć), **test na czas** (multiple-answer) i **paczka ćwiczeń** (predict-output + Swing/JavaFX do uzupełnienia).

Mapa przedmiotu — bloki

Blok	Rozdziały	O czym
A · Rdzeń OOP	1–9	typy, dziedziczenie, override/overload, equals/hashCode, klasy wewnętrzne, abstrakc
B · Java funkcyjna	6–7, 10–12	lambdy, interfejsy funkcyjne, Iterable/Iterator, strumienie, Consumer/Function/Predic
C · Współbieżność	13	wątki, Runnable/Thread, synchronized, wyścig, join
D · GUI Swing	14–18	komponenty, layouty, zdarzenia, JList/JTable + modele (MVC)
E · GUI JavaFX	19	Application/Stage/Scene, properties & bindings, FXML, TableView/ListView, CSS, anim
F · Wzorce	20	Singleton, Factory Method, Memento, Flyweight

Profil wykładowcy: skrupulatny i ostry; testy **multiple-answer** (1–4 poprawne), mało czasu → liczy się **pewna** wiedza, nie zgadywanie. Najwięcej pułapek: override vs overload (typ statyczny!), kolejność inicjalizacji, finally z return, cache Integer, wyścigi wątków, różnica Swing (ręczne fire...) vs JavaFX (ObservableList).

Rozdział po rozdziale

R1 — Podstawy

Zagadnienia

- typy prymitywne vs referencyjne; przekazywanie przez wartość/referencję
- pętle (for, for-each, while), tablice 1D/2D
- operacje na tablicach: przekątne, brzegi, transpozycja

Na co uważać

⚠ for-each nie daje indeksu i nie zmienia elementu tablicy przez zmienną pętli

⚠ kolejność wypełniania tablicy 2D w zadaniach „co wypisze"

R2 — OOP

Zagadnienia

- dziedziczenie, super(), przesłanianie vs przeciążanie
- equals()/hashCode() — kontrakt
- polimorfizm i dynamiczne wiązanie metod
- rekurencja (silnia, Fibonacci, BigDecimal)

Na co uważać

⚠ przeciążenie wybierane w KOMPILACJI wg typu statycznego; przesłonięcie w WYKONANIU wg typu obiektu

⚠ pola wiązane statycznie (nie polimorficznie)

⚠ równe obiekty → równy hashCode (nie odwrotnie)

R3 — I/O i serializacja

Zagadnienia

- strumienie bajtowe i znakowe; FileInput/OutputStream
- ObjectInput/OutputStream, Serializable, transient
- try/catch/finally, try-with-resources

Na co uważać

⚠ finally z return nadpisuje wartość zwracaną

⚠ zasób w try-with-resources musi być AutoCloseable

⚠ Serializable to interfejs znacznikowy (bez metod)

R4 — Klasy wewnętrzne

Zagadnienia

- składowa (member), statyczna zagnieżdżona, lokalna, anonimowa
- outer.new Inner(); new Outer.Static()
- Outer.this.pole, przesłanianie nazw

Na co uważać

⚠ static nested nie ma dostępu do pól instancyjnych

⚠ kolejność przesłaniania: lokalna > pole inner > Outer.this

R5 — Klasy abstrakcyjne

Zagadnienia

- abstract class, metody abstrakcyjne vs konkretne
- różnice względem interfejsu
- szablon zachowania

Na co uważać

⚠ brak instancji klasy abstrakcyjnej

⚠ konkretna podklasa musi nadpisać wszystkie metody abstrakcyjne

⚠ jedna klasa bazowa, wiele interfejsów

R6–7 — Lambdy

Zagadnienia

- interfejs funkcyjny (@FunctionalInterface)
- składnia lambd, klasy anonimowe
- referencje do metod (::)

Na co uważać

⚠ lambda zastępuje tylko interfejs funkcyjny (1 metoda abstrakcyjna)
⚠ rodzaje :: — statyczna, na obiekcie, na typie, konstruktor

R8 — Enumy

Zagadnienia

- pola, konstruktor (prywatny), metody
- przesłanianie metody per stała
- values(), valueOf(), ordinal(), name()

Na co uważać

⚠ konstruktor enuma NIE może być public
⚠ name() ignoruje przesłonięty toString()
⚠ valueOf z błędną nazwą rzuca wyjątek

R9 — Generyki

Zagadnienia

- typy generyczne, ograniczenia (extends)
- wildcardy: <?>, <? extends T>, <? super T> (PECS)
- metody generyczne

Na co uważać

⚠ do <? extends T> NIE wolno dodawać elementów
⚠ PECS: Producer Extends, Consumer Super
⚠ <?> czytamy tylko jako Object

R10 — Iterable/Iterator

Zagadnienia

- implementacja Iterable<T> i Iterator<T>
- hasNext()/next(), for-each
- NoSuchElementException

Na co uważać

⚠ for-each wymaga Iterable
⚠ next() po końcu rzuca NoSuchElementException
⚠ iterator trzyma własny stan

R11 — Strumienie

Zagadnienia

- Stream: operacje pośrednie vs końcowe
- map, filter, sorted, distinct
- collect, count, reduce, Collectors (joining, groupingBy)
- IntStream

Na co uważać

⚠ operacje pośrednie są leniwe; uruchamia je końcowa
⚠ strumień jest jednorazowy
⚠ groupingBy zwraca Map grup

R12 — Interfejsy funkcyjne

Zagadnienia

- Consumer, Supplier, Function, Predicate (+ Bi-)
- andThen vs compose
- łączenie predykatów (and/or/negate)

Na co uważać

⚠ andThen: g(f(x)); compose: f(g(x))
⚠ Consumer.andThen wykonuje oba po kolei
⚠ znać sygnatury metod (accept/get/apply/test)

R13 — Wątki

Zagadnienia

- Thread vs Runnable; start() vs run()
- wyścig (race condition) na wspólnym polu
- synchronized (blok i metoda), monitor
- join()

Na co uważać

⚠ start() = nowy wątek; run() = bieżący
⚠ wykluczanie działa tylko na TYM SAMYM monitorze
⚠ volatile ≠ atomowość value--

R14 — Swing: komponenty i layouty

Zagadnienia

- JFrame, JButton, JLabel, JTextField/Area, JCheckBox, JRadioButton+ButtonGroup, JSlider, JComboBox, JTabbedPane, JSplitPane, JOptionPane
- FlowLayout, BorderLayout, GridLayout
- pack/setSize, setLocationRelativeTo, setDefaultCloseOperation

Na co uważać

⚠ domyślny layout JFrame = BorderLayout
⚠ strefa BorderLayout mieści 1 komponent
⚠ setVisible(true) na końcu

R15 — Swing: zdarzenia

Zagadnienia

- model delegacji (źródło + listener)
- ActionListener, KeyListener/KeyAdapter, WindowListener/Adapter
- Event Dispatch Thread (EDT)

Na co uważać

⚠ adaptery = puste implementacje (nadpisz tylko potrzebne)
⚠ kod GUI na EDT (SwingUtilities.invokeLater)
⚠ długie zadania poza EDT

R16–18 — JList/JTable

Zagadnienia

- wzorzec model–widok (MVC)
- AbstractTableModel / AbstractListModel
- fireTable... (powiadomienie widoku)
- CellRenderer

Na co uważać

⚠ bez fire... zmiana modelu nie odświeży widoku
⚠ metody obowiązkowe: getRowCount/getColumnCount/getValueAt
⚠ isCellEditable steruje edycją

R19 — JavaFX

Zagadnienia

- Application/Stage/Scene, launch(), start()
- Property, ChangeListener/InvalidationListener
- bind() vs bindBidirectional(), DoubleBinding, Bindings
- ObservableList + FXCollections
- TableView (PropertyValueFactory), ListView
- FXML + kontroler, CSS, animacje (Transition)

Na co uważać

⚠ bind = jednostronne (q.set rzuci wyjątek do unbind)
⚠ ObservableList sam odświeża widok (bez fire...)
⚠ FX faworyt wykładowcy: properties/bindings

R20 — Wzorce projektowe

Zagadnienia

- Singleton (prywatny konstruktor + getInstance)
- Factory Method (metoda wytwórcza)
- Memento (Originator/Memento/Caretaker)
- Flyweight (współdzielenie obiektów)

Na co uważać

⚠ Singleton niesynchronizowany NIE jest bezpieczny wątkowo
⚠ Flyweight oszczędza pamięć przez współdzielenie
⚠ umieć rozpoznać wzorzec po opisie ról