

Wymagania edukacyjne — Pracownia programowania obiektowego (Java)

Zawód: technik programista (technikum 5-letnie)

Kwalifikacja: INF.04. Projektowanie, programowanie i testowanie aplikacji

Jednostka efektów kształcenia z podstawy programowej: INF.04.4. Programowanie obiektowe

Źródła: *Program nauczania wydawnictwa Helion — technik programista* (dział 3.9. Pracownia programowania obiektowego), *Podstawa programowa kształcenia w zawodach — INF.04.4. Programowanie obiektowe* (Dziennik Ustaw, poz. 991).

Zasady konstrukcji wymagań

1. Przedmiot **Pracownia programowania obiektowego** (praktyka) jest oceniany na podstawie **działającego kodu** — uczeń tworzy, implementuje, uruchamia, testuje i poprawia programy w środowisku IDE, zgodnie z zaleceniami programu nauczania dla zajęć praktycznych.

2. **Zasada kumulatywności ocen.**

Uzyskanie oceny wyższej wymaga opanowania wiadomości i umiejętności przypisanych do wszystkich ocen niższych. Np. na ocenę **dobrą (4)** uczeń musi dodatkowo spełniać wymagania na oceny **dopuszczającą (2)** i **dostateczną (3)**. Ocena **celująca (6)** oznacza pełne (100%), bezbłędne i w pełni samodzielne opanowanie wymagań przypisanych do ocen 2–5 w danym dziale.

3. **Adaptacja do języka Java.**

Program nauczania i podstawa programowa są sformułowane niezależnie od języka (C++/C#/Java/Python). Poniższe wymagania dostosowano do Javy, w tym do jej specyfiki wobec pojęć ogólnych:

- *konstruktor kopiujący* → w Javie tworzony ręcznie (konstruktor przyjmujący obiekt tego samego typu) lub przez `clone()/Cloneable` — Java nie generuje go automatycznie;
- *destruktor* → Java nie ma destruktorów w rozumieniu C++ (obiekty usuwa **Garbage Collector**); najbliższym odpowiednikiem jest zwalnianie zasobów przez `try-with-resources` i interfejs `AutoCloseable` (ewentualnie przestarzała metoda `finalize()`, wspominana wyłącznie historycznie);
- *funkcje/klasy zaprzyjaźnione* → Java nie zna słowa kluczowego `friend`; zbliżony efekt (uprzywilejowany dostęp) osiąga się przez dostęp pakietowy (brak modyfikatora) lub klasy zagnieżdżone (*nested/inner classes*).

4. Godziny zajęć wg planu nauczania: **Pracownia programowania obiektowego** (praktyka) — klasa III: 2 godz./tydz. (60 godz.), klasa IV: 2 godz./tydz. (60 godz.), razem 120 godz.

5. Ocena niedostateczna.

We wszystkich działach ocenę niedostateczną otrzymuje uczeń, który nie spełnia wymagań na ocenę dopuszczającą — nie posiada wiedzy/umiejętności koniecznych do kontynuowania nauki nawet przy pomocy nauczyciela, nie podejmuje samodzielnej pracy lub oddaje prace niesamodzielne.

KLASA III

Pracownia programowania obiektowego

Dział 1. Tworzenie, uruchamianie i debugowanie aplikacji w Javie

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Tworzy projekt w IDE (np. IntelliJ IDEA, Eclipse) według instrukcji, uruchamia gotowy program, zapisuje pliki źródłowe, wykonuje proste modyfikacje wskazanego kodu.
Dostateczna (3)	Samodzielnie tworzy i uruchamia prostą aplikację konsolową wykorzystującą metody; stosuje Scanner do odczytu danych i System.out.println() do ich wyświetlenia; kompiluje program i poprawia podstawowe błędy składniowe.
Dobra (4)	Samodzielnie korzysta z debuggera (punkty przerwania, krokowe wykonanie, podgląd wartości zmiennych); lokalizuje i poprawia typowe błędy logiczne we własnym programie.
Bardzo dobra (5)	Testuje program dla różnych zestawów danych, w tym granicznych i błędnych; samodzielnie diagnozuje przyczynę wyjątku czasu wykonania i eliminuje ją.
Celująca (6)	Bezблędnie i w pełni samodzielnie tworzy, uruchamia i debuguje programy przewidziane w dziale; organizuje kod w pakietach; konfiguruje ustawienia projektu; wykorzystuje system kontroli wersji (np. Git) do pracy nad ćwiczeniami.

Dział 2. Implementacja metod i przetwarzanie danych

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Wywołuje metodę bez parametrów według wzoru; wykonuje proste operacje na zmiennych podstawowych typów.
Dostateczna (3)	Definiuje i wywołuje metody z parametrami oraz metody zwracające wartość; tworzy prostą metodę rekurencyjną (np. silnia) według wzoru; stosuje tablicę jednowymiarową w programie.

Dobra (4)	Samodzielnie dzieli program na metody pomocnicze; stosuje przeciążanie metod; wykorzystuje tablicę dwuwymiarową oraz metody klasy String do przetwarzania danych; stosuje rekurencję do rozwiązania typowego problemu.
Bardzo dobra (5)	Projektuje samodzielny podział programu na metody dla złożonego zadania; dobiera typ danych i strukturę (tablica/String) do rozwiązywanego problemu; testuje rozwiązanie na kilku zestawach danych.
Celująca (6)	Bezbłędnie i w pełni samodzielnie rozwiązuje najbardziej złożone zadania łączące metody, rekurencję, tablice i przetwarzanie tekstu, przewidziane programem działu; optymalizuje własne rozwiązanie i uzasadnia wybór podejścia.

Dział 3. Implementacja klas i obiektów

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Tworzy prostą klasę na podstawie wzoru (pola, jedna metoda), tworzy obiekt tej klasy i wywołuje jego metodę.
Dostateczna (3)	Implementuje klasę z konstruktorem, polami i metodami; tworzy kilka obiektów tej samej klasy i wykorzystuje je w prostej aplikacji konsolowej.
Dobra (4)	Stosuje pola prywatne, konstruktor z parametrami, gettery i settery; nadpisuje metodę toString(); tworzy kilka współpracujących klas uporządkowanych w osobnych plikach.
Bardzo dobra (5)	Projektuje i implementuje model obiektowy dla opisanego problemu; waliduje dane przekazywane do konstruktora/setterów; testuje działanie metod dla różnych obiektów.
Celująca (6)	Bezbłędnie i w pełni samodzielnie implementuje rozbudowany model obiektowy z kilkoma współpracującymi klasami, w pełni zgodny z zakresem wymagań działu, bez błędów w hermetyzacji, konstruktorach i metodach.

Dział 4. Dziedziczenie i polimorfizm w praktyce

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Tworzy klasę pochodną na podstawie przykładu i korzysta z odziedziczonych pól/metod.
Dostateczna (3)	Implementuje klasę bazową i pochodną; stosuje super() w konstruktorze klasy pochodnej; przesłania metodę z użyciem @Override.

Dobra (4)	Implementuje prostą hierarchię kilku klas; wykorzystuje polimorficzne wywołanie metody (obiekt klasy pochodnej przez referencję typu klasy bazowej).
Bardzo dobra (5)	Tworzy aplikację wykorzystującą dziedziczenie i polimorfizm do obsługi kilku podobnych obiektów wspólnym kodem; testuje działanie różnych wariantów obiektów.
Celująca (6)	Bez błędnie i w pełni samodzielnie tworzy hierarchie klas z wykorzystaniem dziedziczenia i polimorfizmu; rozbudowuje aplikację o nową klasę pochodną bez modyfikowania istniejącej logiki przetwarzania, co potwierdza pełne opanowanie zasady polimorfizmu.

KLASA IV

Pracownia programowania obiektowego

Dział 1. Zaawansowana implementacja klas

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Modyfikuje gotową klasę według instrukcji; tworzy konstruktor z parametrami; wykorzystuje pole statyczne z przykładu.
Dostateczna (3)	Tworzy klasę z kilkoma konstruktorami; implementuje podstawowy konstruktor kopiujący (ręcznie); stosuje pole statyczne (np. licznik utworzonych obiektów).
Dobra (4)	Samodzielnie projektuje klasy współpracujące ze sobą; implementuje mechanizm kopiowania obiektu; stosuje try-with-resources lub jawną metodę zwalniającą zasoby tam, gdzie to uzasadnione.
Bardzo dobra (5)	Implementuje klasę zagnieżdżoną lub dostęp pakietowy tam, gdzie wymaga tego zadanie; poprawnie zarządza cyklem życia obiektów w rozbudowanym programie; testuje poprawność kopiowania obiektów.
Celująca (6)	Bez błędnie i w pełni samodzielnie tworzy projekt z poprawnym zarządzaniem zasobami i danymi, zgodnie z pełnym zakresem wymagań działu; wykrywa i eliminuje potencjalne błędy związane z kopiowaniem obiektów lub wyciekami zasobów.

Dział 2. Dziedziczenie zaawansowane, klasy abstrakcyjne i interfejsy w praktyce

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Tworzy klasę pochodną z pomocą nauczyciela; implementuje jedną metodę abstrakcyjną według wzoru.

Dostateczna (3)	Tworzy klasę bazową (abstrakcyjną lub interfejs) i kilka klas pochodnych/implementujących; implementuje wymagane metody abstrakcyjne.
Dobra (4)	Samodzielnie projektuje i implementuje hierarchię kilku klas z klasą abstrakcyjną lub interfejsem; przetwarza obiekty klas pochodnych za pomocą wspólnego typu (polimorfizm).
Bardzo dobra (5)	Implementuje aplikację wykorzystującą klasy abstrakcyjne, interfejsy i polimorfizm do obsługi różnych typów obiektów; testuje różne warianty działania programu.
Celująca (6)	Bezbłędnie i w pełni samodzielnie implementuje hierarchie klas z klasami abstrakcyjnymi i interfejsami; rozbudowuje aplikację o nowe klasy pochodne/implementujące bez zmiany istniejącej logiki przetwarzania w wielu miejscach kodu, co potwierdza pełne opanowanie polimorfizmu.

Dział 3. Programowanie generyczne w praktyce

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Wykorzystuje gotową klasę generyczną (np. <code>ArrayList<T></code>) do przechowywania obiektów jednego typu, według instrukcji.
Dostateczna (3)	Tworzy prostą klasę generyczną z jednym parametrem typu; dodaje, usuwa i odczytuje elementy z kontenera przechowującego obiekty własnej klasy.
Dobra (4)	Samodzielnie projektuje i implementuje klasę generyczną (np. prosty stos lub para wartości); wykorzystuje ją do przechowywania obiektów różnych własnych klas w kolejnych zadaniach.
Bardzo dobra (5)	Implementuje generyczny kontener z podstawowymi operacjami (dodawanie, usuwanie, wyszukiwanie elementu); testuje działanie dla kilku różnych typów danych.
Celująca (6)	Bezbłędnie i w pełni samodzielnie projektuje klasę generyczną z kilkoma parametrami typu, w pełni wykorzystując możliwości programowania generycznego przewidziane programem działu, i uzasadnia jej uniwersalność.

Dział 4. Obsługa wyjątków w praktyce

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Umieszcza wskazany fragment kodu w bloku <code>try...catch</code> według wzoru i wyświetla komunikat po wystąpieniu błędu.

Dostateczna (3)	Obsługuje typowe wyjątki (np. przy błędnych danych wejściowych); stosuje try...catch...finally; testuje działanie programu dla błędnych danych.
Dobra (4)	Stosuje walidację danych oraz kilka bloków catch; stosuje throw i throws; zapewnia poprawne działanie programu mimo wprowadzenia nieprawidłowych danych.
Bardzo dobra (5)	Tworzy własny wyjątek (klasa dziedzicząca po Exception/RuntimeException) i wykorzystuje go w aplikacji; przygotowuje zestaw danych testowych: poprawnych, granicznych i błędnych.
Celująca (6)	Bez błędnie i w pełni samodzielnie projektuje spójny mechanizm obsługi błędów w rozbudowanym programie, oddzielający walidację danych od logiki aplikacji i komunikatów dla użytkownika.

Dział 5. Projekt końcowy — aplikacja obiektowa w Javie

Przykładowe tematy projektów: system obsługi egzaminów (klasa Egzamin i klasa pochodna EgzaminPraktyczny), system biblioteczny, rejestr uczniów i ocen, ewidencja pojazdów, system rezerwacji.

Ocena	Uczeń wykonuje
Dopuszczająca (2)	Oddaje częściowo działający projekt wykorzystujący przynajmniej jedną klasę i podstawowe metody; realizuje fragment wymagań przy znacznej pomocy nauczyciela.
Dostateczna (3)	Tworzy działającą aplikację konsolową wykorzystującą klasy, obiekty, konstruktory i metody; realizuje najważniejsze funkcje projektu; potrafi krótko wyjaśnić działanie własnego kodu.
Dobra (4)	Samodzielnie realizuje większość wymagań projektu; stosuje hermetyzację i obsługę wyjątków; dzieli kod na kilka współpracujących klas; testuje działanie aplikacji.
Bardzo dobra (5)	Realizuje komplet wymagań projektu; stosuje uzasadnione dziedziczenie, polimorfizm, klasy abstrakcyjne lub interfejsy oraz generykę tam, gdzie to zasadne; zapewnia walidację danych; prezentuje projekt i wyjaśnia przyjęte rozwiązania.
Celująca (6)	Realizuje w 100%, bezbłędnie i w pełni samodzielnie wszystkie wymagania projektu, poprawnie i świadomie łącząc wszystkie poznane mechanizmy obiektowe (dziedziczenie, polimorfizm, klasy abstrakcyjne/interfejsy, generykę, obsługę wyjątków); testuje aplikację dla wielu przypadków, w tym granicznych; przeprowadza refaktoryzację własnego kodu, podnosząc jego czytelność i jakość.