Projekt Świat

Wyobraźmy sobie płaski dwuwymiarowy świat o określonych rozmiarach, który składa się z organizmów podzielonych na różne rodzaje, m.in. rośliny (plants) oraz zwierzęta (animals).

W katalogu LivingWorld:

https://github.com/ZeissIQSPL/CppAtAcademia/tree/main/CPP_01_OOP/LivingWorld

znajduje się szkic takiego projektu. Poniższe zadania dotyczą głównie rozbudowy oraz refaktoringu zaprezentowanego kodu.

Zadania

1. **[5 punktów]** Każdy Organizm powinien posiadać pole przechowujące historię jego przodków. Historia jest zaimplementowana jako lista numerów tur narodzin i śmierci przodków. Wyposaż Organizmy w odpowiednie konstruktory (kopiujące, przenoszące), oraz destruktory. Obiekty powinny posiadać również odpowiednie mechanizmy aktualizacji nowych danych. Jak powinny się zachowywać operatory przypisania?
2. **[5 punktów]** Do świata dodaj nowe organizmy zgodnie z opisem świata z projektu PythonWorld https://github.com/tborzyszkowski/PythonWorld/tree/master/Laboratorium

   (katalogi od `World01` do `World04`). Wszystkie klasy (także te istniejące) wyposaż w odpowiedni model dziedziczenia oraz odpowiednie metody polimorficzne.
3. **[5 punktów]** Po realizacji zadania z poprzedniego punktu, spraw aby klasy Organism, Animal i Plant były klasami abstrakcyjnymi.
4. **[5 punktów]** W klasie World mamy naiwny mechanizm serializacji świata. Nie jest on zgodny z SOLID. Przebuduj ten mechanizm tak, aby każdy obiekt sam zajmował się swoją serializacją i deserializacją. Przetestuj nowy mechanizm na wybranych przykładach.
5. **[5 punktów]** Domyślnie program będzie wyświetlał kolejne epoki w konsoli. Zaprogramuj okienkową wersję programu w dowolnej technologii (SDL, Qt, OpenGL, DirectX lub inne).

Termin ostateczny: 12 czerwca 2025 r.

Kod w szablonie projektu nie zawsze napisany jest zgodnie z zaleceniami nowoczesnego programowania w C++. W ramach refaktoringu kodu popraw i oznacz te fragmenty, które mogłyby być napisane bardziej nowocześnie. Opowiesz o nich prowadzącemu podczas obrony. Dodaj własne autorskie komentarze, które pomogą w zrozumieniu kodu.

Podczas obrony projektu należy znać podstawowe zasady programowania obiektowego oraz mechanizmy języka C++ omawiane na wykładach. Prowadzący może poprosić o małe zmiany w kodzie mające na celu sprawdzenie samodzielności oraz zrozumieniu projektu. Ważne aby orientować się w kodzie całego projektu.

Uwaga! Niniejsza specyfikacja może ulec nieznacznym zmianom, aby dokładniej przedstawić wymagania.