Opracowanie: dr inż. Łukasz Maliński, Politechnika Śląska, 2022

LABORATORIUM 2.2
TWORZENIE INSTANCJI I INTERAKCJA Z OBIEKTAMI

Zadanie podstawowe [2,0 pkt.]

▲ Dla klasy **GraPlanszowa**, zdefiniowanej w pliku „klasy.h” zdefiniuj konstruktor argumentowy, pozwalający na ustawienie wszystkich pól oprócz:

- liczba ocen (zawsze ustawiane na 0 w nowej instancji),
- oceny graczy (zawsze ustawione na 0 w nowej instancji),

na zadane wartości (przesłane w jego argumentach). Użyj 2 różnych sposobów inicjalizacji wartości pól klasy, przy czym pola, które nie wymagają sprawdzenia ograniczeń, ustaw bez użycia sterów.

Wskazówki: Zauważ, że nie wszystkie pola możemy zadać w konstruktorze, gdyż każda nowa gra nie może być już oceniona (pola związane z oceną ustawiaj zawsze na 0 i to w każdym konstruktorze). Zastanów się, które pola należy ustawiać z użyciem seterów, które na liście inicjalizacyjnej, a dla których najlepiej jest zdefiniować tylko wartości domyślne.

▲ Dodaj do klasy konstruktor domyślny, który ustawi pola następująco: tytuł - „Nowa Gra”, liczba graczy – 2, czas trwania - 60 (minut), typ gry - przygodowa.

Wskazówki: Spróbuj wykorzystać tu delegację zadania do konstruktora argumentowego, tak jak to pokazano w przykładzie z instrukcji (zadanie A). Jeśli nie wiesz jak to zrobić to staraj się jak najwięcej pól ustawić przez listę inicjalizacyjną.

▲ Dodaj do klasy funkcjonalność, polegającą na prowadzeniu statystyk ile jest instancji gier ocenionych w przedziałach: <0,0, 1.0>,(1,0 2,0> … (4,0, 5,0>, oraz ile jest gier jeszcze nie ocenionych. Instancje gier, które przestają istnieć powinny wypadać ze statystyk, a każda nowa ocena (użycie metody oceń), powinna aktualizować statystyki. Napisz składnik funkcjonalny dla całej klasy, który w przejrzysty sposób wypisze na ekranie statystyki.

Wskazówki: Zauważ, że do tego celu potrzeba więcej niż jeden licznik. Spróbuj rozwiązać ten problem tak, aby w przyszłości łatwo można było zmodyfikować liczbę przedziałów (patrz zadanie A z instrukcji). Zaliczanie gry do przedziałów najlepiej jest wydzielić do osobnej metody prywatnej. Zauważ, że konstruktory „nie mają dylematu”, do którego przedziału przydzielić grę (zastanów się po czym odróżnić grę o faktycznej ocenie 0.0, od gry jeszcze ani razu nie ocenionej).

■ Napisz dwa testy jednostkowe (jako funkcje globalne w programie):

- pierwszy, sprawdzający czy prawidłowo zliczane są gry nie ocenione,
- drugi, sprawdzający czy prawidłowo zliczane są gry ocenione.

Wskazówki: Na potrzeby tych testów trzeba koniecznie rozbudować klasę o możliwość programowego odczytu wartości liczników. Oba testy powinny na początku upewnić się czy wszystkie liczniki są wyzerowane - nie zeruj ich ręcznie, tylko posługuj się w kodzie testowym wyłącznie instancjami lokalnymi. Jeśli jakiś test na wstępie wykaże, że liczniki nie są zerowe to jest to sygnał, że w klasie jest coś nie tak z inicjalizacją liczników lub z ich dekrementacją przy usuwaniu instancji. W każdym teście utwórz instancje pasujące do testowanego problemu i inne, aby mieć grupę kontrolną. Nie trać czasu na wymyślanie danych opisujących gry. Instancje domyślne są w pełni wystarczające do działania testów.

Ocenianie:

Konstruktor argumentowy:	0,2 pkt.
Konstruktor domyślny:	0,2 pkt.
Statystyki ocen:	0,8 pkt.
Testy jednostkowe:	0,6 pkt.
Ogólna jakość kodu:	0,2 pkt.

Zadanie ambitne [2,0 pkt.]:

▲ Dostosuj klasę **Point** z zadnia domowego (za 2,0 pkt.) do możliwości pracy z instancją stałą, tak aby dało się używać funkcji odległość(). Absolutnie nie zmieniaj nic w nagłówku funkcji – zmiany dopuszczalne są tylko w klasie.

Wskazówki: Zauważ że funkcja przyjmuje 2 referencję do instancji klasy i chroni je przed zapisem. Przypomnij sobie co trzeba zrobić, aby można było wywołać metodę (szczególnie geter) dla instancji chronionej przed zapisem.

▲ Zmodyfikuj klasę **Point** tak, aby mimo ochrony jej instancji przed zapisem możliwa był zamiana symbolu punktu.

Wskazówki: Zastanów się jakie specyfikatory należy dodać do pola klasy i setera aby możliwa była zmiana wartości tego pola nawet, gdy instancja jest chroniona przed zapisem.

▲ Napisz funkcję znajdującą dwa najdalej odległe od siebie punkty z tablicy punktów. Odległość licz tylko w jednym wymiarze (X lub Y) i przekaż funkcji, z którego getera (X czy Y) ma korzystać. Funkcja powinna zwrócić symbole obu tych punktów przez return.

Wskazówki: Zauważ, że trzeba tu przekazać do funkcji wskazanie getera. Oba getery (X i Y) mają taką samą listę argumentów i typ zwracany, co znacząco ułatwia sprawę. Zauważ że nie da się przekazać metody przez zwykły wskaźnik do funkcji. Warto posłużyć się tu aliasami typów do uproszczenia definicji typu. Trudność tego polecenia głownie opiera się na zawiłości składniowej związanej z przekazaniem getera do funkcji. Algorytm nie powinien stanowić na tym etapie nauki większego wyzwania.

■ Pokaż w programie głównym działanie wszystkich napisanych funkcjonalności.

Wskazówki: Skup się na pokazaniu tylko funkcjonalności z tego zadania, jeśli funkcja main zawiera już kod prezentujący działanie zadania domowego, to daj możliwość wyboru aktywności kodu przez kompilację warunkową. Zauważ, że funkcjonalności z zadania domowego nie są konieczne do wykonania tego zadania.

Ocenianie:

Praca z instancją stałą:	0,4 pkt.
Modyfikowalność symbolu:	0,4 pkt.
Funkcja licząca odległość:	
- przekazywanie informacji:	0,4 pkt.
- algorytm:	0,4 pkt.
Prezentacja w programie głównym:	0,2 pkt.
Ogólna jakość kodu:	0,2 pkt.