Programowanie obiektowe 2

Pliki źródłowe zadań należy przechowywać na repozytorium.

Na pierwszych zajęciach studenci dopisują się do listy swojej grupy przez formularz internetowy.

Zakładanie repozytorium i udostępnianie prowadzącemu odbywa się na pierwszych zajęciach.

format nazwy repozytorium (podawany w formularzu listy zajęć)

zajecia sobota 10-12
po2java-sb_2017_2018_l_<imie>_<nazwisko>.git

zajęcia niedziela 16-18
po2java-nd_2017_2018_l_<imie>_<nazwisko>.git

Zadanie 1.
Napisz program losujący liczbę z zakresu 0-100. Następnie program pyta się użytkownika, co to za liczba. Jeżeli użytkownik nie zgadł, dowiaduje się czy wylosowana liczba jest większa, czy mniejsza od podanej. Jeżeli zgadł, dowiaduje się ile wykonał prób i jest pytany o ochotę do dalszej gry.

Zadanie 2.
Napisz program, który pobiera dwie liczby oraz łańcuch znaków z wiersza poleceń (czyli jako parametry przy wywołaniu programu w konsoli) oraz wyświetla fragment podanego łańcucha, określony wprowadzonymi liczbami. Na przykład:
java Substring Witaj 2 4 powinien wyświetlić: taj
Obsłuż wszystkie możliwe wyjątki, które mogą wystąpić w przypadku złego zestawu argumentów.

Zadanie 3.
Napisz program liczący miejsca zerowe funkcji kwadratowej. Po uruchomieniu programu w konsoli użytkownik jest pytany o współczynniki. Wynik jest wyświetlany na konsoli oraz dopisywany do pliku, którego nazwa jest przekazana jako argument wywołania programu. Należy obsłużyć wszystkie spodziewane wyjątki (zła liczba argumentów, tekst zamiast numeru, itp).

Zadanie 4.
Napisz klasę Invoice, która może reprezentować fakturę wystawioną na przedmiot sprzedany w sklepie. Obiekt powinien przechowywać cztery informacje – ID przedmiotu, opis przedmiotu, ilość kupionych sztuk, cenę za jedną sztukę.

Napisz metody – konstruktor inicjalizujący pola, set/get do każdego pola, getInvoiceAmount zwracającą całkowity koszt.
Napisz aplikację demonstrującą możliwości klasy Invoice.

Zadanie 5.
Napisz program proszący o podanie 2 wektorów. Koniec wektora oznacza się za pomocą wciśnięcia klawisza enter. Jeżeli podany ciąg nie jest liczba, jest ignorowany. Następnie należy spróbować dodać wektory, jeżeli są równej długości. Jeżeli nie są, rzucany jest własny wyjątek DifferentVectorSizeException, za pomocą którego można podać a następnie odczytać długości tych wektorów.
Jeżeli są równej długości, wynik dodawania zapisywany jest do pliku. Jeżeli nie są równej długości, użytkownik jest proszony o ponowne wprowadzenie tych wektorów.

Zadanie 6.
Opracuj klasę SavingsAccount dla kont oszczędnościowych. Wykorzystaj zmienną statyczną annualIInterestRate do przechowywania rocznego oprocentowania wszystkich kont. Każdy obiekt klasy posiada prywatną instancję zmiennej savingsBalance przechowującą stan konta. Napisz konstruktor oraz metody pozwalające na wpłaty/wypłaty z konta.
Napisz metodę calclulateMonthlyInterest do obliczenia miesięcznego oprocentowania (roczne oprocentowanie * stan konta / 12), o które zostanie powiększony stan konta, po jej wywołaniu.
Napisz statyczną metodę modifyInterestRate, która ustawia pole annualIInterestRate do nowej wartości.
Napisz program testujący możliwości klasy SavingsAccount.

Zadanie 7.
Opracuj klasę Car, posiadającą pola Speed, Price, Color, konstruktor inicjalizujący pola oraz metodę getSalePrice();
Napisz klasę Truck dziedziczącą z klasy Car, posiadającą pole Weight oraz metodę getSalePrice(), zwracającą cenę z 10% zniżki, gdy waga przekracza 3000kg.
Napisz klasę Hatchback dziedziczącą z klasy Car, posiadającą pola Year (rok produkcji), manufacturerDiscount oraz metodę getSalePrice(); zwracającą cenę pomniejszoną o zniżkę producenta.
Napisz klasę Sedan dziedziczącą z klasy Car, posiadającą pola Length oraz metodę getSalePrice(); dającą 10% zniżki, gdy auto jest krótsze niż 4500cm.
Trzy wspomniane klasy powinny posiadać konstruktory umożliwiające inicjalizację wszystkich pól, także z klasy Car.
Opracuj klasę AutoShop, która instancje wszystkich czterech klas, ustawia ich parametry oraz wyświetla ceny.

Zadanie 8.
Napisz abstrakcyjna klasę Location a następnie dziedziczące z niej klasy Building, Floor, Room. Klasa Location ma abstrakcyjna metodę opis, która opisuje daną lokalizację i wszystkie jej podlokalizacje. Zaproponuj takie pola i metody Building, Floor, Room, by zapewniały one możliwość utworzenia trójstopniowej struktury hierarchicznej (Building -> Floor (przypisane do budynku) -> Room (przypisane do piętra)). Każdy obiekt klasy Location powinien posiadać pole klasy LocationID. Klasa LocationID posiada 3 pola (numer budynku, numer pietra, numer pokoju) i implementuje interfejs Comparable. Zapewnij by w trakcie tworzenia hierarchicznej struktury zapewnione było automatyczne tworzenie identyfikatorów. W oparciu o klasę LocationID napisz metody sprawdzające relacje pomiędzy dwoma dowolnymi lokalizacjami.

Terminy oddawania zadań – za każdy tydzień opóźnienia ocena jest obniżana o 0,5.

Zadania 1,2 – drugie zajęcia
Zadania 3,4,5 – trzecie zajęcia
Zadania 6,7,8 – czwarte zajęcia
Oddanie nieoddanych zadań – piąte zajęcia

Do zaliczenia laboratorium niezbędne jest oddanie wszystkich zadań i maksymalnie jedna nieusprawiedliwiona nieobecność. W przypadku wystąpienia niezapowiedzianych kolokwiów, ich zaliczenie jest również konieczne.

Możliwość wcześniejszego zaliczenia:
- zadania można zrobić wcześniej niż na przedstawione terminy, można oddać wszystkie nawet już na drugich zajęciach.
- oddawanie zadań odbywa się tylko w czasie zajęć grupy, a o ocenie decyduje data commitu na repozytorium

Proponowane projekty do zaliczenia laboratorium w przypadku braku obecności na zajęciach (wymagane jest także oddanie zadań przedstawionych powyżej). Istnieje możliwość uzgodnienia innego, indywidualnego projektu.

1) Liczenie funkcji kwadratowej
- obliczanie miejsc zerowych funkcji
- wykreślanie funkcji
- możliwość dodawania kolejnych funkcji na ten sam obszar wykresu
- ręcznie ustawiany zakres osi wykresu
- eksport współrzędnych funkcji (w podanym zakresie i kroku) do pliku - zapis ustawień programu do pliku

2) Kółko i krzyżyk dla dwóch graczy działające poprzez LAN – gra oparta na graficznym interfejsie użytkownika (GUI) – komunikacja poprzez UDP lub TCP
– możliwość wybrania portu z poziomu GUI

3) Tetris