Typy wyliczeniowe

1. Generator piwa na imprezę. Utwórz typ wyliczeniowy GatunekPiwa, który może przyjmować takie wartości jak: LAGER, PILZNER, STOUT, PORTER, MIODOWE. Następnie utwórz klasę Impreza, która w bezargumentowym konstruktorze generuje losowo 2 wybrane gatunki piwa i przypisuje je do tablicy 2 elementowej (pola klasy Impreza). Następnie dodaj metodę wyświetl nazwę piwa oraz kolor np. „Dostępne piwa to: Piwo Miodowe (jasne) oraz Piwo Stout (ciemne).”[Czyli opcja 1. Metoda w enumie opisPiwa():String → Piwo Miodowe (jasne) , opcja 2. Metoda w imprezie – enum.pobierzNazwePiwa():String + enum.pobierzTypPiwa():String]
2. Wygenerowanie i wyświetlenie talii kart.
    a. Utwórz klasę Karta
    b. Nadaj pola:
        i. Kolor kolor;
        ii. Figura figura;
    c. Zadeklaruj Kolor i Figure jako typ public enum
    d. Przygotuj możliwe wartości np. dla koloru:
        i. KIER
        ii. KARO
        iii. TREFL
        iv. PIK
    e. Nadpisz metodę .toString() klasy Karta, tak aby drukowała taki opis karty: „As kier”, „Walet trefl” lub „J ♣”
    f. Wygeneruj talię (enum.values) i wyświetl ją

Obiektowość

1. Utwórz klasę Samochod o polach predkosc, kolor, marka, rocznik. Niech prędkość będzie domyślnie równa 0 dla nowo utworzonego auta (nie trzeba jej podawać w konstruktorze). Następnie :
    a. Utwórz metodę przyspiesz(), która rozpędzi auto o 20 km/h ale do max 140 km/h
    b. nadpisz metodę .toString() tak aby wyświetlić opis auta (np. „Czerwone BMW rocznik 2000”)
    c. nadpisz metodę equals(), która zwróci true jeśli kolor, marka i rocznik będą takie same
    d. Przetestuj powyższe rozwiązania
    e. Utwórz podklasę SzybkiSamochod
    f. Nadpisz metodę przyspiesz() aby osiagac max 200 km/h
    g. Utwórz obiekt SzybkiSamochod o tych samych parametrach co Samochod – czy są sobie równe ?
    h. Jeśli tak, to nadpisz .equals() dla SzybkiegoSamochodu, tak aby Samochod!=SzybkiSamochod
2. Utwórz własny typ daty np. MojaData, który przyjmuje 3 argumenty typu int dla określenia dni, miesięcy i lat. Dodaj metodę wyświetl datę, który wyświetli datę w następujący sposób:
    a. 1.11.2011 (pierwszy listopada 2011)
    b. 01.11.2011
    c. * 1 lis 2011
3. Rozwinięcie poprzedniego zadania – Utwórz klasę Wydarzenie o 2 polach: nazwa wydarzenia (String) oraz data wydarzenia (MojaData). Nadaj metody klasie Wydarzenie:

    a. ilePozostaloLat():int – zwraca czas do wydarzenia w latach
    b. ilePozostaloMiesiecy(): int – zwraca czas do wydarzenia w miesiącach
    c. ilePozostaloDni():int – zwraca czas do wydarzenia w dniach
    d. Dla podpunktów a-c przyjmij jakąś datę na sztywno tj. 24.09.2017 jako dziś
    e. *Spróbuj pobrać aktualną datę z systemu i odpowiednio zmodyfikować metody a,b,c aby wyświetlić aktualny wynik (tips: klasy Date, Calendar)
4. Utwórz klasę Książka o takich polach jak nazwa, autor, rok wydania. Dodaj konstruktor, aby łatwiej tworzyć nowe książki oraz nadpisz .toString(), aby wyświetlić czytelny opis książki jak np. „Krzyżacy”, Henryk Sienkiewicz 2004.
5. Rozwinięcie poprzedniego zadania – dodaj klasę Biblioteka, która będzie przechowywać książki i umożliwiać ich wypożyczanie. Utwórz takie metody jak:
    a. czyKsiazkaDostepna(String nazwa Ksiazki):boolean
    b. wypozyczKsiazke(String nazwa Ksiazki):Ksiazka
    c. wyswietlWypozyczoneKsiazki():void
    d. wyswietlDostepneKsiazki():void
    e. Przetestuj rozwiązanie w psvm!
6. Utwórz klasę Punkt o polach x:int i y:int – obie wartości zainicjuj w konstruktorze klasy. Nadpisz metode .toString() aby zwracała informacje o współrzędnych punktu np. „(5,12)” dla x=5 i y=12.
7. Rozwinięcie poprzedniego zadania – dodaj klasę Plansza z metodą statyczną obliczOdleglosc(Punkt, Punkt):double
8. Utwórz klasę RownaniaKwadratowe o trzech polach a,b i c (nadawanych w konstruktorze) oraz metodach obliczDelte, obliczX1, obliczX2. Zaimplementuj według linka
9. Utwórz zaawansowany kalkulator, który po wykonaniu działania pyta użytkownika czy zapamiętać wynik ostatniej operacji (t/n). Zaś przy podawaniu liczby jest możliwość podania litery ‘w’, która oznaczać będzie wczytanie poprzednio zapisanego wyniku.
10. Utwórz klasę Pracownik o polach imie, nazwisko, miesięczne wynagrodzenie. Utwórz klasę Firma o polu pracownicy[] oraz metodach: obliczMiesiecznyKosztFirmy(), obliczRocznyKosztFirmy();
    a. Przetestuj rozwiązanie
    b. Rozszerz projekt o typ wyliczeniowy – Czas o wartościach DZIEŃ, MIESIĄC, ROK
    c. Dodaj metodę obliczKosztFirmy(Czas, int jednostka):double
    d. Przetestuj!
11. *Rozszerzenie zadania poprzedniego 11. Dodaj następujące metody do klasy Firma:
    a. dodajPracownika(Pracownik):void
    b. dodajPracownikaInteraktywnie():void – pyta użytkownika o podanie wszystkich niezbędnych danych do utworzenia nowego obiektu typu Pracownik i dołączeniu go do już istniejących
    c. srednieWynagrodzenie():double
12. ** Wyścigi Koni – napisz program, który symuluje gry w wyścigi koni. Zadaniem gracza jest typowanie faworyta przed rozpoczęciem wyścigu. Gracz rozpoczyna z saldem 100 złota i może obstawiać jednorazowo za nie więcej niż 50% swojego salda. W przypadku wygranej otrzymuje 2krotność postawionej stawki. Możesz użyć takich klas jak Koń, Trasa, Zakład, Punkt i/lub metod obstaw(double stawka), jedź():void
13. Przeciążanie metody – proste. Utwórz klasę Suma i wzbogać ją o 4 metody o tej samej nazwie dodaj, przyjmującej za argumenty: parę int, parę double, 3x int, 3x double. Przetestuj swoje rozwiązanie

14. Napisać program do obsługi zamówień. Program powinien składać się z klas Zamowienie oraz Pozycja. Zamowienie ma pole pozycje (Pozycje[]), maksRozmiar oraz konstruktor(maxRozmiar) i konstruktor bezargumentowy (gdzie maksRozmiar=10) a także etody dodajPozycje(Pozycja) i obliczWartosc():double, toString() zwraca spis pozycji zamówienia. Pozycja posiada 3 pola: nazwaTowaru, iloscSztuk, cenaSztuki, oraz metody: obliczWartosc() i toString().
    Przykładowy wynik działania programu:
    Zamówienie:

Chleb	2.00 zł	2 szt.	4.00 zł
Banany	6.00zł	1 szt.	6.00 zł

    Razem: 10.00 zł
15. Napisać klasę Lista, która będzie przechowywać liczby całkowite.
    a. Pola :
        i. int[] liczby,
        ii. int pojemność,
        iii. int rozmiar.
    b. Metody:
        i. konstruktor(int maxPojemnosc),
        ii. dodajElement(int):boolean,
        iii. znajdz(int):int – zwraca indeks pierwszego wystąpienia wskazanej liczby lub -1 gdy takiej nie ma,
        iv. wyswietlListe():void – wyswietla dane o liscie, tj. aktualna pojemność, max rozmiar oraz przechowywane elementy,
        v. usunPowtorzenia():void,
        vi. odwrocListe():void
16. Napisać klasę Czas do przechowywania czasu w godzinach i minutach.
    a. Pola:
        i. Int godziny
        ii. Int minuty
    b. Metody:
        i. Konstruktor(godziny, minuty)
        ii. toString():String → np. „29h 14 min”
        iii. dodaj(Czas t):Czas → zwróci nowy obiekt typu Czas będący sumą aktualnego czasu i podanego w argumencie metody
        iv. odejmij(Czas t): Czas → analogicznie jak w dodawaniu
        v. pomnóż(int krotnosc):Czas → okres czasu pomnożony przez podaną liczbę

<u>Dziedziczenie</u>

1. Pan Janusz prowadzi mały sklep zajmujący się tworzeniem brył przestrzennych na zamówienie szkół gimnazjalnych. W swojej ofercie umożliwia zakup brył drucikowych (tylko krawędzie, bez ścian i wnętrza), brył szklanych (tylko szklane ściany), oraz brył drewnianych (wypełnionych drewnem). Przygotuj program, który pozwoli mu obliczyć długości/wielkości niezbędnych materiałów dla utworzenia zamówionych brył, wiedząc, że w pojedynczym zamówieniu występuje wiele różnych brył takich jak:
    a. Prostopadłościan
    b. Sześcian
    c. Graniastosłup prawidłowy 3 kątny

d. Graniastosłup prawidłowy 5 kątny
e. Walec
f. Stożek
g. Ostrosłup 3 kątny
h. Ostrosłup 4 kątny
   Ściąga → https://matfiz24.pl/bryly

2. Napisz program do symulacji oferty lokaty bankowej. Celem programu jest przeprowadzenie symulacji różnych scenariuszy i wyświetlenie najlepszej dostępnej na rynku lokaty. W tym celu:

a. Utwórz klasę OfertaPodstawowa, przyjmującą oprocentowanie jako argument konstruktora oraz Kapitalizację
b. Utwórz klasę OfertaSpecjalna dziedziczącą po OferciePodstawowej
c. Utwórz enum Kapitalizacja przyjmujący 3 wartości: MIESIĘCZNA, KWARTALNA, ROCZNA (chodzi o kapitalizację odsetek)
d. OfertaPodstawowa zakłada kapitalizację o stałym oprocentowaniu, natomiast w OfercieSpecjalnej oprocentowanie początkowe jest niższe, ale wraz z czasem rośnie (1% rocznie, ale nie więcej niż 10%)
e. Utwórz klasę SymulatorZysku z metodą statyczną pobierającą Ofertę, depozyt oraz czas lokaty (w miesiącach)
f. Utwórz po co najmniej 2 obiekty OfertyPodstawowej i OfertySpecjalnej i przeprowadź symulację, która z ofert jest najbardziej korzystna
g. Nadpisz metodę .toString(), aby łatwiej wyświetlić wynik