Kolekcje

1. Utwórz 10 elementową tablicę przechowującą liczby całkowite (int)
    a. Dodaj 10 kolejnych liczb do tablicy rozpoczynając od liczby 1
    b. Wyświetl długość tablicy
    c. Wyświetl kolejno elementy tablicy wykorzystując pętlę for.
    d. Wykorzystując pętlę for pomnóż przez 2 kolejno każdy element w tablicy i zapisz go ponownie w tablicy.
    e. Wyświetl kolejno elementy tablicy wykorzystując pętlę for.
2. Utwórz listę ArrayList przechowującą liczby całkowite.
    a. Dodaj 3 dowolne liczby, a następnie wyświetl długość tablicy.
    b. Wyświetl pierwszy element zapisany w liście.
    c. Wyświetl ostatni element zapisany w liście
3. Utwórz listę ArrayList przechowującą liczby zmiennoprzecinkowe. Dodaj ręcznie 10 liczb do listy.
    a. Wyświetl sumę pierwszego i ostatniego elementu zapisanego w liście..
    b. Wyświetl iloczyn pierwszego i ostatniego elementu zapisanego w liście.
    c. Wyświetl iloraz drugiego i przedostatniego elementu zapisanego w liście.
4. Utwórz listę ArrayList przechowującą imiona.
    a. Dodaj 5 imion do listy,
    b. Wykorzystując pętle for pobierz i wyświetl kolejno wszystkie elementy z listy.
    c. Wykorzystując pętle for pobierz i wyświetl kolejno wszystkie elementy z listy zaczynając od ostatniego, a kończąc na pierwszym.
5. Utwórz tablicę ArrayList przechowującą imiona
    a. Korzystając z klasy Scanner pobierz od użytkownika 5 imion i zapisz je w liście.
    b. Wykorzystując pętlę while wyświetl wszystkie elementy znajdujące się w liście.
6. Utwórz kolejkę Queue korzystając z implementacji PriorityQueue.
    a. Pobierz od użytkownika ilość liczb jakie mają zostać wylosowane
    b. Wylosuj wskazaną przez użytkownika ilość liczb korzystając z klasy Random z zakresu <1,100> i dodaj je do kolejki.
    c. Wyświetl kolejno liczby występujące w kolejce.

Wyjątki

7. Utwórz listę ArrayList przechowującą liczby całkowite.
    a. Dodaj 5 liczb do listy.
    b. Pobierz 6 element z listy.
    c. Uruchom aplikację i sprawdź co pokaże IntelliJ w logach oraz w stacktrace.
    d. Zabezpiecz kod za pomocą klauzuli try catch.
    e. W przypadku wystąpienia błędu wyświetl stosowny komunikat.
8. Utwórz zmienną typu String, a następnie przypisz do niej imię “Olgierd”.
    a. Wyświetl imię tak aby imię było wyświetlone dużymi literami
    b. Wyświetl imię tak aby imię było wyświetlone małymi literami
    c. Wyświetl długość imienia wykorzystując odpowiednią metodę klasy String.
    d. Korzystając z odpowiedniej metody sprawdź czy imię rozpoczyna się od litery “O”
    e. Korzystając z odpowiedniej metody sprawdź czy imię zawiera literę “e”.

f. Korzystając z odpowiedniej metody usuń litery “i” oraz “e” z imienia, a następnie wyświetl imię po zmianach.
g. Wykorzystując pętlę for wyświetl kolejno litery imienia wraz z ich indeksem. Spodziewany rezultat wygląda następująco

[1] = O
[2] = l
[3] = g
[4] = i
[5] = e
[6] = r
[7] = d

Data i czas

9. Wykorzystując obiekt typu LocalTime wyświetl aktualny czas.
10. Wykorzystując obiekt typu LocalDate wyświetl aktualną datę.
11. Wykorzystując obiekt typu LocalDateTime wyświetl aktualną datę i godzinę.
12. Utwórz obiekt typu LocalDate przechowujący datę 01.01.2017 oraz obiekt typu LocalDate przechowujący datę 05.05.2017. Wykorzystując obiekt typu Period Wyświetl ile czasu minęło pomiędzy datami.
13. Utwórz obiekt typu LocalTime przechowujący godzinę 14:11 oraz obiekt typu LocalTime przechowujący godzinę 18:41. Wykorzystując klasę Duration oblicz ile czasu upłynęło pomiędzy godzinami.
14. Wyświetl bieżącą datę i godzinę w Tokyo.
15. Wyświetla bieżącą godzinę w Bydgoszczy. Wykorzystaj DateTimeFormatter aby wyświetlić datę w następującym formacie

```
3 lutego 2018 roku, sobota 22:12:27
```

OOP

16. Utwórz klasę Person posiadającą pola name, surname, age.
17. Utwórz konstruktor bezparametrowy.
18. Utwórz konstruktor przyjmujący wszystkie możliwe parametry do ustawienia klasy Person.
19. Dodaj metodę introduce wyświetlającą na konsoli imię oraz nazwisko osoby.
20. Utwórz klasę Address. Dodaj do klasy pola street, city, country, flatNo, homeNo.
21. Utwórz konstruktor przyjmujący wszystkie możliwe parametry do ustawienia klasy Address (street, city, country, flatNo, homeNo)
22. Rozbuduj klasę Person tak aby przechowywała klasę Address.
23. Utwórz nowy konstruktor przyjmujący wszystkie możliwe parametry do ustawienia klasy Person (name, surname, age oraz Address)
24. Utwórz metody umożliwiające ustawienie każdego parametru/pola klasy Person
25. Utwórz metody umożliwiające ustawienie każdego parametry/pola klasy Address
26. Utwórz metody umożliwiające pobranie każdego parametru/pola klasy Person
27. Utwórz metody umożliwiające pobranie każdego parametru/pola klasy Address

28. Utwórz klasę Engine posiadającą pola capacity, horsePower, fuelConsumption.
29. Utwórz konstruktor przyjmujący wszystkie możliwe parametry do ustawienie klasy Engine
30. Utwórz metody umożliwiające ustawienie każdego parametru/pola klasy Engine

31. Utwórz metody umożliwiające ustawienie każdego parametry/pola klasy Engine
32. Utwórz klasę abstrakcyjną Car posiadającą pola producer, model, color, seatsNumber oraz Engine.
33. Utwórz klasę SportCar dziedziczącą po klasie Car.
34. Utwórz konstruktor bezparametrowy w klasie Car, który inicjalizuje pole seatsNumber wartością 2.
35. Utwórz konstruktor przyjmujący wszystkie możliwe parametry do ustawienie klasy Car.
36. Utwórz konstruktor przyjmujący wszystkie możliwe parametry do ustawienie klasy SportCar.
37. Zmodyfikuj utworzony konstruktor tak aby wywołał wieloparametrowy konstruktor klasy Car.
38. *Zadanie na tworzenie odpowiedniego typu
    a. Utwórz klasę abstrakcyjną Figure posiadającą metodę abstrakcyjną
       `float countArea()`
    b. Utwórz metodę abstrakcyjną `void displayArea()` w klasie Figure.
    c. Utwórz klasę Rectangle dziedziczącą po klasie Figure
        i. dodaj pole sideA będące długością boku prostokąta.
        ii. dodaj pole sideB będące długością boku prostokąta.
        iii. nadpisz metodę countArea() klasy Figure i zaimplementuj ją.
        iv. Utwórz konstruktor przyjmujący wszystkie parametry potrzebne do obliczenia pola.
        v. nadpisz metodę displayArea() tak aby wyświetlała informację:
           `“Figura: Prostokąt, pole: <obilczone_pole>”`
       `gdzie <obliczone_pole> to wynik wykonania metody countArea()`
    d. Utwórz klasę Square dziedziczącą po klasie Figure
        i. dodaj pole side będące długością boku kwadratu
        ii. nadpisz metodę countArea() klasy Figure i zaimplementuj ją.
        iii. Utwórz konstruktor przyjmujący wszystkie parametry potrzebne do obliczenia pola.
        iv. nadpisz metodę displayArea() tak aby wyświetlała informację:
           `“Figura: Kwadrat, pole: <obilczone_pole>”`
       `gdzie <obliczone_pole> to wynik wykonania metody countArea()`
    e. Utwórz klasę Circle dziedziczącą po klasie Figure
        i. dodaj pole radius będące promieniem okręgu
        ii. nadpisz metodę countArea() klasy Figure i zaimplementuj ją.
        iii. Utwórz konstruktor przyjmujący wszystkie parametry potrzebne do obliczenia pola.
        iv. nadpisz metodę displayArea() tak aby wyświetlała informację:
           `“Figura: Koło, pole: <obilczone_pole>”`
       `gdzie <obliczone_pole> to wynik wykonania metody countArea()`
    f. Pobierz od użytkownika ilość poszczególnych figur (prostokąt, kwadrat, koło) jakie mają zostać wygenerowane.
    g. Utwórz listę figur o type List<Figure> figures

h. Utwórz odpowiednią ilość poszczególnych figur podanych przez użytkownika i dodaj każdą z nich do listy figures. Podczas tworzenia figur wylosuj wartości boków/promienia korzystając z klasy Random
i. Wyświetl po kolei powierzchnię każdej figury znajdującej się w tablicy za pomocą pętli for korzystając z metody displayArea().

Różne

39. Napisz program obliczający pole powierzchni koła. Promień koła użytkownik wprowadza z klawiatury. Program powinien zasygnalizować błędne dane (liczbę ujemną lub zero) i ponownie zapytac o długosc promienia. Wykorzystaj metodę Parse klasy float. Oczywiście nie zapomnij o przechwyceniu ewentualnych wyjątków.
40. Napisz program, który pobiera od użytkownika serię liczb różnych od zera (Zero kończy wprowadzanie danych). Po wprowadzeniu danych program oblicza sumę liczb wprowadzonych przez użytkownika, najmniejszą oraz największą liczbę. Na koniec działania wyświetla obliczone wartości. Oczywiście nie zapomnij o przechwyceniu ewentualnych wyjątków.
41. Napisz program rozwiązujący równanie kwadratowe ax2 + bx + c. Użytkownik musi podać wartości a, b oraz c. Program musi sprawdzić czy wprowadzone liczby są cyframi. Jeśli nie użytkownik musi podać wartości do momentu aż będą poprawne. Korzystając z pętli switch wyznacz pierwiastki równania kwadratowego w zależności od ilości rozwiązań (Mogą być 2, 1 lub wcale). Postaraj się przygotować klasę i metodę odpowiedzialną za obliczanie rozwiązania. Klasa powinna zawierać metodę o następującej sygnaturze (sygnatura do definicja metody składająca się z modyfikatora dostępu, zwracanego typu, nazwy, oraz przyjmowanych argumentów)

```
public Solution calculate(int a, int b, int c)
```