PO II Laboratorium 2-3 – kolekcje: listy

Zadanie 1.

- Utwórz nową listę tablicową, dodaj kilka stringów zawierających nazwy kolorów jak na rysunku,

- wydrukuj kolekcję (skorzystaj z pętli for z indeksami)
- wyświetl listę poczynając od danego indeksu, np. 2
- wyświetl listę od końca
- wstaw na początek listy element ”Pink”
- wstaw przed ”Black” kolor żółty ”Yellow”
- wyświetl listę (pętla for-each)
- wyświetl listę korzystaja z iteratora
- pobierz pierwszy i trzeci element z listy
- zamień wartość drugiego elementu na ”Blue”
- usuń 2gi element z listy
- sprawdź czy na liście znajduje się element „Red”
- posortuj listę
- do listy dodaj listę color_list o elementach 

- stwórz podlistę składającą się z pierwszych 3ch elementów listy
- usuń z listy elementy występujące w liście color_list
- wyczyść listę color_list

- Stwórz listę wiązaną

- Dodaj element Black na początku i końcu listy (użyj metod addFirst, addLast)
- Pobierz (ale nie usuwaj) pierwszy element z listy (peek)
- Zamień listę na listę tablicową

Zadanie 2.

A) Stwórz klasę `Pasazer` posiadającą następujące pola:

- `imie`
- `nazwisko`
- `wiek`

Klasa posiada następujące publiczne metody:

- konstruktor ustawiający pola na podstawie parametrów
- metody get
- `toString`
- `wyswietl` wyświetlającą wszystkie dane pasażera

Pobierz z klawiatury dane kilku pasażerów. Wyświetl pełną listę pasażerów.

B) Stwórz klasę `Wagon` posiadającą następujące pola:

- `maxMiejsc  // liczba wszystkich miejsc w wagonie`
- `listaPasazerow  // lista podróżujących aktualnie pasażerów`

Klasa posiada następujące publiczne metody:

- konstruktor ustawiający pola na podstawie parametrów
- metody get
- metodę `setListaPasazerow`
- metoda logiczna `dodajPasażera`, która jeśli jest miejsce w wagonie dodaje pasażera jeśli nie ma zwraca false
- `wyswietlPasazerow` wyświetlającą imiona i nazwiska wszystkich pasażerów
- `wyswietlInfoWagonu` która wyświetla:
    - liczbę wszystkich miejsc w wagonie
    - liczbę pasażerów w wagonie
    - liczbę wolnych miejsc
    - imiona i nazwiska pasażerów

C) Pasażerowie którzy ukończyli 65. rok życia otrzymują 50% zniżki na cenę biletu. Dodaj do klasy `Wagon` metodę `ileZeZnizkaSeniora` która zwraca liczbę pasażerów uprawnionych do otrzymania zniżki seniora.

Zadanie 3

Utwórz klasę ***`Student`*** zawierającą następujące pola:

- imie
- nazwisko
- lista ocen z przedmiotu

oraz metodę *`calculateFinalGrade`*() zwracającą średnią z ocen (lub -1 jeśli lista jest pusta).

Następnie utwórz klasę ***`StudentService`***, która będzie zawierała

-nazwę zajęć

-listę studentów uczęszczających na dane zajęcia

oraz poniższe metody:

- konstruktor inicjalizujący nazwę zajęć na podstawie nazwy przekazanej parametrem oraz tworzący pustą listę studentów
- *get* i *set* dla nazwy zajęć
- *`addStudent`*`(Student student)` - dodaje studenta na koniec listy
- *`fetchStudentNames()`* - zwracającą listę nazwisk studentów
- *`fetchStudentsAmount()`* - zwracającą obecną liczbę studentów
- *`showStudentNames()`* - wyświetlającą studentów w formacie: “Nr_na_liście imie nazwisko”
- *`removeStudent(int index)`* - usuwającą studenta z listy

- *`showStudentsWithSurname(String surname)`* - wyświetlającą studentów o podanym nazwisku
- *`fetchStudentsWithGradeHigherThan(double grade)`* - zwracającą listę studentów studentów ze średnią ocen wyższą niż ta podana w parametrze
- *`fetchStudentsGrades`* - zwracającą `String` zawierający listę ocen wszystkich uczniów w formacie (0 - 3,4,5; 1 - 4,5,2 itd.) *Wsk*. Skorzystaj z klasy `StringBuilder`.

Przetestuj wszystkie metody.

Zadanie 4

Utwórz 2 listy: zawodników oraz wyników. Lista wyników powinna pozwalać na przypisanie dowolnej liczby wyników dla każdego zawodnika (lista w liście). Przygotuj metodę zwracającą najwyższy czas dla każdego zawodnika oraz metodę wyświetlającą wszystkich uczestników oraz ich wyniki.

Zadanie 5

A) Stwórz klasę `Pacjent` posiadającą następujące pola chronione:

- `imie`
- `nazwisko`
- `PESEL`
- `nazwaOddzialu`
- `liczbaDniPobytu`
- `wiek`

Klasa posiada publiczne metody:

- konstruktor ustawiający liczbę dni pobytu pacjenta na 0 oraz wartości pozostałych pól na podstawie parametrów
- metody get i set
- *`sprawdzCzyPowyzej`* która zwraca *true*, jeżeli liczba dni pobytu pacjenta w szpitalu jest większa niż wartość podana jako parametr i *false* w przeciwnym wypadku
- *`dodajKolejnyDzien`* która zwiększa liczbę dni pobytu pacjenta o 1
- *`sprawdzOddzial`* która zwraca *true*, jeżeli pacjent znajduje się na oddziale o nazwie podanej jako parametr i *false* w przeciwnym wypadku
- *`toString`*
- *`wyswietl`* która wyświetla wszystkie dane pacjenta

❖ Utwórz listę pacjentów (ArrayList) i dodaj do niej kilku pacjentów.
❖ Zmień liczbę dni pobytu pacjentów na liście.
❖ Wyświetl listę pacjentów.

- Zdefiniuj klasę *`Szpital`* posiadającą pola:
- *`maxPacjentow`* `// maksymalna liczba osób (pacjentów) w szpitalu`
- *`listaPacjentow`* `// lista pacjentów – typu Pacjent`

Klasa posiada publiczne metody:

- konstruktor ustalający pola na podstawie parametrów

- konstruktor bezparametrowy
- metody get i set
- *`wyswietl`* która wyświetla maksymalne obłożenie szpitala, aktualną liczbę pacjentów w szpitalu oraz listę pacjentów
- *`wyswietlPacjentowZOdzialu`* która wyświetla imiona i nazwiska wszystkich pacjentów przebywających na oddziale podanym jako parametr
- *`zwrocSredniaDniPobytu`* która zwraca średnią liczbę dni pobytu pacjentów w szpitalu
- *`dodajPacjenta`* która pozwala dodać do listy pacjenta przekazanego jako parametr
- *`zwrocNajPobyt`* która zwraca wartość najdłuższego pobytu w szpitalu
- *`zwrocIlePowyzejWieku`* która zwraca liczbę pacjetnów powyżej wieku podanym jako parametr

❖ Utwórz obiekt `Szpital` wykorzystując listę z części A.
❖ Wyświetl informację o szpitalu.
❖ Wyświetl informację o średniej liczbie dni pobytu.
❖ Wyświetl ile pacjentów w szpitalu jest powyżej 65 roku życia.
❖ Wyświetl PESELe pacjentów przebywających najdłużej w szpitalu.

Zadanie 6

Napisz metodę, która jako parametr otrzyma listę napisów. Metoda ma zwrócić wszystkie wiersze, których długość jest równa maksymalnej długości napisu na liście. Metoda ma zwracać listę znalezionych wierszy jako obiekt klasy `List<String>`.
Wskazówka: skorzystać z metod `add` i `clear` interfejsu `List`.