Zadanie 1 (grupa poniedziałkowa: 13.11 (zajęcia wg. planu pon), grupy środowe: 20.11)
Celem zadania jest nauka podstaw Javy, posługiwanie się podstawowymi klasami, interfejsami oraz wykorzystaniem mechanizmu dziedziczenia. Dodatkowo w ramach zadania należy napisać jako ćwiczenie własną implementację struktury danych Mapa (analogia do java.util.Map), która jest abstrakcją funkcji (przyporządkowuje elementom jednego zbioru dokładnie jeden element drugiego zbioru).

Zadanie polega na napisaniu prostego systemu zarządzania hotelem. Interfejs programu jest tekstowy. Hotel posiada określoną liczbę pięter i określoną liczbę pokoi na każdym piętrze, każdy też posiada swoją cenę i pojemność. Pokoje w hotelu są numerowane liczbami całkowitymi, przy czym pierwsza cyfra oznacza nr piętra (np. 101 oznacza pierwszy pokój na pierwszym piętrze). Konfiguracja hotelu (liczba pięter, pokoje) jest albo zahardcodowana w programie (brak bonusów) albo wczytywana z pliku (bonus 1).

Program oczekuje na podanie komendy, które są następujące (wielkość liter w komendzie nie ma znaczenia):

- prices: listuje wszystkie pokoje wraz z cenami za dobę
- view: program prosi o podanie nru pokoju a następnie wypisuje wszystkie informacje o pokoju wraz z gościem (jeśli jakiś jest zameldowany). Jeśli podany nr pokoju jest błędny to wyświetlany jest komunikat błędu.
- checkin: pozwala na dokonanie rejestracji gościa w pokoju. Program prosi o podanie nru pokoju (jeśli podany pokój jest zajęty to wyświetlany jest komunikat błędu) a następnie prosi o podanie danych gości (gość główny oraz ew. dodatkowi jeśli pojemność pokoju jest większa) i oznacza pokój jako zajęty przez tego gościa, zapisuje też datę zameldowania (w przypadku niepodania daty zameldowania przyjmowana jest data bieżąca) oraz czas trwania pobytu oraz ew. informacje dodatkowe (opcjonalne).
- checkout: operacja odwrotna do checkin. Program prosi o podanie nru pokoju (jeśli podany pokój jest wolny lub podano nieprawidłowy nr pokoju to wyświetlany jest komunikat błędu) a następnie oznacza pokój jako wolny, dokonuje kalkulacji należności za nocleg, na bazie różnicy pomiędzy datą zameldowania i datą wymeldowania (bieżąca)
- list: listuje wszystkie pokoje wraz z informacjami o zajętości oraz gościach (dane gościa, data zameldowania, planowana data wymeldowania).
- save: bonus 2 – zapis bieżącego stanu do pliku CSV lub XLSX
- exit: wyjście z programu.

W ramach zadania należy dokonać implementacji struktury mapy – generyczna klasa MyMap implementująca interface Map z metodami (zbiór minimalny metod: put(K, V), get(K), List<Key> keys(), remove(K)). Klasa powinna wykorzystywać mechanizm *generics*. Do implementacji klasy MyMap można wykorzystać klasy: java.util.List, java.util.LinkedList, java.util.ArrayList. Implementacja powinna bazować na dwóch listach: liście kluczy i liście wartości.

Należy zapewnić pokrycie kodu implementacji wszystkich operacji testami.

Bonus 1:
Dane dotyczące konfiguracji hotelu mogą być zahardocowane. Jednak jeśli dane będą wczytywane z pliku .csv (lub XLSX) o strukturze: nr pokoju, opis, cena, dane gości opcjonalne, data zameldowania opcjonalnie – można zdobyć dodatkowe 0.5 stopnia.

Bonus 2:
Zapis aktualnego stanu hotelu do pliku CSV (lub XLSX) o strukturze j.w. – wszystkie pokoje wraz z gośćmi premiowany jest dodatkowym 0.5 stopnia.

Kryteria oceny (na 4.0):

- poprawność działania programu (na 3.0)
- projekt, w tym (na 3.5):
    - wykorzystanie mechanizmów obiektowości przy realizacji komend (wskazówka: dodanie obsługi nowej komendy nie powinno się wiązać ze zmianami w istniejącym kodzie)
    - podział projektu na moduły/pakiety/klasy
- Testy jednostkowe (logika działania plus testy struktury Map) (na 4)
- Analiza Sonar Cube oraz badanie pokrycia testami (do +0.5 jeśli projekt wyczyszczony z issues blocker/critical/major – poza sugestią zmiany System.out na loggery)

Zadanie 2 (koniec grudnia, grupa poniedziałkowa: 09.12, grupy środowe: 18 .12)
Zasady gry w pokera 5-kartowego dobieranego:

https://pl.wikipedia.org/wiki/Poker_pi%C4%99ciokartowy_dobierany
https://terazpoker.pl/poker-pieciokartowy-dobierany-5-card-draw/

Na podstawie powyższych zasad gry napisać grę konsolową dla dwóch do czterech osób w pokera. Komunikacja poprzez sieć sockety (serwer gry + klienci konsolowi). Należy zaprojektować protokół komunikacji, który będzie *„human readable”* i opierał się na wymianie tokenów kontrolnych w formacie np. „ID-GRY ID-GRACZA RODZAJ-RUCHU PARAMETRY-RUCHU”. Obsługa zdarzeń: nowa gra (z ustaleniem ante), dołączenie do gry, wymiana kart, status (sprawdzenie czyja kolej, czy koniec), wynik końcowy itd. Serwer powinien walidować poprawność ruchu gracza.

UWAGA: możliwe jest też wybranie innego wariantu lub nawet innej gry pod warunkiem dostarczenia wraz z zadaniem linka do przepisów danego wariantu.

Bonus 1
Wykorzystanie do komunikacji sieciowej klas z pakietu java.nio

Bonus 2:
Obsługa więcej niż jednej gry jednocześnie

Zadanie 3 (koniec semestru, grupa poniedziałkowa: 20.01, grupy środowe: 22.01)
Implementacja klasycznego problemu współbieżności: *Problem czytelników i pisarzy.* Należy napisać klasę Library, która ma metody: startReading, startWriting, stopReading, stopWriting wypisujące: liczbę czytelników/pisarzy w kolejce wraz ze szczegółami kto czeka/wchodzi/pisze/czyta, liczbę czytelników/pisarzy piszących wraz ze szczegółami kto czyta/pisze/wychodzi i zapewniającą, że:

- Do czytelni może wejść tylko jeden pisarz i zajmuje czytelnię na wyłączność,
- Do czytelni może wejść maks. 5 czytelników
- Ani czytelnik ani pisarz nie zostaną nigdy zagłodzeni jeśli działają w wątkach na tym samym priorytecie tj. ktokolwiek będzie chciał wejść do czytelni ma gwarancję tego, że się na to doczeka (pod warunkiem, że czas czytania/pisania pojedynczego wątku jest rozsądnie mały – dla ustalenia – nie większy niż 3 sek.) bez względu na liczbę wątków pisarzy i czytelników
- Czytelnia jest zorganizowana w sposób uczciwy: tzn. kolejkuje chętnych do jej skorzystania przed drzwiami – kto przyszedł do kolejki jako pierwszy ten ją opuści również jako pierwszy i wejdzie do czytelni – nikt kto przyszedł później nie wejdzie przed tym co przyszedł wcześniej.
- Przy każdym zdarzeniu (wejście/wyjście do/z czytelni) wypisywane jest:
    - Kto chce wejść (przyszedł do kolejki), wchodzi, wychodzi
    - kto jest w czytelni (wg. kolejności wejścia)
    - kto jest w kolejce przed wejściem do czytelni (wg. kolejności napływania)

Czytelnik w pętli nieskończonej wywołuje na czytelni operacje:

- startReading
- stopReading

Pisarze w pętli nieskończonej wywołuje na czytelni operacje:

- startWriting
- stopWriting

Pomiędzy wywołaniami tych metod może być wywoływana metoda Thread.sleep – jednak czas podany w sleep nie powinien mieć wpływu na poprawność działania czytelni (czyli na to czy dojdzie do zaglodzenia lub na to czy czytelnia wpuszcza chętnych w sposób uczciwy i zgodny z podanymi ograniczeniami).

Parametrem wywołania programu podczas uruchamiania ma być liczba czytelników, liczb czytelników. Opcjonalnie można przekazywać dodatkowe parametry takie jak czas na jaki wątek ma zostać uśpiony po wywołaniu każdej operacji.