Przygotuj poniższe aplikacje wykorzystując klasy i metody (OOP). Dla każdego zadania przygotuj osobną klasę, a w niej metodę lub metody rozwiązujące problem bądź realizujące określone zadanie zadanie.

1. Pamiętaj o poprawnym nazywaniu klas, metod, argumentów metod i zmiennych.
2. Korzystaj z domyślnego formatowania dostępnego w IntelliJ (CTRL + ALT + L)
3. Ustaw odpowiedni package dla projektu
4. Postaraj się tworzyć opisy metod i klas.

## Wyrażenia regularne

1. Przygotuj aplikację pobierającą od użytkownika dowolny ciąg znaków i sprawdzającą czy wprowadzona wartość jest wartością liczbową. Jeśli użytkownik wprowadził liczbę, sprawdź czy jest parzysta czy nieparzysta. Wyświetl komunikat informujący użytkownika o wprowadzeniu poprawnej lub błędnej liczby oraz o jej parzystości lub nieparzystości.
2. Przygotuj aplikację pobierającą od użytkownika ciąg znaków i sprawdzającą czy podany tekst jest poprawnym polskim kodem pocztowym (np. 85-155, 00-122)
3. Przygotuj aplikację pobierającą od użytkownika dowolny ciąg znaków i sprawdzającą czy wprowadzona wartość jest poprawnym loginem użytkownika. Za poprawny login uważamy tekst zawierający małe i duże litery oraz cyfry. Jego minimalna długość to 8 a maksymalna 16 znaków.
4. Przygotuj aplikację pobierającą od użytkownika dowolny ciąg znaków i sprawdzającą czy wprowadzona wartość zawiera słowo "ala".
5. Przygotuj aplikację pobierającą od użytkownika dowolny ciąg znaków i sprawdzjącą czy wprowadzona data jest poprawna. Za poprawną datę uważamy zapis w postaci "10.02.2018r.". Na potrzeby zadania nie weryfikujemy wartości dnia miesiąca. 45 to też poprawna wartość.
6. Przygotuj aplikację pobierającą od użytkownika dowolny ciąg znaków i sprawdzającą czy wprowadzony numer seryjny jest poprawny. Numer seryjny składa się z 3 dużych liter, 5 cyfr, 1 małej litery i 1 dużej litery np. VSD43281fA.
7. Numer seryjny oprogramowania ma postać "CFG&Y-TYH67-GH56T-UIO99-RY4RT", gdzie każdy blok może składać się z dużych liter lub cyfr. Bloki oddzielone są myślnikami "-". W numerze występuje dokładnie 5 bloków z wartościami. Przygotuj wyrażenie regularne sprawdzające numer seryjny.
8. Przygotuj wyrażenie regularne sprawdzające czy numer faktury VAT jest poprawny. Przykładowy numer faktury to "FV/1024/02/2018", gdzie
   FV - stały wpis
   / - stały znak rozdzielający sekcje
   1024 - kolejny numer faktury w danym miesiącu. Numer rozpoczyna się od 1
   / - stały znak rozdzielający sekcje
   02 - numer miesiąca w danym roku kalendarzowym
   / - stały znak rozdzielający sekcje
   2018 - rok
9. Przygotuj aplikację pobierającą od użytkownika dowolny tekst. Wprowadzony tekst powinien zostać podzielony na słowa (korzystając z metody split klasy String), a następnie program powinien wyświetlić statystyki wpisanego przez użytkownika tekstu.

```
Ilość słów: <ilość_słów>
Ilość wprowadzonych znaków: <ilość_znaków>
Średnia długość wprowadzonego słowa: <ilość_znaków>
Najdłuższe słowo: <ilość_znaków>
Najkrótsze słowo: <ilość_znaków>
```

Do testów użyj podanego zdania:

*"Drogi Marszałku, Wysoka Izbo. PKB rośnie. Z pełną odpowiedzialnością mogę stwierdzić iż realizacja określonych zadań stanowionych przez organizację. Dalszy rozwój jest ważne zadanie w większym stopniu tworzenie odpowiednich warunków aktywizacji. Często niezauważanym szczegółem jest to, że zakres i rozwijanie struktur pociąga za najważniejszy punkt naszych działań obierzemy praktykę, nie zaś teorię, okazuje się jasne."*

10. Przygotuj aplikację, która pobierze od użytkownika numer pesel, a następnie wykorzystując wyrażenia regularne (a w nich grupy) wyświetli informacje:

```
Data urodzenia:
dzień: <dzień>
miesiąc: <miesiąc> (zapisany słownie np. listopad)
rok: <rok>
Numer serii: <numer>
Płeć: M lub K
Cyfra kontrolna: <cyfra>
```

11. *Przygotuj aplikację, która pobierze od użytkownika ścieżkę do pliku tekstowego, a następnie wczyta do pamięci podany plik (korzystając z metody readAllLines klasy java.nio.file.Files). Pamiętaj o zabezpieczeniu się przed wprowadzeniem niepoprawnego pliku za pomocą klauzuli try {} catch {}. Pobrane przez użytkownika dane przetwórz za pomocą wyrażeń regularnych wykorzystując grupy tak aby pobrać odpowiednie dane do uzupełnienia klasy Person posiadającej pola name, surname, dateOfBirth, sex, NIN).

    Plik powinien mieć następującą zawartość:

```
Jan Kowalski M 11.12.1956 56121101177
Marianna Gąszczkiewicz F 14.02.1945 45021404509
Paweł Zalewski M 23.05.1992 92052305219
Michał Jak M 11.12.1986 86121102274
```

Obiekty Person dodaj do listy, a następnie korzystając z pętli while...do wyświetl informacje o każdej osobie. Format wyświetlania każdej osoby ma być następujący:
Wyświetlana osoba to Jan Kowalski, mężczyzna urodzony 11 grudnia 1956 roku o PESEL-u 56121101177.
Uwaga: Dane w pliku nie powinny zawierać białych znaków na początku i końcu linii

## Typy generyczne

12. Przygotuj program, który pobiera od użytkownika imiona i dodaje je do listy generycznej typu String. Koniec wprowadzania imion następuje po wprowadzeniu słowa "koniec". Następnie program korzystając z pętli foreach wyświetla po kolei wszystkie imiona podając na końcu każdego imienia ilość znaków z których się składa (np. Jan (3))

13. Przygotuj generyczną klasę Clipboard przechowującą obiekt typu T. Klasa powinna mieć metody insertIntoClipboard oraz getFromClipboard.
14. Przygotuj klasę BasePerson posiadającą pola name oraz surname, klasę Employee dziedziczącą z klasy BasePerson posiadającą pole companyName, klasę Teacher dziedziczącą z klasy BasePerson posiadającą pole school, degree. Wykorzystaj klasę Clipboard z poprzedniego zadania, ale zmodyfikuj ją tak, aby przyjmowała klasę BasePerson oraz jej wszystkie klasy potomne.

## Programowanie współbieżne i równoległe

15. Przygotuj aplikację uruchamiającą 1 wątek wyświetlający losową liczbę z zakresu 10-100. Wątek powinien wyświetlić liczbę 5 razy w odstępie 1 sekundy. Przygotuj wątek poprzez dziedziczenie z klasy Thread.
16. Przygotuj poprzednie zadanie poprzez implementację interfejsu Runnable.
17. Wykorzystaj pulę wątków (5 wątków). Niech każdy z wątków po uruchochomieniu losuje czas uśpienia z zakresu 1-5 sekund u usypia się. Po wybudzeniu każdy wątek powinien wyświetlić losową godzinę w postaci 12h13m14s34ms
18. Przygotuj aplikację pobierającą i zapisującą imię od użytkownika w zmiennej lokalnej. Przygotuj wątek sprawdzający co sekundę czy wprowadzone imię uległo zmianie. Jeśli imię uległo zmianie wątek powinien wyświetlić komunikat. “Stare imię: Jan, nowe imię: Małgorzata”
19. Przygotuj klasę Numbers posiadającą listę typu Integer. Przygotuj 2 wątki. Wątek o nazwie ThreadA niech losuje co sekundę 2 liczby całkowite z zakresu <0,1000), tworzy instancję klasy Numbers i uzupełnia w niej listę wylosowanymi liczbami. Wątek ThreadA po każdorazowym wylosowaniu liczb i utworzeniu klasy Numbers dodaje ją do kolejki (wykorzystaj kolejkę threadSafe). Liczby powinny być losowanie co sekundę. Po wylosowaniu liczb wątek wyświetla informacje: “Wylosowano liczby x i y”, gdzie za x oraz y należy wstawić wylosowane wartości. 2 wątek ThreadB pobiera obiekt z tablicy i sumuje liczby znajdujące się na liście wyświetlając komunikat “Suma liczb x i y wynosi z”, gdzie x i y to wylosowane wartości a z to obliczona syma.

## Testy jednostkowe

20. Przygotuj aplikację pobierającą od użytkownika imię i sprawdzającą ile samogłosek i ile spółgłosek jest podanym imieniu. Przygotuj klasę z 2 metodami sprawdzającymi ilość samogłosek i ilość spółgłosek. Przygotuj testy jednostkowe weryfikujące działanie dwóch metod.
21. Przygotuj klasę kalkulator posiadającą metody sum, divide, multiply, substract. Każda z metod ma przyjmować dwie liczby i zwracać wynik. Przygotuj odpowiednią implementację oraz testy jednostkowe sprawdzające działanie każdej z metod.
22. Przygotuj testy jednostkowe do 2 zadania.
23. Przygotuj testy jednostkowe do 4 zadania.
24. Przygotuj program obliczający BMI. Przygotuj testy sprawdzające działanie programu

## Optional

25. Przygotuj odpowiednie metody do przygotowanego projektu OptionalExercises w klasie OptionalExample. Masz 12 testów wykonujących określone zadania. Przygotuj

odpowiednie metody spełniające kryteria testów. Pamiętaj aby w każdym zadaniu używać tylko klasy Optional. Nie korzystaj z if. Usuwaj komentarz z kolejnych testów dodając właściwą implementację.

## Lambda Expression & Streams

26. Przygotuj tablice z 10 różnymi imionami. Posortuj tablicę według
    a. długości rosnąco
    b. długości malejąco
    c. pierwszej litery imienia rosnąco - wykorzystaj metodą charAt
27. Przygotuj interfejs funkcyjny, a w nim metodę, która będzie pobierać 2 zmienne typu String i zwracać Boolean. Następnie przygotuj metodę statyczną betterString przyjmującą jako argumenty 2 zmienne typu String oraz predykat (przygotowany wcześniej interfejs funkcyjny) tak aby poniższy kod się kompilował i działał.

```
String string1 = "test";
String string2 = "testtest";
String longer = StringUtils.betterString(string1, string2,
(s1, s2) -> s1.length() > s2.length());
String first = StringUtils.betterString(string1, string2,
(s1, s2) -> true);
System.out.println(longer); // zwraca dłuższy wyraz
System.out.println(first); // zwraca zmienną string1
```

28. Przygotuj klasę BankAccount posiadającą pola name and balance. Utwórz 7 instancji klasy BankAccount ze stanem konta odpowiednio 1000,2000,3000,4000, 5000,7500 i 10000 złotych. Dodaj je do listy generycznej. Korzystając ze strumieni znajdź wszystkie konta które mają zgromadzonych środków więcej niż 5000zł.
29. Wykorzystując przygotowaną listę z poprzedniego zadania za pomocą strumieni znajdź konto, które ma najwięcej zgromadzonych środków.
30. Za pomocą strumieni i klasy IntStream wygeneruj liczby losowania lotto.
31. Wykorzystując klasę Stream wygeneruj 10 losowych liczb z zakresu <0,100), następnie wymnóż każdą wylosowaną liczbę za pomocą map przez 2 i korzystając z forach wyświetl każdą wylosowany i wymnożoną liczbę.
32. Utwórz tablicę typu int[]. Wylosuj 20 wartości z przedziału <100,200> i wyświetl tablicę. Wykorzystując stream posortuj tablicę z wygenerowanymi wartościami rosnąco, a następnie wyświetl tablicę, *Posortuj tablicę malejąca, a następnie ją wyświetl.

## Logowanie

33. Utwórz aplikację, która będzie pobierała od użytkownika imię i wyświetlała je na konsoli korzystając z log4J (nie używaj w tym celu polecenia System.out.print*)
    a. Dodaj do projektu plik konfiguracyjny log4j
    b. Utwórz w klasie Logger odpowiedzialny za logowanie w klasie
    c. Zaloguj wprowadzone imię przez użytkownika.
    d. Dodaj logi w trybie INFO informujące u uruchomieniu i zamknięciu aplikacji.
    e. Ustaw poziom logowania na info i sprawdź czy logi pojawiły się w konsoli.

34. Zmodyfikuj konfigurację log4j z poprzedniego zadania tak, aby logi były dodatkowo zapisywane do pliku
35. Przygotuj aplikację która demonstruje poziomy działanie poziomów logowania. Korzystając z loggera zaloguj dowolny komunikat dla każdego poziomu oddzielny. Zmodyfikuj plik konfiguracyjny tak, aby każdy komunikat został zapisany do oddzielnego pliku. Przykładowo poziom logowania TRACE powinien zostać zapisany do pliku trace.log, DEBUG do debug.log itd.
36. Dodaj mechanizmy logowania do zadania 21 dla każdej metody wykonującej działanie arytmetyczne. Każda metoda powinna zalogować informacje o rozpoczęciu wykonywania obliczeń wraz z podanymi argumentami oraz informację o zakończeniu obliczeń razem z wynikiem. Zmodyfikuj logger tak aby logi zapisywane były w formacie html.