Zadania 1

1.1

Klasa `pl.sda.calc.MathOperations` ma metody:

- `MathOperations(int a, int b)` – konstuktor przyjmujący argumenty operacji
- `multiply` - mnożenie
- `subtract` - odejmowanie
- `divide` – dzielenie
- `power` - potęgowanie
- `throwWhenEqual` – rzuca wyjątkiem, jeśli argumenty są równe
- `min` – zwraca mniejszą liczbę
- `max` – zwraca większą liczbę
- `subtractResultHaveToBeGreaterThanZero` – rzuca wyjątkiem jeśli różnica argumentów jest ujemna, jeśli nie jest po prostu zwraca różnicę
- `sum` – dodawanie

a) Napisz testy jednostkowe do wszystkich wymienionych wyżej metod, sprawdź czy wszystkie przechodzą. Jeśli nie wszystkie przechodzą (może być tam jeden błąd) popraw kod klasy MathOperations i sprawdź czy teraz testy teraz przechodzą 😊

b) Pamiętaj o używaniu odpowiednich nazw metod testujących i rozdzielaniu części given, when, then

c) Zmień kod tak, aby w metodzie `divide` było sprawdzenie czy b jest mniejsze od 0, jeśli tak wyrzuć wyjątek „unchecked” z odpowiednim komunikatem, dostosuj testy

d) Dodaj metodę `boolean isEven()` i `boolean isOdd()` sprawdzające czy obie podane liczby są parzyste lub nie, dodaj testy

e) Dodaj metodę `int mod()` zwracającą resztę z dzielenia a przez b

f) *zamień int na BigDecimal w przynajmniej 5 operacjach

1.2

Klasa `pl.sda.doctor.PatientService` obsługuje kolejkę w przychodni. Ma 3 metody:

- **`boolean`** `addPatient(Patient patient)` – jeśli kolejka jest większa niż 5 osób zwraca false w przeciwnym wypadku dodaje pacjenta do kolejki
- `List<String> removeAllRemaining()` – w tej chwili zwraca pustą listę, Twoim zadaniem będzie ją zaimpelentować, o tym dalej
- `String nextPatient()` – „zdejmuje" kolejnego pacjenta z kolejki:
  - jeśli stan pacjenta jest dobry (`Condition.GOOD`) trafia do dr. Gatesa
  - jeśli stan pacjenta jest zły (`Condition.BAD`) trafia do dr. Jobsa
  - jeśli stan pacjenta jest bardzo zły (`Condition.VERY_BAD`) trafia do dr. Wozniaka

Manualnie działanie klasy możesz przetestować w `pl.sda.doctor.Main`

a) Napisz testy jednostkowe dla klasy PatientService, nie zapomnij o obsłużeniu wyjątku, który może wyrzucić metoda nextPatient w przypadku braku pacjentów w kolejce!
b) Zaimplementuj metodę `removeAllRemaining()` – powinna zwrócić listę Stringów zawierającą połączone imiona i nazwiska (np. „Jan Kowalski", „Maciej Nowak" itd.) i jednocześnie opróżnić kolejkę
c) `PatientService` ma na „sztywno" przypisaną długość (`MAX_QUEUE_LENGTH = 5`) kolejki do sprawdzenia przy dodawaniu, zmień kod tak, aby długość kolejki podawało się przez konstruktor `PatientService,` dostosuj testy

1.3

`pl.sda.shop.impl.DefaultAvailableShopsService` zawiera implementację serwisu wyszukującego otwarte sklepy, ma metody:

- `public List<Shop> findOpenShopsWithNameContaining(String name)` – zwraca wszystkie otwarte sklepy mające w nazwie argument name
- `public List<Shop> findOpenShops()` – zwraca wszystkie otwarte sklepy
- dodatkowo klasa implementuje interfejs posiadający wyżej wymienione metody

Manualnie działanie serwisu możesz przetestować w `ShopsMain`

a) Dokończ pisanie testów w `DefaultAvailableShopsServiceImplTest`
b) Dodaj implementację do `PolandAvailableShopsServiceImpl` uwzględniając, że jeśli dzisiaj jest niedziela to sklepy są zamknięty – ten warunek powinien być sprawdzany najpierw, a następnie powinna być wywoływana metoda z klasy wyżej (super.nazwaMetody(…)), pamiętaj o testach