Geodesic App

Dokumentacja techniczna

Agnieszka Banaś
Krzysztof Doniec
Filip Gaida
Piotr Gazda
Sylwia Molitor
Dawid Rudy

1. Sposób wykonania programu

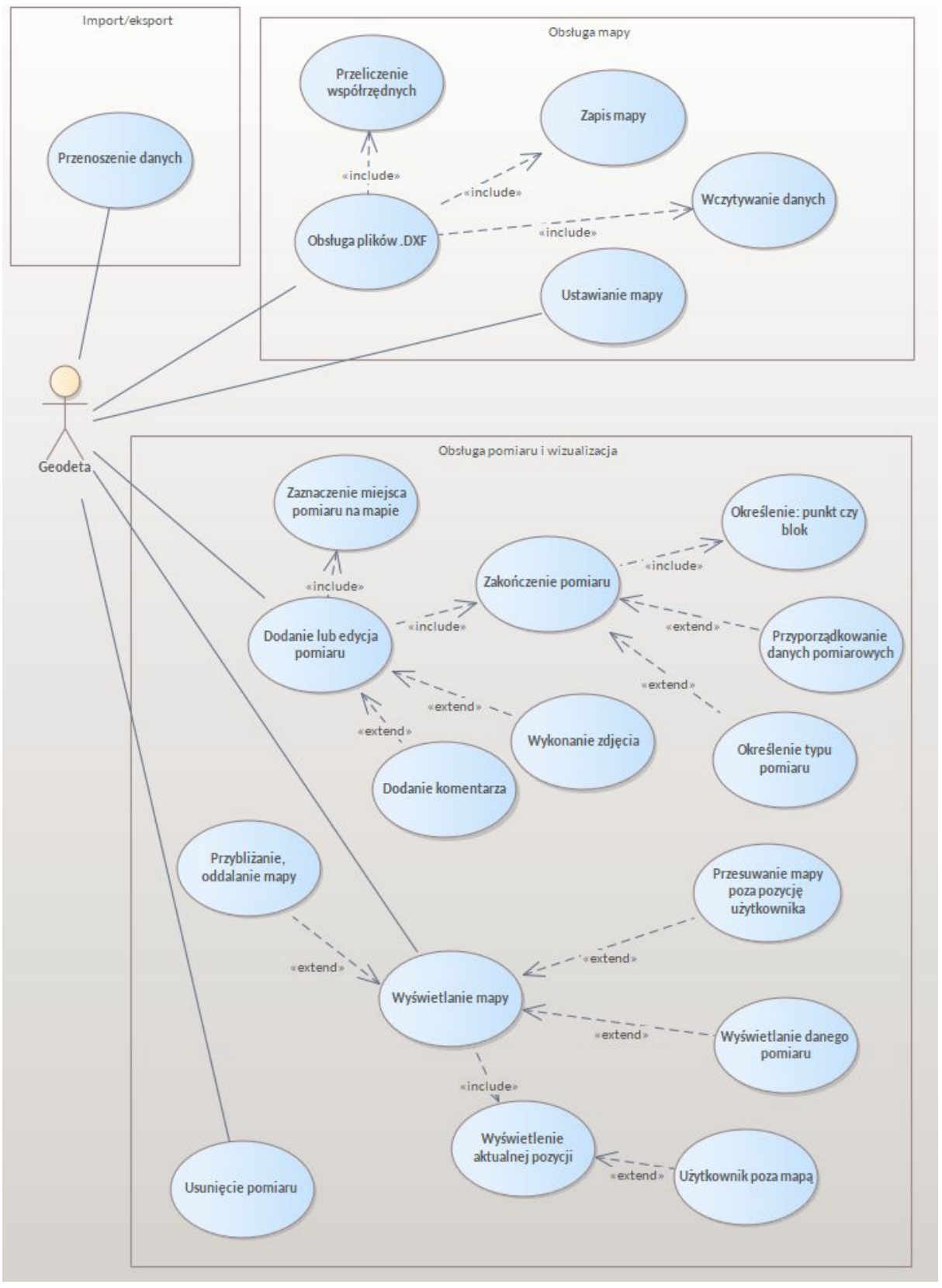
Program jest aplikacją stworzoną przy pomocy języka Kotlin do działania na systemie operacyjnym Android 7.1 i wyżej. Funkcjonalności są podzielone na klasy zgodnie ze wzorcem MVC. W programie oprócz standardowego API Androida została użyta zewnętrzna biblioteka Kabeja używana do obsługi plików DXF. Program umożliwia wykonywanie czynności związanych z pracą geodety: wyświetlanie mapy geodezyjnej, dodawanie nowych pomiarów i edycja już istniejących.

2. Realizowane funkcjonalności i diagram przypadków użycia.

Funkcjonalności realizowane przez program można podzielić na trzy główne obszary. Pierwszym i najprostszym z nich jest import oraz eksport plików DXF pomiędzy programem, a lokalizacjami w systemie plików urządzenia.

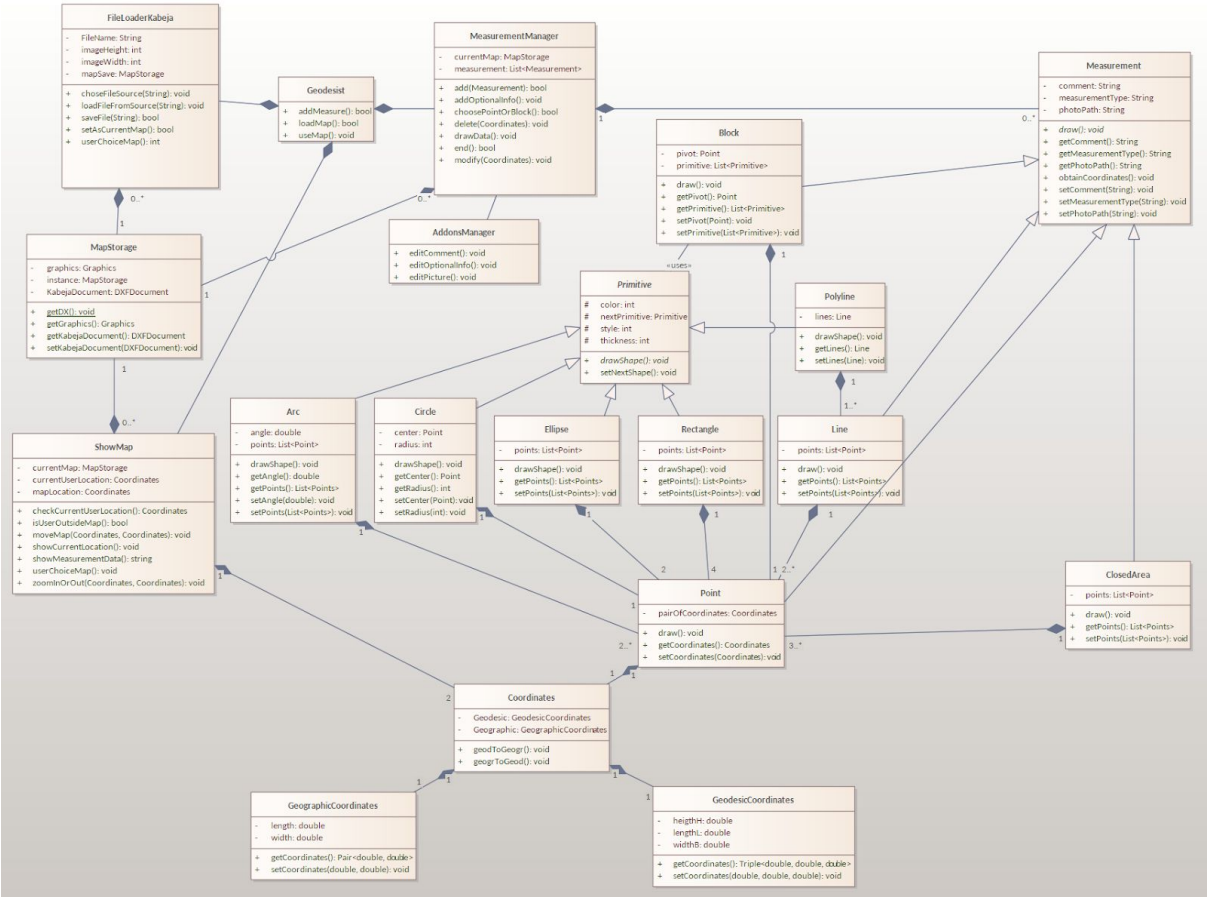
Drugim znacznie bardziej złożonym obszarem są funkcjonalności związane z obsługą przekazanej do programu mapy. Obsługa plików mapy zapisanych w formacie DXF zawiera w sobie możliwości odczytywania danych zawartych w pliku i przeliczania współrzędnych geograficznych na geodezyjne i na odwrót. Zmodyfikowana przez aplikacje mapa zapisywana jest do nowego pliku.
Użytkownik ma możliwość wyboru aktualnie wyświetlanej mapy.

Trzecia grupa to funkcje umożliwiające obsługę pomiarów i ich wizualizację.
Pierwszą podgrupę stanowią działania związane z dodawaniem i edycją pomiarów.
Możliwe jest zaznaczenie miejsca pomiaru na mapie, a po jego zakończeniu następuje określenie czy dany pomiar ma być punktem czy blokiem, doprecyzowanie jego typu i przyporządkowanie konkretnych danych liczbowych. Pomiar może być wzbogacony o zdjęcie i komentarz.
Druga podgrupa to możliwości związane z wyświetlaniem mapy. Widok mapy można przybliżać i oddalać oraz przesuwać poza obszar w pobliżu aktualnej pozycji użytkownika. W widoku mapy można wyświetlić dany pomiar oraz aktualną pozycję użytkownika, sprawdzając czy znajduje się on w obrębie obszaru przedstawionego przez używaną mapę.
Ostatnią funkcjonalnością definiowaną w tej grupie jest możliwość usuwania pomiaru.


Import/eksport
Przenoszenie danych
Obsługa mapy
Przeliczenie współrzędnych
Zapis mapy
«include»
«include»
Obsługa plików .DXF
Wczytywanie danych
«include»
Ustawianie mapy
Geodeta
Obsługa pomiaru i wizualizacja
Zaznaczenie miejsca pomiaru na mapie
Określenie: punkt czy blok
«include»
Zakończenie pomiaru
«include»
Dodanie lub edycja pomiaru
«include»
«extend»
Przyporządkowanie danych pomiarowych
«extend»
«extend»
«extend»
Wykonanie zdjęcia
Określenie typu pomiaru
Dodanie komentarza
Przybliżanie, oddalanie mapy
Przesuwanie mapy poza pozycję użytkownika
«extend»
«extend»
Wyświetlanie mapy
«extend»
Wyświetlanie danego pomiaru
«include»
Wyświetlenie aktualnej pozycji
«extend»
Użytkownik poza mapą
Usunięcie pomiaru

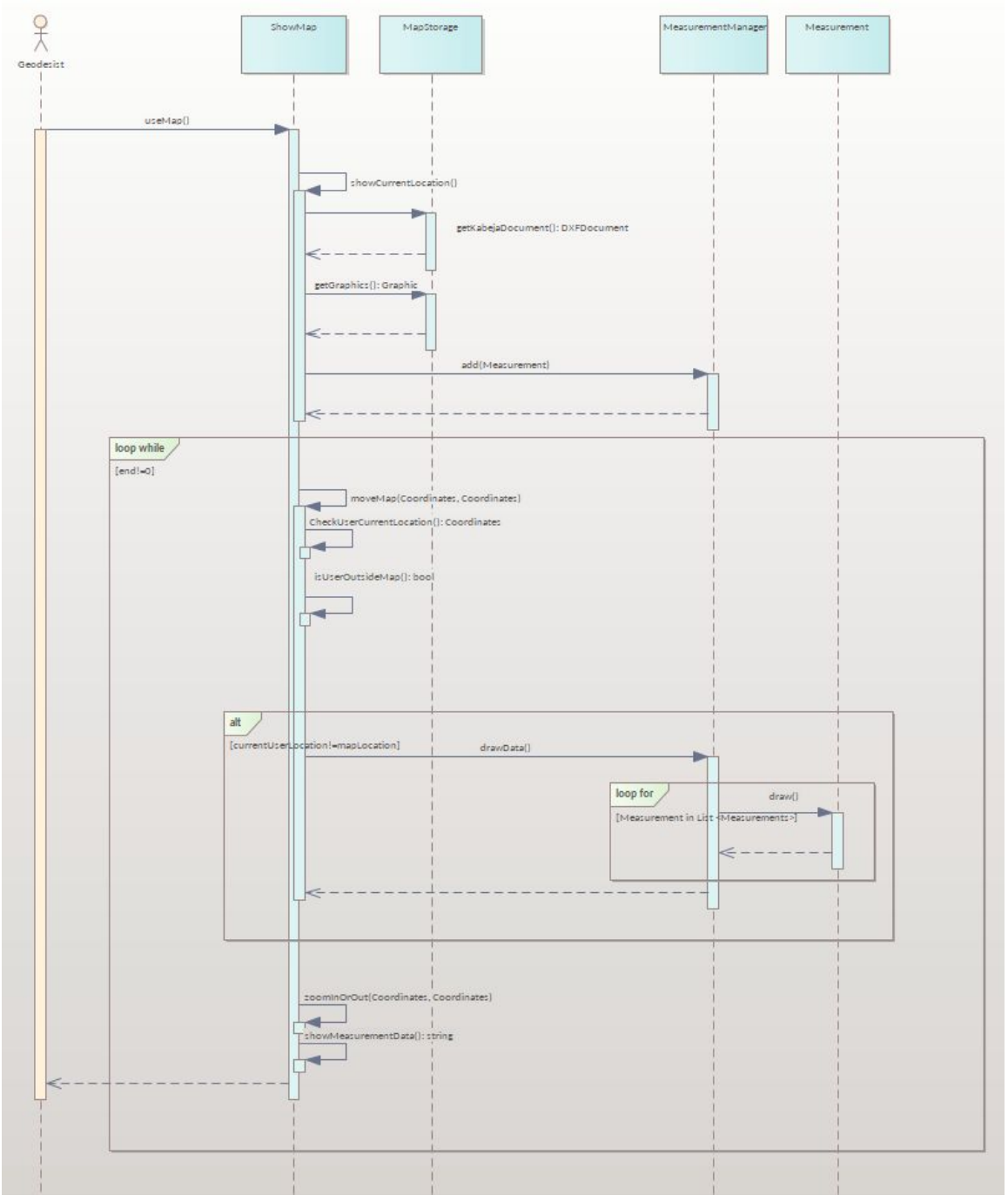
3. Klasy i zależności między nimi - diagram klas.

Główną klasą jest klasa Geodesist, która posiada metody do wykonywania trzech głównych operacji w programie - uzyskiwania pliku mapy, operacji na pomiarach i wyświetlania mapy wraz z pomiarami.
Do załadowania pliku wykorzystywana jest klasa FileLoaderKabeja, a do jego przechowywania klasa MapStorage, realizująca wzorzec projektowy Singleton.
Do wyświetlania mapy służy klasa ShowMap.
Do obsługi pomiarów służy klasa MeasurementManager, której funkcjonalności są uzupełniane w klasie AddonsManager.
Do przechowywania typów pomiarów służą implementacje abstrakcyjnej klasy Measurement, do których zaliczają się klasy Block, Line, Point oraz ClosedArea. Ponieważ klasa Block może reprezentować złożone geometrycznie elementy, korzysta ona z abstrakcyjnej klasy Primitive, która przedstawia podstawowy obiekt geometryczny. Jej implementacjami są klasy Arc, Circle, Ellipse, Rectangle i Polyline. Połączone są one ze sobą za pomocą wzorca projektowego Łańcuch zobowiązań.
Klasa Point korzysta z klasy Coordinates by przechowywać współrzędne punktu, tutaj następuje także przeliczanie pomiędzy dwoma rodzajami współrzędnych. Klasa Coordinates z kolei używa klas GeographicCoordinates i GeodesicCoordinates by przechowywać odpowiednio współrzędne geograficzne i geodezyjne.

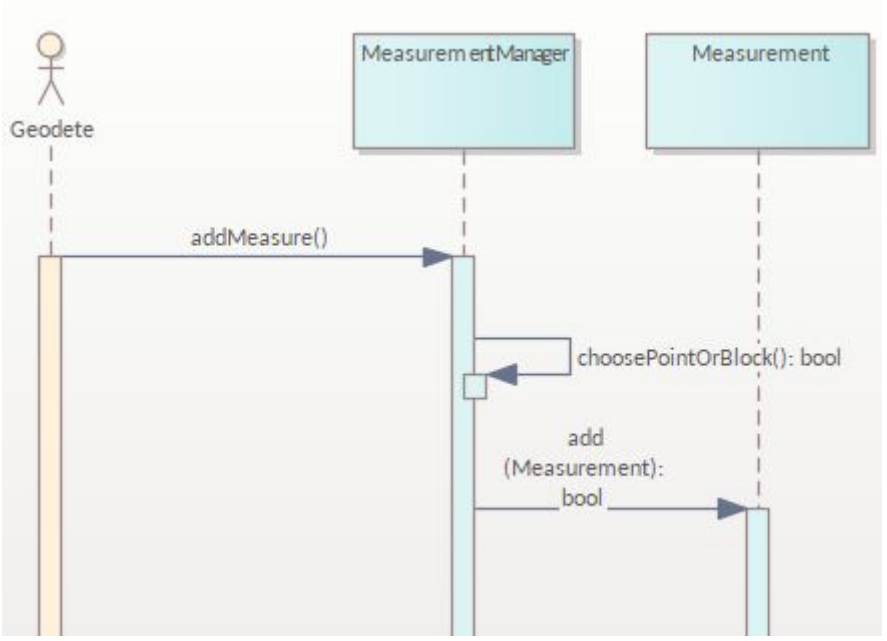


4. Działanie programu w czasie rzeczywistym - diagramy sekwencji.

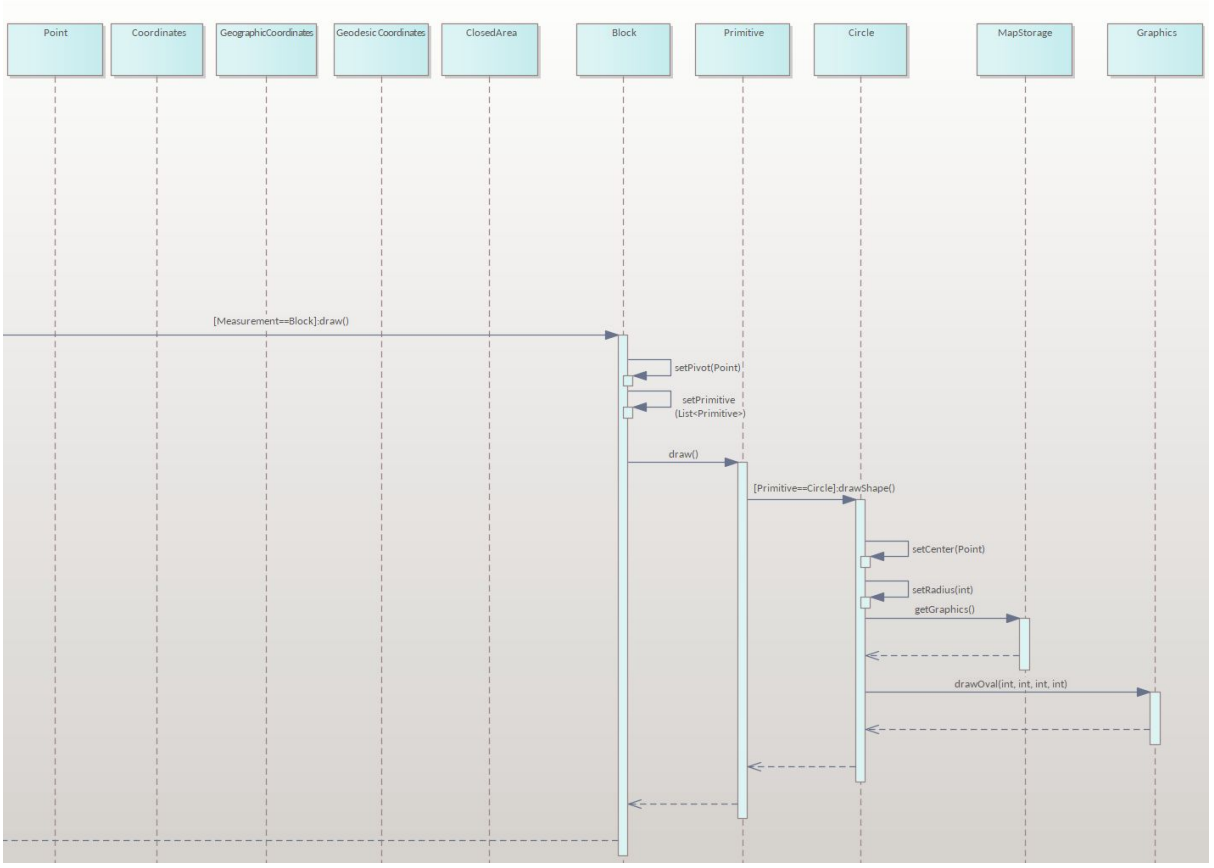
Pierwszy diagram reprezentuje sposób wyświetlania mapy. Działanie zaczyna się od wywołania przez obiekt klasy Geodesist metody useMap(). Powoduje to wywołanie metody showCurrentLocation() w należącym do obiektu klasy Geodesist obiekcie klasy ShowMap. Metoda ta wywołuje metodę getKabejaDocument() na rzecz statycznej klasy MapStorage, a następnie metodę getGraphics(). Później następuje wywołanie metody add() na rzecz obiektu klasy MeasurementManager. W następnej kolejności wykonywany jest iteracyjny fragment programu w ramach którego następuje sprawdzenie pozycji użytkownika i dostosowanie widoku mapy, po czym wyrysowanie elementów listy pomiarów. Działanie zwieńczone jest wywołaniem metody umożliwiającej przybliżanie bądź oddalanie widoku. Możliwe jest też wyświetlenie szczegółowych danych danego pomiaru za pomocą metody showMeasurementData().



Drugi diagram przedstawia w jaki sposób program realizuje dodanie przez użytkownika nowego pomiaru. Na początku z obiektu klasy Geodesist wywoływana jest metoda addMeasure(). Powoduje ona działanie w obiekcie klasy MeasurementManager polegające na wywołaniu metody choosePointOrBlock(),
a następnie metody add(). Jako parametr tej metody przekazywany jest obiekt klasy dziedziczącej z abstrakcyjnej klasy Measurement.

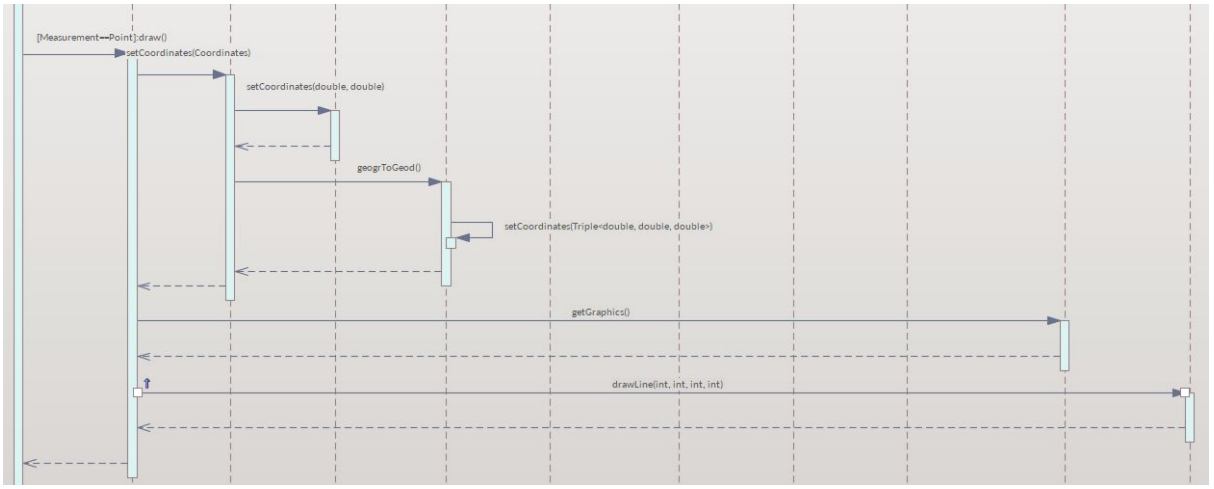


Na jego rzecz wywoływana jest polimorficzna metoda draw(). W przypadku dodawania pomiaru typu Block, w tej metodzie następuje wywołanie metod odpowiedzialnych za ustawienie wartości koniecznych do narysowania kształtów reprezentujących, czyli osi (Pivot) oraz ustawienie listy z obiektami geometrycznymi. Następnym krokiem jest wywołanie metody draw(), która powoduje wywołanie metody drawShape() dla kolejnych obiektów klas dziedziczących z abstrakcyjnej klasy Primitive. W niej następuje ustawienie wartości koniecznych do wyrysowania konkretnych elementów geometrycznych, np. środek okręgu, a następnie wywołanie odpowiedniej metody bibliotecznej do fizycznego narysowania przygotowanego elementu na ekranie.

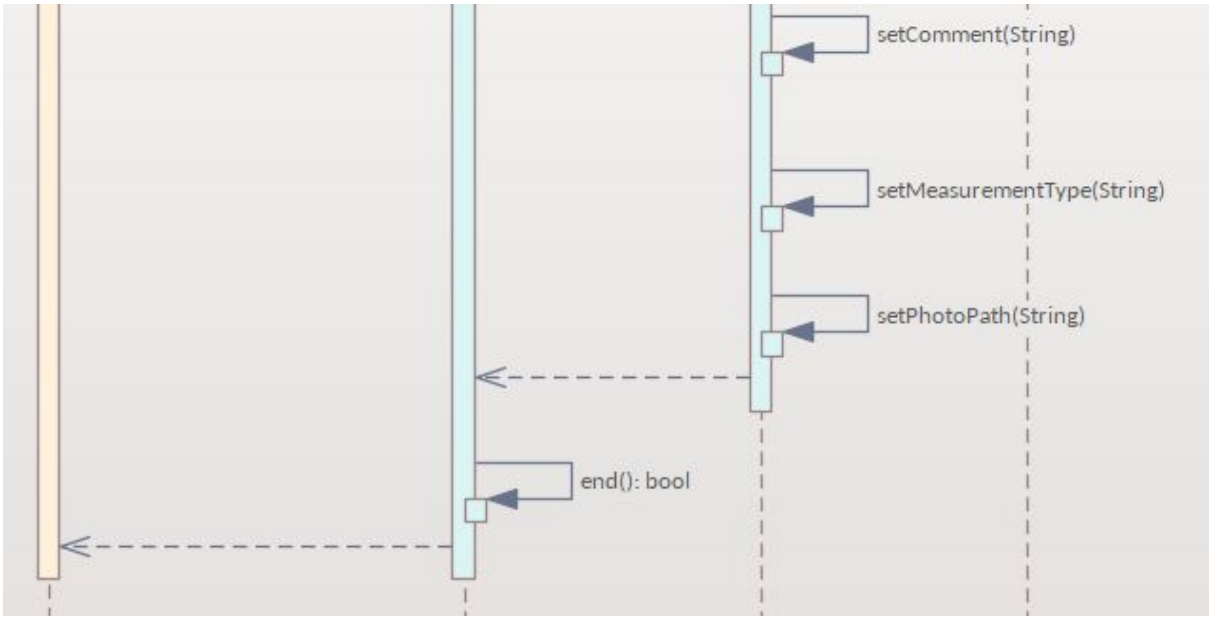


W sytuacji dodawania pomiaru typu ClosedArea lub Line z metody add() wywoływana jest metoda draw(), a z niej metoda setPoints(). Dalsza część procesu dodawania jest taka sama jak dla typu Point.

Proces dodawania pomiaru typu Point rozpoczyna się od wywołania metody draw(). Następnie wywoływana jest metoda setCoordinates(), która ustawia wartości współrzędnych dodawanego punktu. Ponieważ występują one w postaci obiektów klas GeographicCoordinates oraz GeodesicCoordinates, najpierw wywoływana jest metoda o takiej samej nazwie dla obiektu klasy GeographicCoordinates, a następnie metoda geogToGeod() która konwertuje wartości współrzędnych geograficznych do wartości współrzędnych geodezyjnych. Ta część programu kończy się poprzez wywołanie metody setCoordinates() dla obiektu klasy GeodesicCoordinates.
Po wykonaniu ustawienia wartości współrzędnych miała być wywoływana metoda getGraphics(), a następnie drawLine(), które rysują linię. Niestety, okazało się, że biblioteka nie działa zgodnie z dokumentacją i metoda getGraphics() nie działała w Kotlinie, dlatego zdecydowaliśmy się ją zmienić na metodę związaną z Android Graphics.



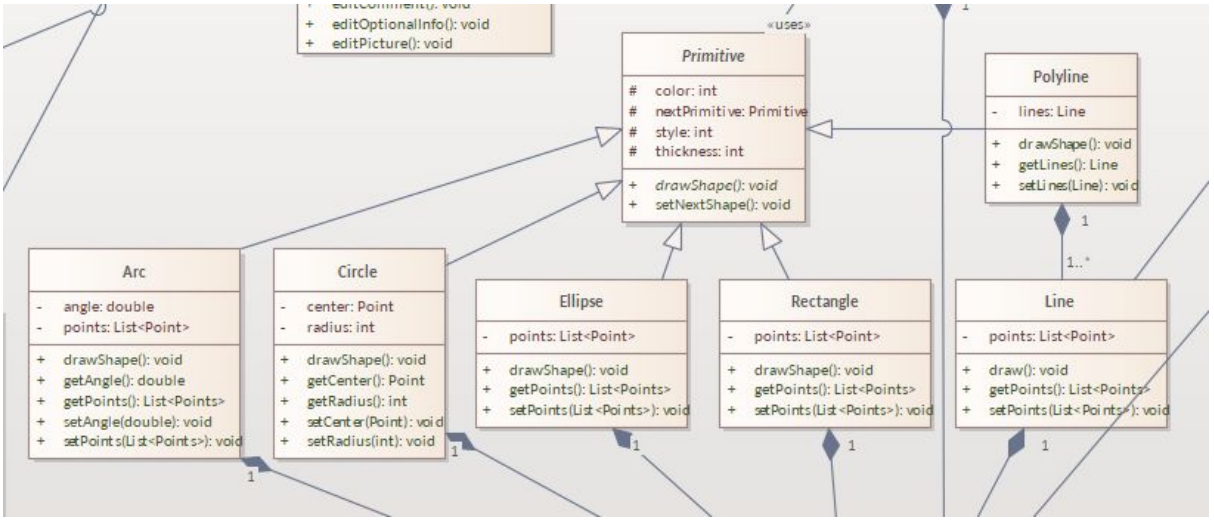
Po wykonaniu rysowania, wywołane zostają metody setComment(), setMeasurementType() oraz setPhotoPath() w celu dodania odpowiednio komentarza, informacji o typie pomiaru oraz ścieżki do zapisu zdjęcia dla danego pomiaru. Potem następuje powrót do metody addMeasure() i wywołanie metody end(), która przekazuje informację o udanym zakończeniu pomiaru.



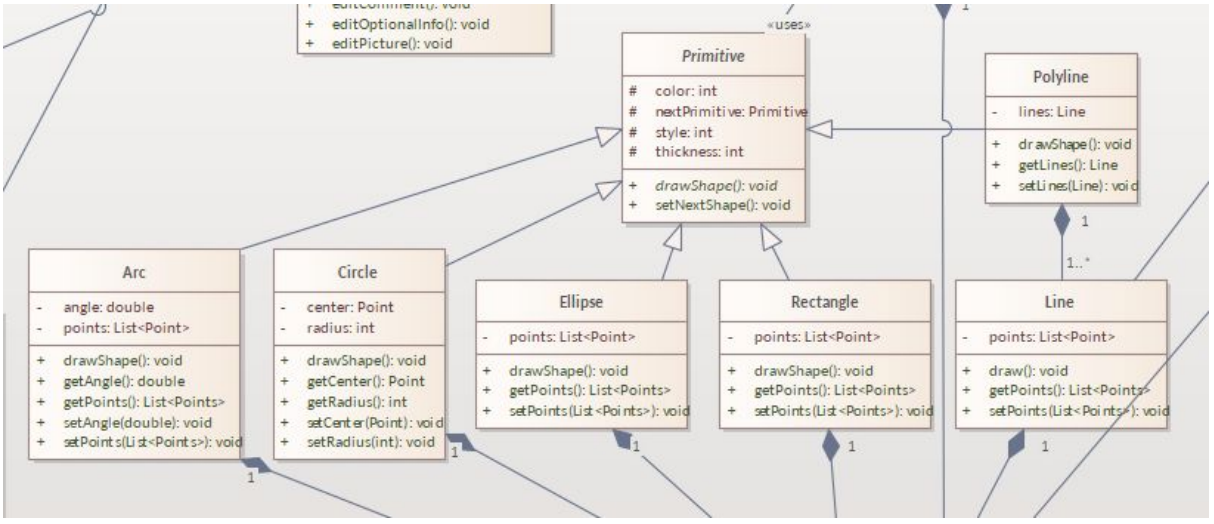
5. Użyte wzorce projektowe.

W naszym programie zdecydowaliśmy się użyć dwóch powszechnie znanych wzorców projektowych - Singleton i Łańcuch Zobowiązań (ang. Chain of responsibility).

Singleton to kreacyjny wzorzec projektowy, którego celem jest ograniczenie możliwości tworzenia obiektów danej klasy do jednej instancji i zapewnienie globalnego dostępu do tworzonego obiektu. W przypadku naszego programu był on przydatny, aby stworzyć jeden obiekt przechowujący mapę i zapewnić dostęp do niego operującym na nim klasom.



Łańcuch zobowiązań to czynnościowy wzorzec projektowy. Zakłada on stworzenie oddzielnej klasy dla każdej procedury obsługi żądania dziedziczącej po pewnej klasie bazowej. Obiekt każdej z procedur może posiadać wskaźnik na następny obiekt, tworząc w ten sposób łańcuch procedur przetwarzania. Aby zrealizować zadanie, wykonujemy metodę na pierwszym elemencie łańcucha. Gdy nie potrafi on przetworzyć żądania, przekazuje je następnemu elementowi w liście. W naszym projekcie wykorzystaliśmy ten wzorzec przy rysowaniu bloków, składających się z różnych kształtów - prymitywów.



6. Dokonana implementacja

Zdecydowaliśmy się na implementację wyświetlania mapy i dodania pomiaru - założyliśmy rysowanie wyłącznie linii i punktów. Z tego powodu na wyświetlanej na ekranie mapie nie są zauważane kształty składające się z wielu linii. Założyliśmy także, że wszystkie linie są jednakowej grubości.