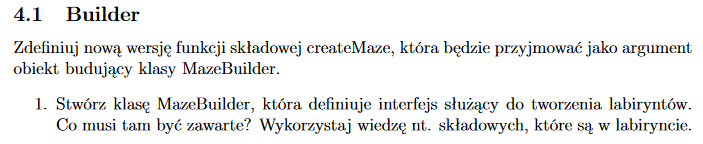
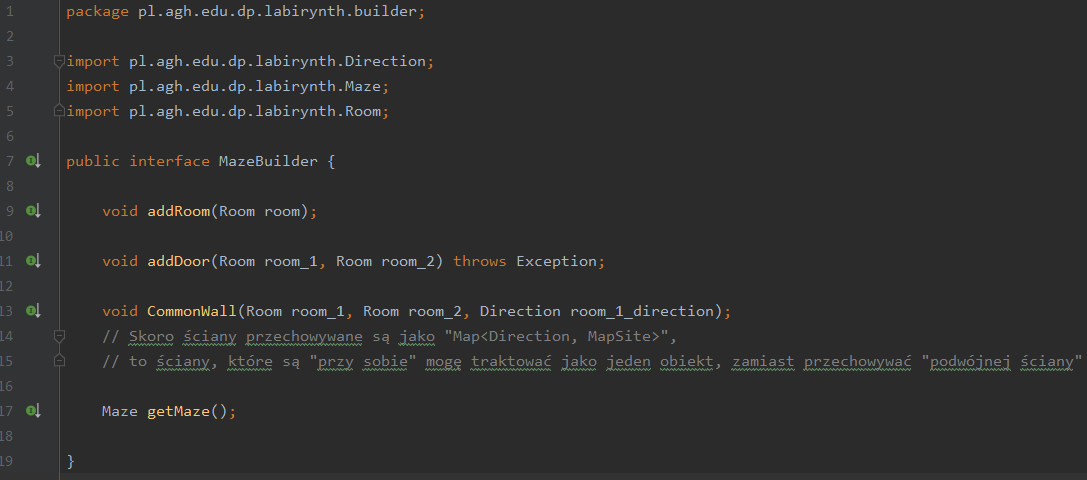
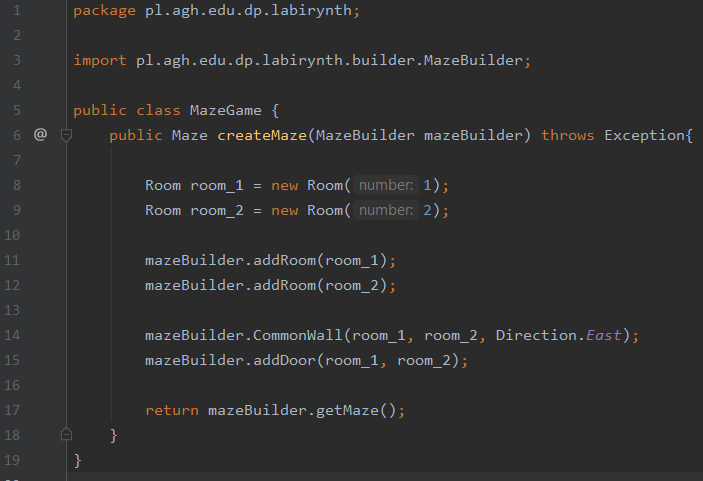
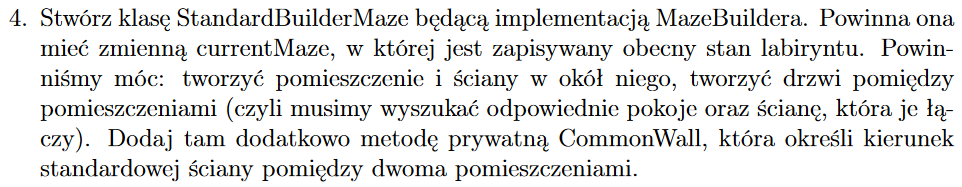
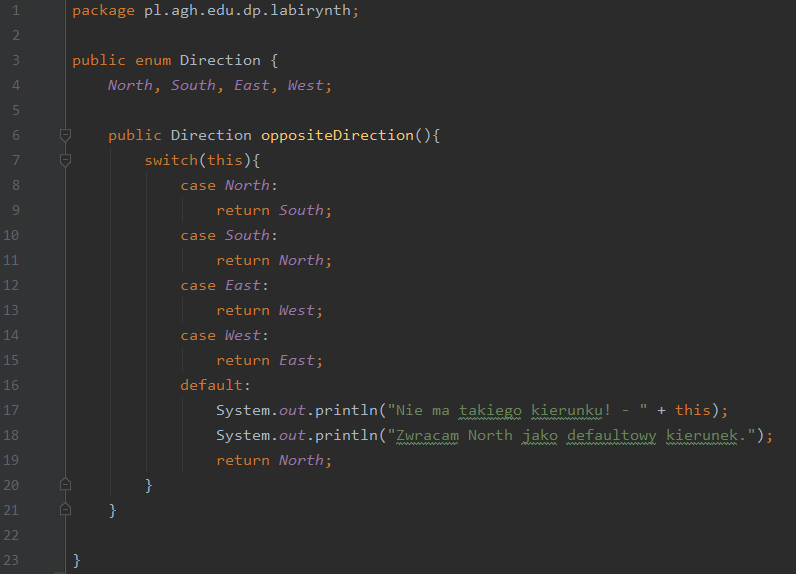
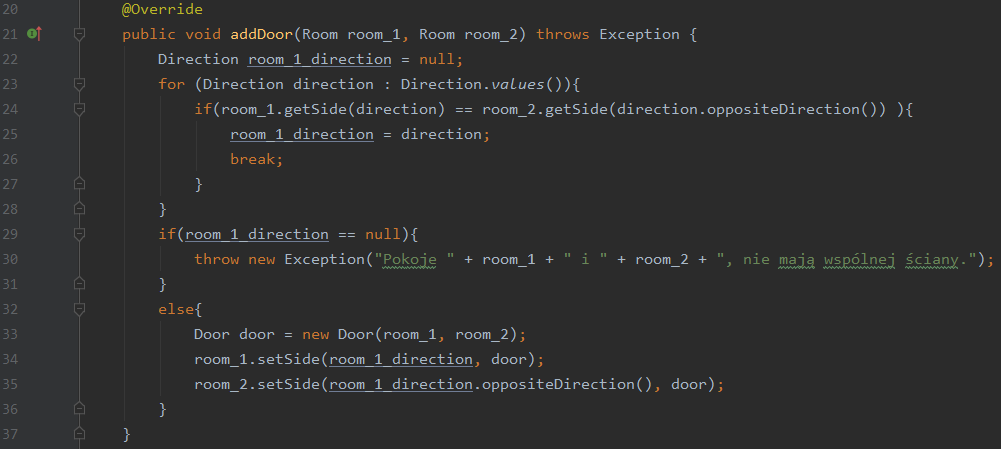
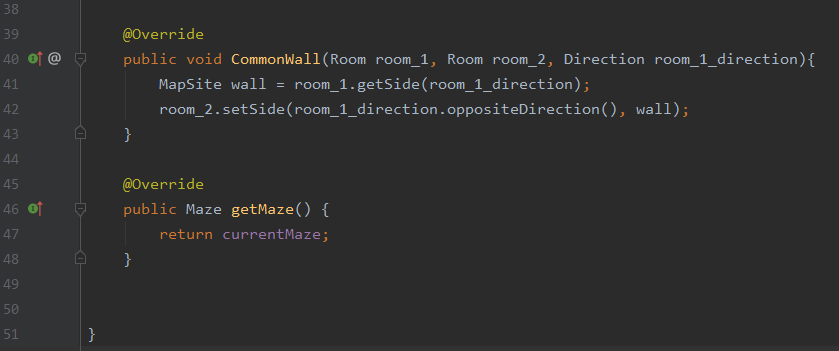
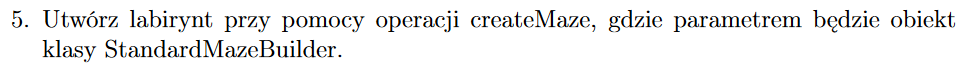
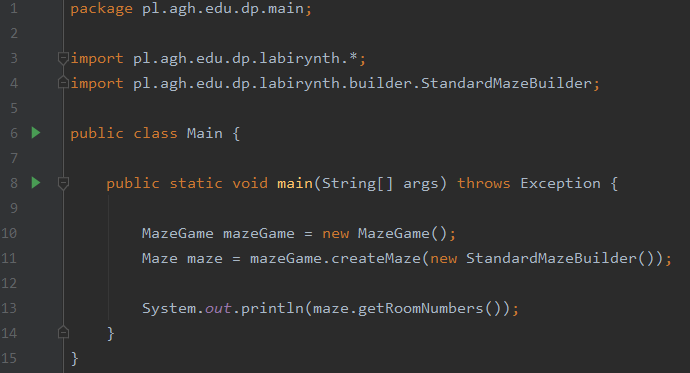
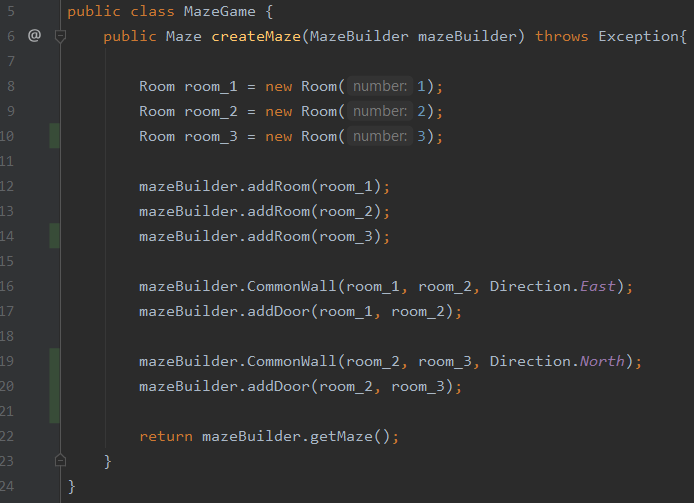
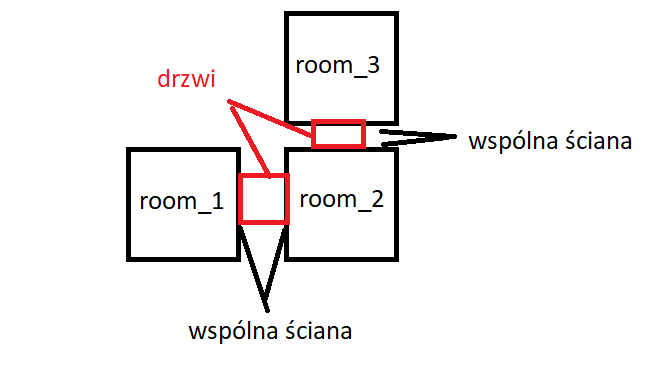
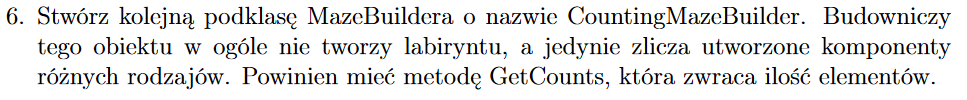
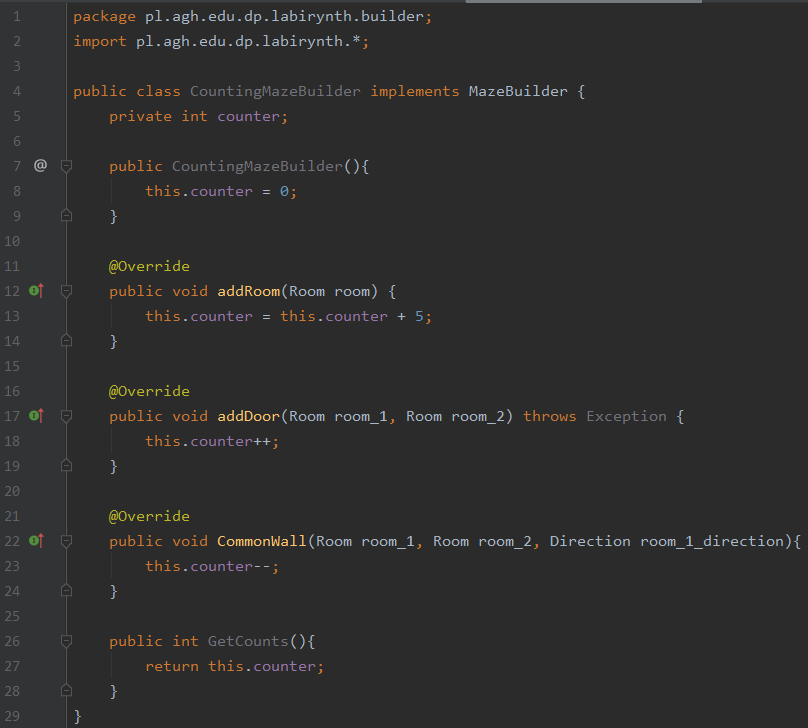
Wojciech Kosztyła  
nr. indeksu 305 338

Projektowanie obiektowe

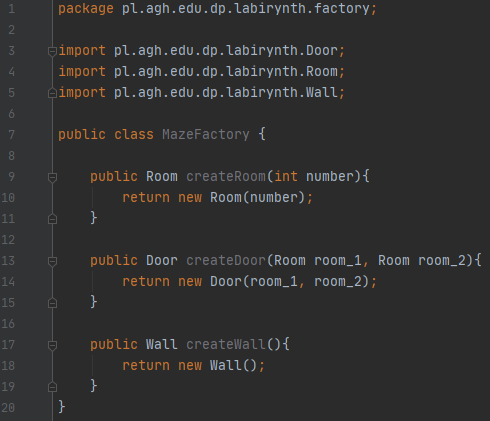
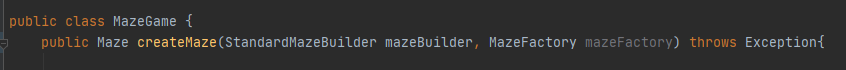
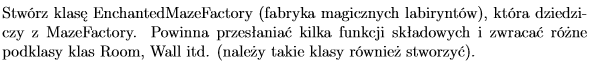
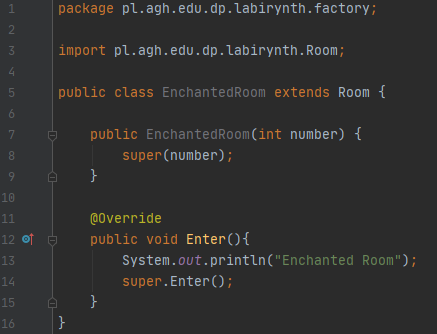
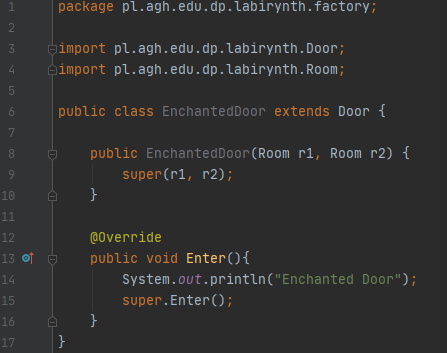
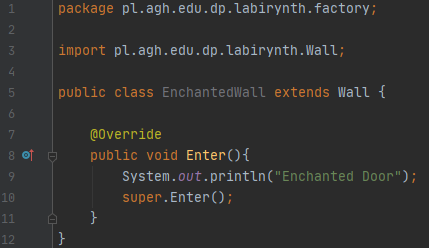
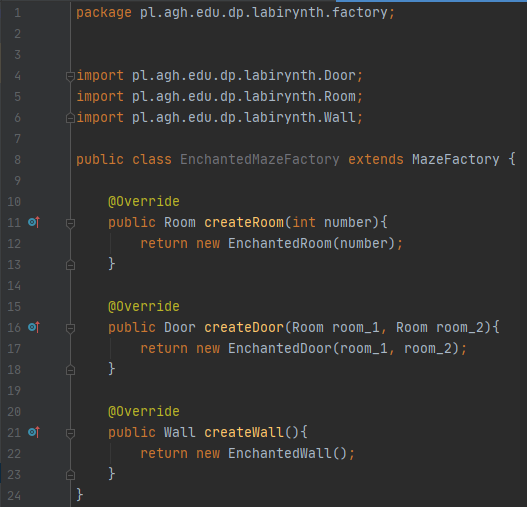
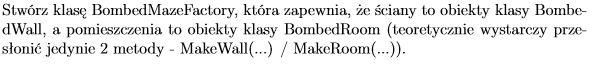
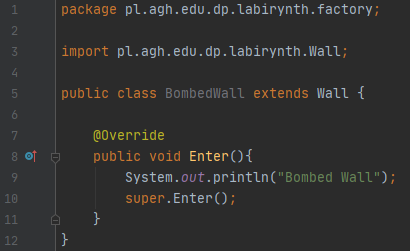
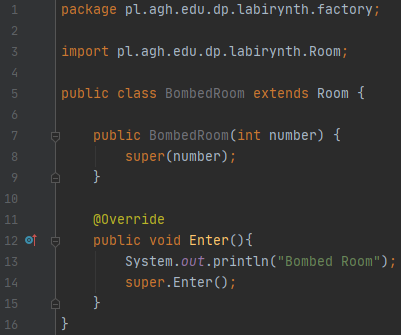
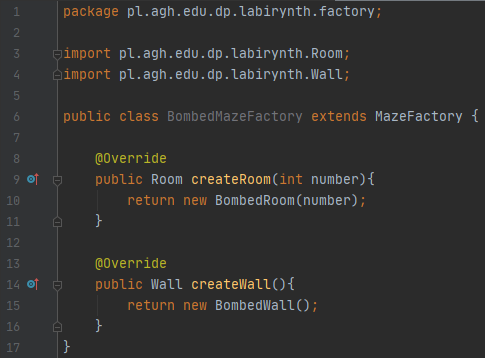
Laboratorium 3

Wzorce projektowe

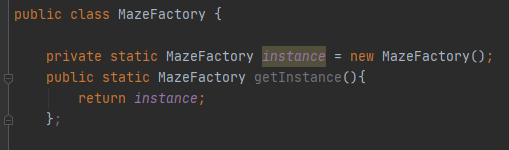
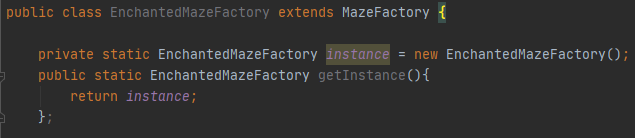
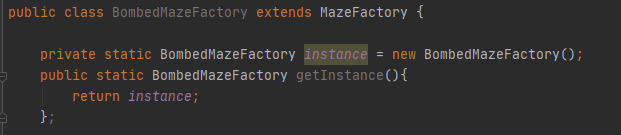
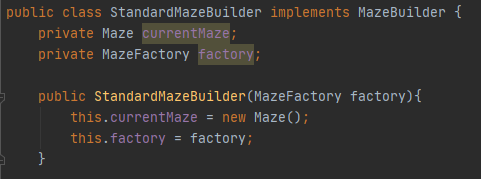
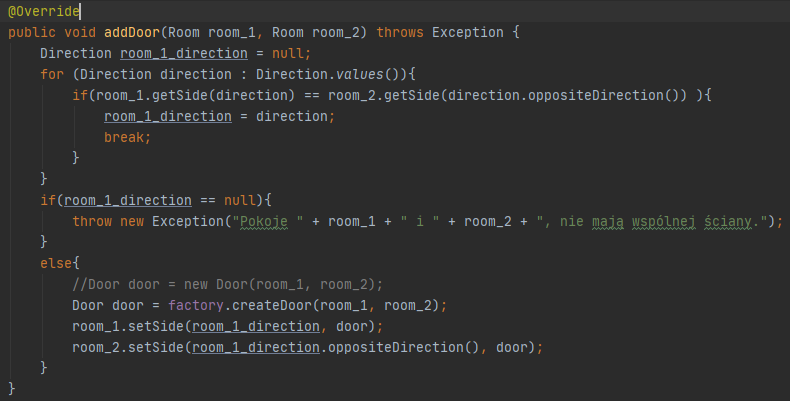
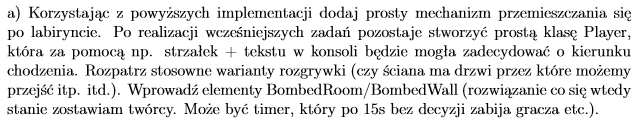
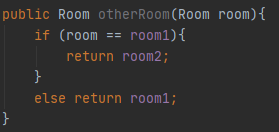
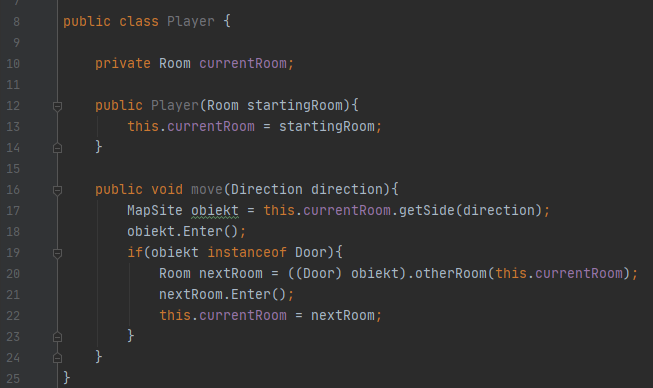
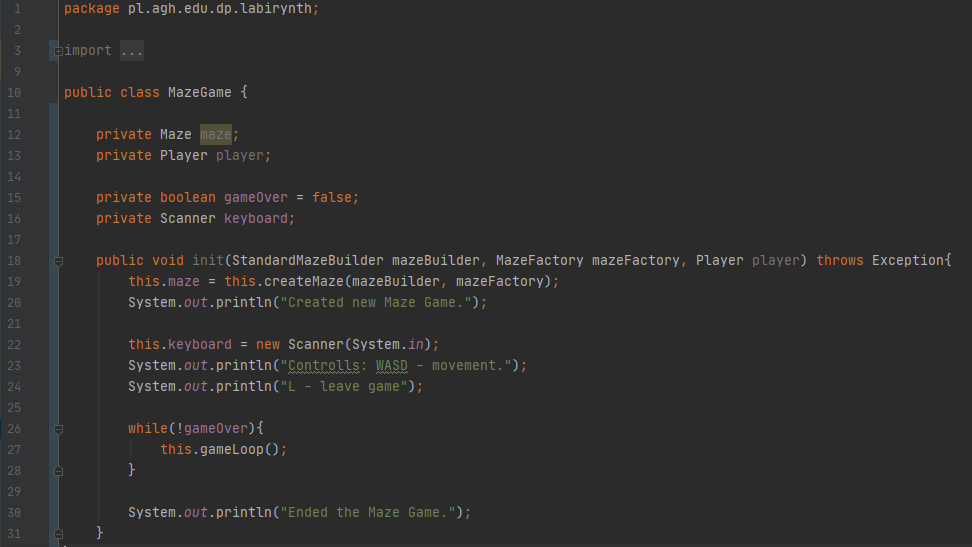
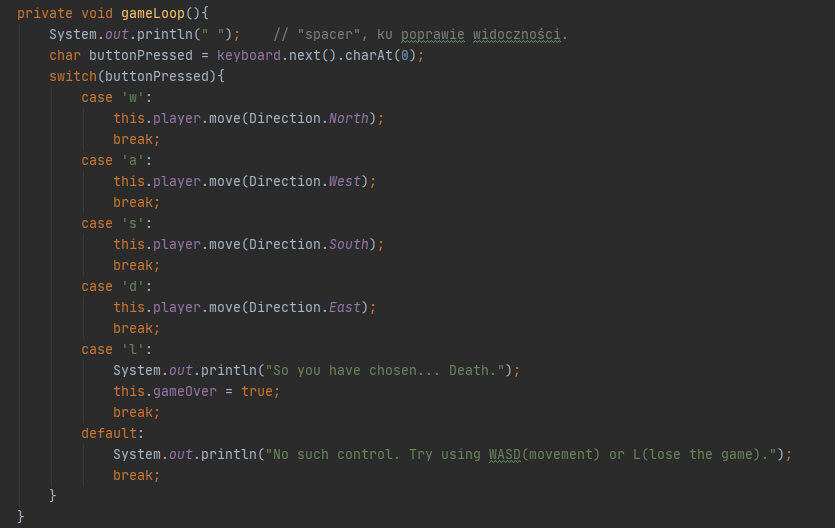
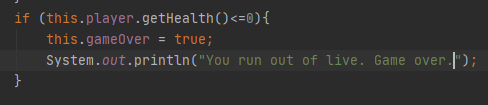
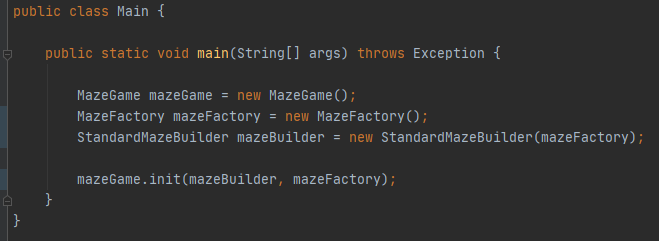
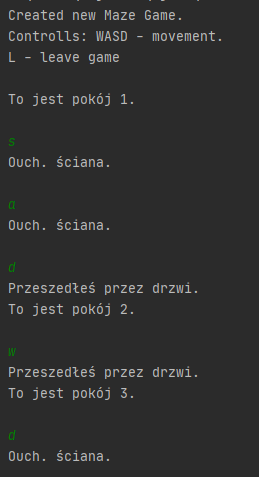
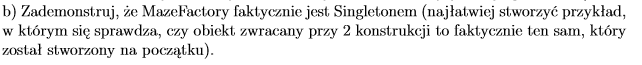
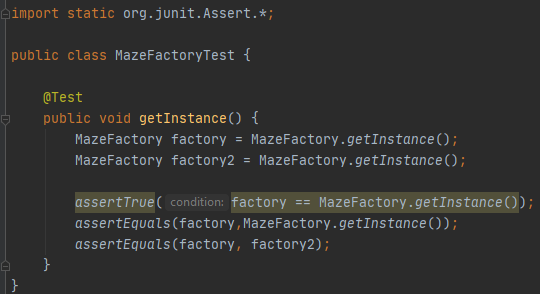


1. Stworzyłem interfejs MazeBuilder, a w niej:
   1. dodawanie pokoju
   2. dodawanie drzwi
   3. „uwspółdzielanie” (czynienie wspólną) ściany
      * 1. 
2. 
   * + 1. 
3. 
   1. Wprowadzone zmiany znacznie zwiększyły czytelność kodu.
   2. Zachowanie zasady DRY – Don’t Repeat Yourself
   3. Ułatwienie wprowadzania zmian poprzez „modularyzację”, tworzenie klas implementujących MazeBuilder.
4. 
   1. aby łatwiej implementować te metody, stworzyłem metodę w „Direction”, która zwraca mi przeciwny kierunek
      * 1. 
   2. Następnie wykorzystując metody zdefiniowane w „Room.java” stworzyłem zadane metody.
      1. Nie byłem pewny, co do „dodatkowej metody CommonWall”, która „określi kierunek standardowej ściany pomiędzy dwoma pomieszczeniami”.  
         Pomieszczenia nie są ustawione w macierzy AxB, sam labirynt może być nieskończenie długi i nieregularny, a numery pokojów nie muszą odzwierciedlać ich „sąsiedztwa”.   
         Uznałem, że metoda CommonWall „określi” = „ustawi” wspólną ścianę dla dwóch pomieszczeń.
         1. 
         2. 
         3. 
5. 
   1. Zmieniłem „Main.java”:
      * 1. 
   2. Następnie zmodyfikowałem „MazeGame.java”, aby tworzyła nieco bardziej „skomplikowany” labirynt.
      * 1. 
   3. Szkic labiryntu:
      * 1. 
6. 
   * + 1. 



1. 
   1. Stworzona klasa:
      * 1. 
2. 
   * + 1. 
3. 
   1. Rozpocząłem od stworzenia „enchanted” elementów:
      * 1. 
        2. 
        3. 
   2. Następnie stworzyłem EnchantedMazeFactory:
      * 1. 
4. 
   1. Podobnie do przykładu powyżej, stworzyłem BombedWall, BombedRoom i BombedMazeFactory:
      * 1. 
        2. 
        3. 



1. 
   1. Stworzyłem statyczne pole w MazeFactory:
      * 1. 
   2. I w klasach pochodnych MazeFactory:
      * 1. 
        2. 
2. Zmodyfikowałem StandardMazeBuilder, by korzystał z MazeFactory:
   * + 1. 
       2. 
       3. 
3. 
   1. 
      1. Rozpocząłem od problemu „przechodzenia przez pokoje”, jako że nie miałem sposobu, na zyskanie referencji do pokoju „za tą ścianą”. Dodałem więc w klasie Door metodę, która zwróci mi „ten inny pokój”:
         1. 
      2. Stworzyłem klasę Player, która umie przemieszczać się po pokojach i przechodzić przez drzwi:
         1. 
   2. Zmodyfikowałem klasę MazeGame, aby wprowadzić tam gracza. Skorzystałem z wbudowanej w Java.util klasy „Scanner”, aby zczytywać wejście konsoli.
      * 1. 
        2. 
   3. Aby dodać funkcjonalność BombedWall/BombedRoom rozszerzyłem klasę Player o atrybut Health, który będzie się zmniejszał z wartości początkowej (100) o 10 za każdym razem, gdy gracz wejdzie w taką ścianę/pokój. Gdy życie gracza spadnie do 0 - gameOver, gra przerwana.
      * 1. 
      1. I na końcu gameLoop() dodałem funkcję sprawdzającą, czy gracz aby nie umarł.  
         
   4. Zmodyfikowałem klasę main, żeby korzystała z nowych klas i metod.
      * 1. 
      1. Zauważyłem problem, jak owóż zwykłe „Room”, „Door” i „Wall”, w metodzie „Enter()” nie miały żadnego wypisywania „Przeszedłeś przez drzwi”, więc pewne dodałem.
      2. Przeniosłem również tworzenie „Player” do MazeGame, jako że musi mieć on „wstrzyknięty” pokój startowy.
      3. Dodałem do klasy „Maze” metodę „getFirstRoom()”.
      4. „Player” teraz „wchodzi” do pokoju początkowego, a nie znajduje się w nim od razu (kwestia wiadomości gdzie się znajdujemy na początku gry).
4. Grę przetestowałem i nie znalazłem w niej błędów.  
   (mapa ta sama co wcześniej (z diagramu))
5. 
   1. 
   2. 