Dokumentacja projektu

Anna Wojtkiewicz

Contents

1 Spis klas

System składa się z następujących klas:

- Bike - rowery,
- Station - stacje wypożyczalni,
- User - użytkownik,
- Rental - wypożyczenie roweru,
- Reservation - rezerwacja roweru,
- Repair - naprawa roweru,
- Simulation - główna logika projektu,
- Report - raport końcowy ze statystykami z działania symulacji.

2 Całościowy opis architektury systemu

System wypożyczalni rowerów oparty jest na podejściu obiektowym. Poszczególne klasy odwzorowują rzeczywiste obiekty w systemie: rowery, stacje, użytkowników, wypożyczenia itd.

Interakcja między obiektami zachodzi poprzez metody instancji – np. użytkownik wypożycza rower z konkretnej stacji, tworząc obiekt *Rental8, lub zgłasza jego usterkę, tworząc obiekt* Repair*.

Główne elementy systemu:

- Użytkownik (User) może wypożyczać rowery, zwracać je, rezerwować oraz zgłaszać ich usterki.
- Rower (Bike) posiada typ, status oraz informację o bieżącej lokalizacji (stacji).
- Stacja (Station) to miejsce przechowywania rowerów, ma ograniczoną pojemność i listę rowerów.
- Wypożyczenie (Rental) przechowuje informacje o rozpoczęciu i zakończeniu korzystania z roweru.
- Rezerwacja (Reservation) umożliwia czasowe zablokowanie roweru danego typu.
- Naprawa (Repair) pozwala zarządzać cyklem zgłaszania i naprawiania rowerów.
- Symulacja (Simulation) zarządza całym systemem i interakcją z użytkownikiem.
- Raport (Report) podsumowuje dane statystyczne z działania systemu.

System działa w trybie tekstoweo interfejsu użytkownika. System uwzględnia również obsługę danych wejściowych/wyjściowych w postaci plików (np. stacje, rowery) oraz przetwarza dane dynamicznie podczas działania symulacji.

3 Opis klas

3.1 Bike (Rower)

Znaczenie: Reprezentuje pojedynczy rower w systemie.

Atrybuty:

- bike_id: int - unikalny identyfikator roweru.
- type: str - typ roweru (np. miejski, górski, elektryczny).
- status: str - status roweru (np. dostępny, wypożyczony, w naprawie).
- current_station: Station - referencja do stacji, w której rower aktualnie się znajduje.

Metody:

- **init**(bike_type): tworzy instancję reprezentującą rower z określonym typem.
- change_status(new_status): zmienia status roweru.
- assign_to_station(station): przypisuje rower do stacji.

3.2 Station (Stacja)

Znaczenie: Miejsce przechowywania rowerów, posiada pojemność i nazwę.

Atrybuty:

- station_id: int - unikalny identyfikator stacji.
- name: str - nazwa stacji.
- location: str - lokalizacja stacji.
- capacity: int - maksymalna liczba rowerów.
- bikes: List[Bike] - lista rowerów obecnie znajdujących się na stacji.

Metody:

- **init**(name, location, capacity): tworzy instancję reprezentujacą stację wraz z jej nazwą, lokacją oraz limitem rowerów.

- add_bike(bike): dodaje rower do stacji.
- remove_bike(bike): usuwa rower ze stacji.
- get_available_bikes(): zwraca listę dostępnych rowerów.
- print_available_bikes(): wypisuje dostępne rowery.

3.3 User (Użytkownik)

Znaczenie: Osoba korzystająca z systemu – wypożycza, rezerwuje rowery i zgłasza usterki.

Atrybuty:

- user_id: int - identyfikator użytkownika.
- name: str - imię użytkownika.
- active_rentals: List[Rental] - lista aktywnych wypożyczeń.
- reservations: List[Reservation] - lista rezerwacji użytkownika.

Metody:

- **init**(name): tworzy instancję reprezentującą użytkownika.
- wypozycz_bike(bike, station): wypożycza rower ze stacji.
- zwroc_bike(rental, station): zwraca rower do stacji.
- aktywne_rezerwacje(): zwraca listę aktywnych rezerwacji.
- rent_bike_from_station(station): obsługuje logikę wypożyczenia z uwzględnieniem rezerwacji.
- return_bike_to_station(station): proces zwrotu roweru.
- make_reservation(station): tworzy rezerwację na rower danego typu.
- print_reservations(): wypisuje listę rezerwacji.
- report_issue(bike, simulation): zgłasza rower do naprawy.

3.4 Rental (Wypożyczenie)

Znaczenie: Przechowuje informacje o procesie wypożyczenia roweru.

Atrybuty:

- rental_id: int - identyfikator wypożyczenia.
- user: User - użytkownik wypożyczający.
- bike: Bike - wypożyczany rower.
- start_station: Station - stacja początkowa.
- start_time:datetime - czas rozpoczęcia wypożyczenia.
- end_station: Station - stacja końcowa.
- end_time: datetime - czas zakończenia.
- active: bool - status wypożyczenia.

Metody:

- **init**(user, bike, start_station): tworzy instancję reprezentującą wypożyczenie roweru z uwzględnieniem użytkownika, który wypożycza, roweru oraz stacji, z której wypożyczany jest rower.

- end_rental(station): kończy wypożyczenie.

3.5 Reservation (Rezerwacja)

Znaczenie: Umożliwia użytkownikowi wcześniejsze zarezerwowanie roweru danego typu na określony czas.

Atrybuty:

- reservation_id: int - identyfikator rezerwacji.
- user: User - użytkownik rezerwujący.
- station: Station - stacja, dla której dokonano rezerwacji.
- bike_type: str - typ roweru.
- reserved_at: datetime - czas rezerwacji.
- expiration_time: datetime - czas wygaśnięcia rezerwacji.
- fulfilled: bool - informacja, czy rezerwacja została zrealizowana.

Metody:

- **init**(user, station, bike_type, reserved_at, expiration_time): tworzy instancję reprezentującą rezerwację roweru z określonym użytkownikiem (który rezerwuję), stacją (z której chcemy rezerwować rower), typem roweru oraz terminem aktywacji i wygaśnięcia rezerwacji.
- fulfill(): oznacza rezerwację jako zrealizowaną.
- is_active(): sprawdza, czy rezerwacja nadal obowiązuje.

3.6 Repair (Naprawa)

Znaczenie: Reprezentuje proces zgłaszania i wykonywania naprawy roweru.

Atrybuty:

- bike: Bike - rower wymagający naprawy.
- user: User - użytkownik zgłaszający usterkę.
- start_time:datetime - moment zgłoszenia.
- duration: timedelta - czas potrzebny na naprawę.
- finished: bool - informacja, czy naprawa została zakończona.
- completed_at: datetime - czas zakończenia naprawy.

Metody:

- **init**(bike, user): tworzy instację reprezentującą proces naprawy roweru.
- is_done(): sprawdza, czy naprawa się zakończyła.
- complete(station): kończy naprawę i przywraca rower do dostępności.
- handle_issue_report(user, bikes, repairs_list): umożliwia użytkownikowi wybór roweru do zgłoszenia (proces zgłaszania usterki.).

3.7 Simulation (Symulacja)

Znaczenie: Reprezentuje główną pętlę działania systemu. Zarządza logiką działania aplikacji, w tym użytkownikami, stacjami, wypożyczeniami, rezerwacjami i naprawami.

Atrybuty:

- stations: List[Station] - lista stacji.
- bikes: List[Bike] - lista rowerów.
- users: List[User] - lista użytkowników.
- rentals: List[Rental] - lista wypożyczeń.
- reservations: List[Reservation] - lista rezerwacji.
- repairs: List[Repair] - lista napraw.

Metody:

- **init**(): ładuje dane z plików JSON, potrzebne do uruchomienia procesu.
- run(): uruchamia główną pętlę systemu.
- run_user_session(user): obsługuje sesję konkretnego użytkownika.
- choose_station(): umożliwia wybór stacji.
- process_repairs(): przetwarza naprawy, kończąc je w odpowiednim czasie.

3.8 Report (Raport)

Znaczenie: Generuje podsumowanie końcowe działania systemu na podstawie danych zgromadzonych podczas symulacji.

Atrybuty:

- users: List[User] - lista użytkowników.
- rentals: List[Rental] - lista wypożyczeń.
- stations: List[Station] - lista stacji.

Metody:

- **init**(users, rentals, stations): tworzy instancję reprezentująca raport końcowych statystyk.
- generate(): generuje statystyki i raport końcowy, np. najczęściej wypożyczane rowery i stacje.

4 Testy jednostkowe

W celu weryfikacji poprawności działania systemu zastosowano bibliotekę `unittest`. Testy zostały zaimplementowane w pliku `testy.ipynb`, a ich zakres obejmuje kluczowe funkcjonalności systemu zarządzania wypożyczalnią rowerów.

4.0.1 Zakres testów:

1. **Rezerwacje:**
   - Sprawdzenie, czy po dokonaniu rezerwacji jest ona oznaczona jako aktywna.
   - Weryfikacja, czy po upływie określonego czasu rezerwacja staje się nieaktywna.

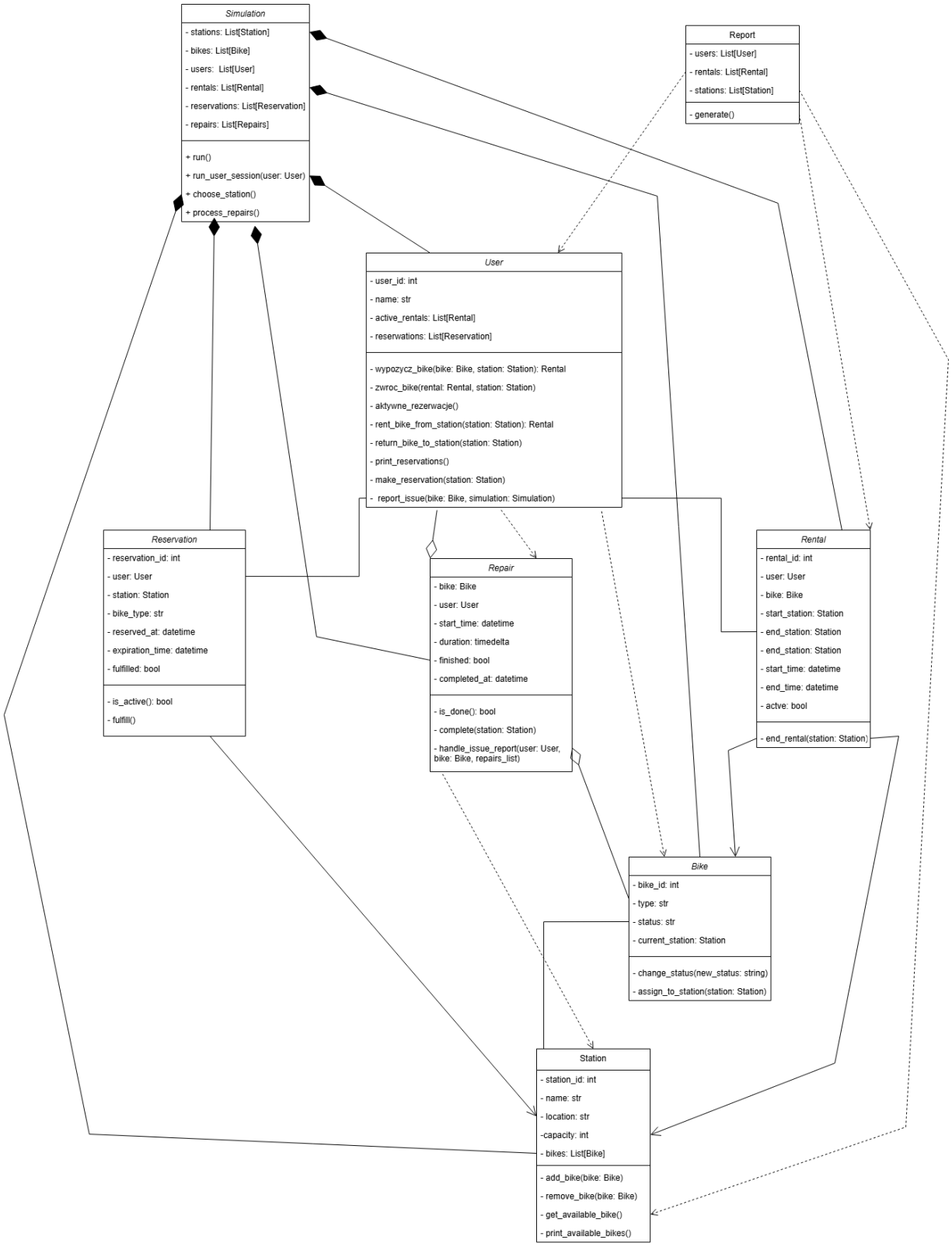
2. **Rower:**
   - Sprawdzenie, czy rower ma poprawnie przypisany typ (np. *górski*).
   - Weryfikacja, czy domyślny status nowego roweru to *dostępny*.
   - Sprawdzenie, czy rower został poprawnie dodany do stacji.
   - Sprawdzenie, czy rower został poprawnie usunięty ze stacji.
3. **Wypożyczenia:**
   - Weryfikacja, czy użytkownik może poprawnie wypożyczyć rower.
   - Sprawdzenie, czy wypożyczenie jest dodawane do listy `active_rentals`.
   - Upewnienie się, że po zwrocie wypożyczenie znika z listy `active_rentals`.
   - Sprawdzenie, czy po zwrocie rower zmienia status na *dostępny*.
   - Weryfikacja, czy przy wypożyczeniu rower otrzymuje status *wypożyczony*.
4. **Naprawy:**
   - Sprawdzenie, czy po zgłoszeniu naprawy status roweru zmienia się na *w naprawie*.
   - Weryfikacja, czy po upływie ustalonego czasu naprawa zostaje automatycznie uznana za zakończoną.
   - Po zakończeniu naprawy sprawdzane są:
     - status roweru (*dostępny*),
     - status naprawy (*finished*),
     - przypisanie roweru do odpowiedniej stacji.
5. **Stacje:**
   - Weryfikacja, czy system uniemożliwia dodanie do stacji liczby rowerów przekraczającej jej limit pojemności.

5 Diagram UML

W celu zobrazowania struktury systemu, w pliku **`diagram.png`** stworzono diagram UML przedstawiający relacje pomiędzy klasami projektu. Diagram ten uwzględnia klasy systemu, ich atrybuty oraz metody, a także zależności pomiędzy obiektami.

5.1 Diagram UML (grafika)

Poniżej znajduje się graficzna reprezentacja klas i ich relacji:


Simulation
- stations: List[Station]
- bikes: List[Bike]
- users: List[User]
- rentals: List[Rental]
- reservations: List[Reservation]
- repairs: List[Repairs]
+ run()
+ run_user_session(user: User)
+ choose_station()
+ process_repairs()
Report
- users: List[User]
- rentals: List[Rental]
- stations: List[Station]
- generate()
User
- user_id: int
- name: str
- active_rentals: List[Rental]
- reservations: List[Reservation]
- wypozycz_bike(bike: Bike, station: Station): Rental
- zwroc_bike(rental: Rental, station: Station)
- aktywne_rezerwacje()
- rent_bike_from_station(station: Station): Rental
- return_bike_to_station(station: Station)
- print_reservations()
- make_reservation(station: Station)
- report_issue(bike: Bike, simulation: Simulation)
Reservation
- reservation_id: int
- user: User
- station: Station
- bike_type: str
- reserved_at: datetime
- expiration_time: datetime
- fulfilled: bool
- is_active(): bool
- fulfill()
Repair
- bike: Bike
- user: User
- start_time: datetime
- duration: timedelta
- finished: bool
- completed_at: datetime
- is_done(): bool
- complete(station: Station)
- handle_issue_report(user: User, bike: Bike, repairs_list)
Rental
- rental_id: int
- user: User
- bike: Bike
- start_station: Station
- end_station: Station
- end_station: Station
- start_time: datetime
- end_time: datetime
- actve: bool
- end_rental(station: Station)
Bike
- bike_id: int
- type: str
- status: str
- current_station: Station
- change_status(new_status: string)
- assign_to_station(station: Station)
Station
- station_id: int
- name: str
- location: str
-capacity: int
- bikes: List[Bike]
- add_bike(bike: Bike)
- remove_bike(bike: Bike)
- get_available_bike()
- print_available_bikes()