

Sprawozdanie

- **standardy kodowania**

Standard kodowania, który przyjęliśmy polegał na:

- używaniu nazw metod zaczynających się małą literą, a w przypadku nazw dwuczłonowych – oddzielanie ich znakiem podkreślenia, np. *super_metoda()*
- nazwy klas zaczynamy wielką literą, np. *Klasa*
- nazwy pól, obiektów i zmiennych – małą, np. *jakas_zmienna*
- nazwy tablic w liczbie mnogiej, np. *wypozyczenia*
- nazwy testów konstruktorów w formie *UnitTest_JakasNazwa*
- nazwy pozostałych testów są w formie *TestCoTestujemy*
- ogólne zasady to używanie możliwie krótkich, ale też wymownych nazw i niestosowanie polskich znaków.

- **standardy dokumentowania**

Dokumentacja była pisana w formie komentarzy do kodu na bieżąco, a także uzupełniana w ostatnim etapie prac nad programem.

Przyjęliśmy tutaj następującą zasadę: kiedy komentujemy pojedynczą linijkę, to komentarze umieszczamy po prawej stronie, a jeśli cały segment to komentarz piszemy nad wybranym segmentem, oprócz tego używamy tylko komentarzy do pojedynczych linii, czyli „//”.

Przykład dokumentacji

```
Wypozyczenie(Data d1=0, Data d2=0, string nr_rej="0", string pes="0", int cena=0); //konstruktor domyslny

void zaplac(Wypozyczenie* tab_w); //ustalanie sposobu platnosci i zaplacenie lub ewentualne odroczenie

bool skroc_okres(Data nowe_zakonczenie); //metoda do skracania okresu wypozyczenia przyjmujaca nowe zakonczenie

bool wydłuż_okres(Data nowe_zakonczenie); //metoda do wydłużania okresu wypozyczenia przyjmujaca nowe zakonczenie

static Wypozyczenie* wczytaj_z_pliku(int& w); //wczytywanie wypozychen z pliku do tablicy

void aktualizuj_plik(Wypozyczenie* wypozyczenia); //aktualizacja pliku z wypozyzeniami, także z tymi, które już minely

//getterzy do pol klasy
string get_numer_rejestracyjny();
string get_pesel();
bool get_zakonczone();
Data get_data_od();
Data get_data_do();
Data get_termin_platnosci();
Platnosc * get_rachunek1(); //zwraca wskaznik, zeby pracowac na oryginalne
Platnosc get_rachunek();
```

- **język implementacji i IDE**

- Język implementacji: C++

- Środowisko: Visual Studio 2019

- Biblioteki z jakich skorzystaliśmy:

<iostream> - wypisywanie tekstu na ekran i wczytywanie z klawiatury

<windows.h> - zamrażanie ekranu konsoli w celu wyświetlania czytelniejszych

komunikatów

<time.h> - pobieranie aktualnego czasu z komputera

<fstream> - obsługa plików tekstowych do przechowywania danych

<sstream> - do pobierania linii z pliku

- **wykorzystanie repozytorium**

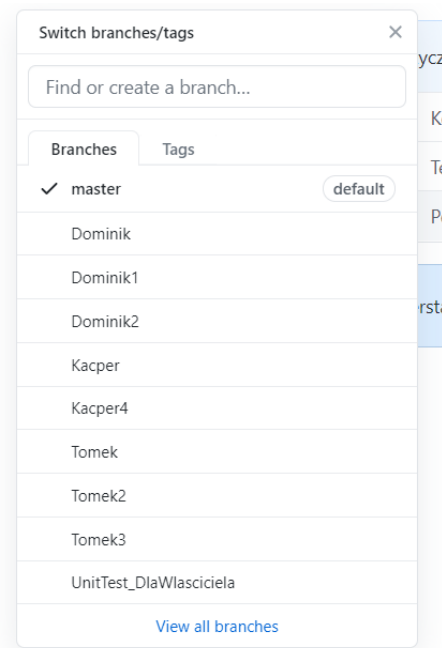
- Adres repozytorium: <https://github.com/Bielak2000/Wypozyczalnia-Samochodow>

- Sposób tworzenia:

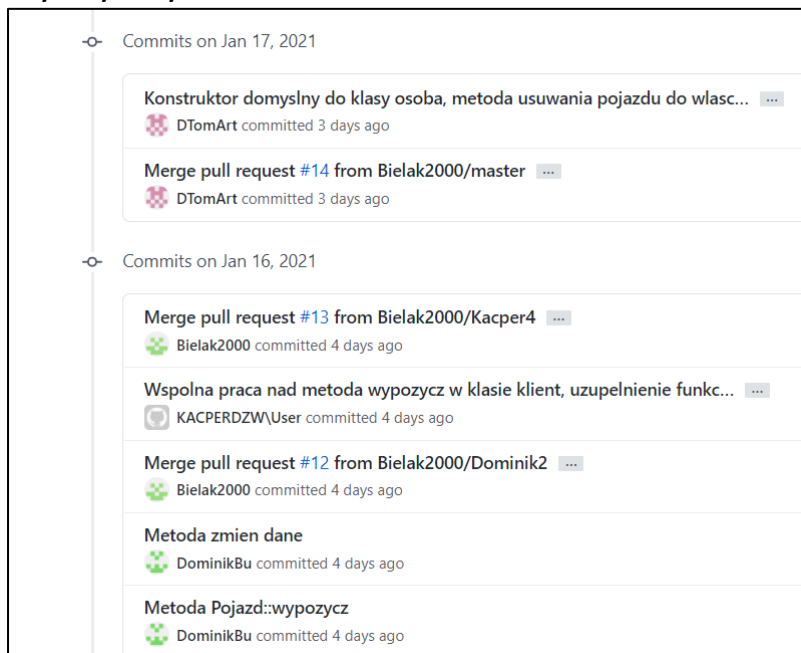
Każda osoba z naszego zespołu miała korzystać ze swojej gałęzi. W praktyce jednak, ze względu na to, iż był to nasz pierwszy projekt mieliśmy czasami problemy z obsługą GitHuba, głównie ze scalaniem, więc każdy miał w końcowej fazie projektu po kilka gałęzi.

Rozdzielaliśmy między siebie określone zadania, najczęściej w postaci napisania konkretnych metod, które następnie commitowaliśmy i pushowaliśmy na swoje gałęzi.

Wyznaczaliśmy przy tym terminy spotkań, na których scalaliśmy nasze rozwiązania do gałęzi głównej, wspólnie zastanawialiśmy się nad dalszym kształtem projektu i wymienialiśmy spostrzeżenia.



Przykłady naszych commitów:

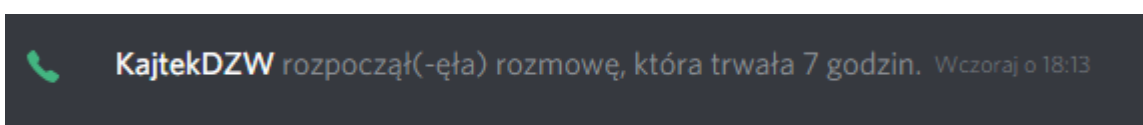


	Commits on Jan 6, 2021
	Metoda dodaj_pojazd DominikBu committed 14 days ago
	Dodanie konstruktora właściciela DominikBu committed 14 days ago
	UnitTest_DlaWlasciciela DominikBu committed 14 days ago
	Drobne poprawki wykonanene przez nas wszystkich w porozumieniu. KACPERDZW\User committed 14 days ago
	Merge pull request #1 from DTomArt/master ... Bielak2000 committed 14 days ago
	Stworzono konstruktor dla klienta, Uzupełniono funkcje wypisując dan... ... DTomArt committed 15 days ago

Przykłady naszych Pull Request'ów:

<input type="checkbox"/>	0 Open ✓ 34 Closed
<input type="checkbox"/>	Tomek2 #34 by Bielak2000 was merged 16 minutes ago
<input type="checkbox"/>	Refaktoryzacja, zmiana nazw pól i dodanie komentarzy #33 by Bielak2000 was merged 25 minutes ago
<input type="checkbox"/>	Refaktoryzacja kodu, uzupełnianie dokumentacji oraz inne zmiany. #32 by Bielak2000 was merged 27 minutes ago
<input type="checkbox"/>	Aktualizacja do ostatecznej wersji #31 by DTomArt was merged 2 hours ago
<input type="checkbox"/>	Poprawa maina #30 by Bielak2000 was merged 2 hours ago
<input type="checkbox"/>	Poprawa metody usun_pojazd w klasie Wlasciciel. #29 by Bielak2000 was merged 3 hours ago
<input type="checkbox"/>	Kacper4 #28 by Bielak2000 was merged 4 hours ago
<input type="checkbox"/>	male zmiany #27 by DTomArt was merged 5 hours ago
<input type="checkbox"/>	Tomek2 #26 by DTomArt was merged 6 hours ago

Podczas omawiania projektu niejednokrotnie robiliśmy dłuższe sesje wspólnego programowania, co znacznie pomagało w pisaniu bardziej skomplikowanych funkcji, które na bieżąco trzeba było obmyślać.

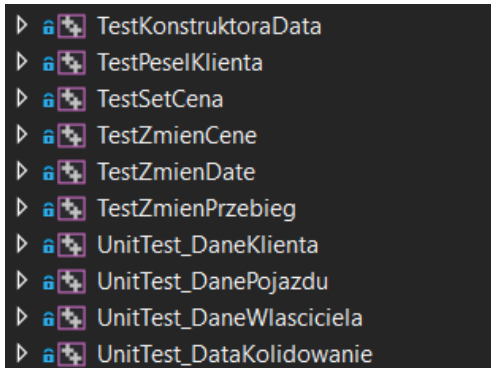


- **informacje o zaimplementowanych testach jednostkowych**

Do tworzenia testów jednostkowych używaliśmy „*natywnych projektów testów jednostkowych*” dostępnych w Visual Studio 2019.

Efekt naszych testów są testy różnych metod używanych w programie. W przypadku testów konstruktorów, chcieliśmy sprawdzać głównie kolejność danych podanych do konstruktora, tak aby mieć pewność że w trakcie refaktoryzacji nie zamienimy kolejności danych w obiekcie.

Testy widoczne z okna Visual Studio

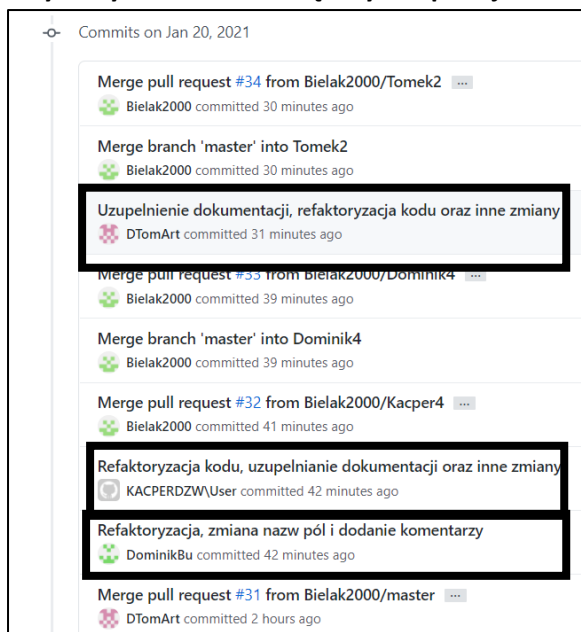


- **TDD**

W swojej pracy staraliśmy się korzystać z podejścia TDD. Przy wszystkich testach dzieliliśmy implementację na 3 fazy:


- **Red** – dodanie niedziałającego testu oraz samej deklaracji bez ciała funkcji,
- **Green** – dodanie zawartości do ciała funkcji na podstawie testu, który na tym etapie już przechodził,
- **Refactor** – faza refaktoryzacji, czyli poprawienia kodu, którą w głównym stopniu zostawiliśmy na sam koniec implementacji, tak aby wyrobić się z całością projektu.


Przykłady commitów związanych z podejściem TDD:




Commits on Jan 18, 2021


Merge pull request #18 from Bielak2000/Kacper4 ...


 Bielak2000 committed 2 days ago


Faza GREEN - zaimplementowanie funkcji set_cena(), która pozytywnie ...
 KACPERDZW\User committed 2 days ago


Faza Red - utworzenie testu do funkcji set_kwota() w klasie Platnosc.
 KACPERDZW\User committed 2 days ago

Merge pull request #16 from Bielak2000/Tomek2 ...

 DTomArt committed 2 days ago


no message
 KACPERDZW\User committed 2 days ago


Faza Green - metoda klasy Data "czy_koliduje" sprawdzajaca czy daty k...
 DTomArt committed 2 days ago


Faza Red - dodano test do metody statycznej w klasie Data oraz sama j...
 DTomArt committed 2 days ago


Commits on Jan 14, 2021

Revert "no message" ...


 KACPERDZW\User committed 6 days ago


no message
 KACPERDZW\User committed 6 days ago

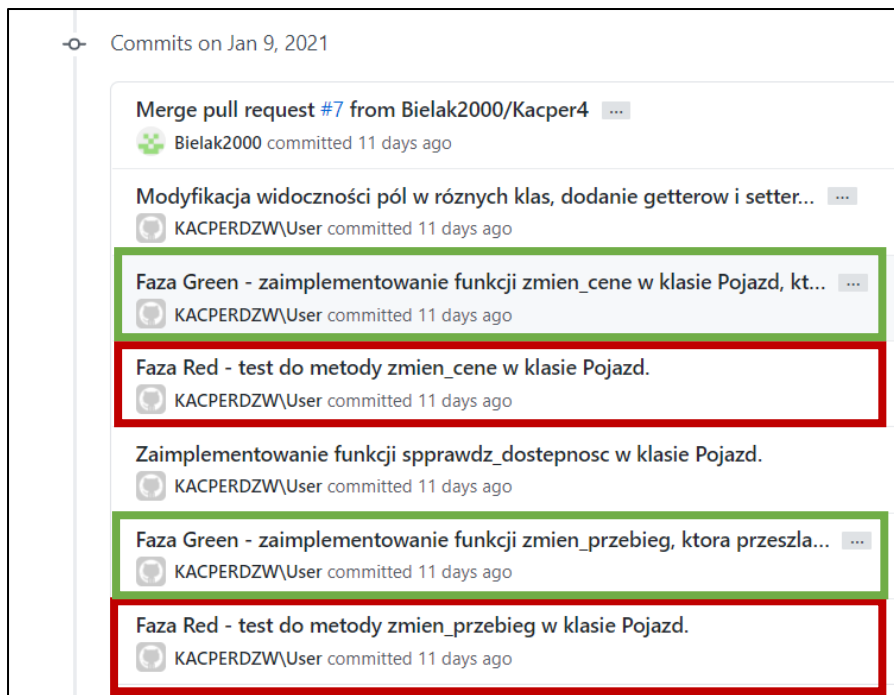
Uzupełnienie głównego menu w mainie, refaktoryzacja funkcji aktualizac...
 KACPERDZW\User committed 6 days ago

Poprawa testu konstruktora Właściciela oraz refaktoryzacja konstruktorów
 DominikBu committed 6 days ago

Commits on Jan 13, 2021

Faza Green - zaimplementowano konstruktor klasy pojazd (według testu)
 DTomArt committed 7 days ago

Faza Red - dodano test do konstruktora pojazdu; dodano gettery do pól...
 DTomArt committed 7 days ago



- **odniesienie do metodyk wytwarzania** - czy cokolwiek jest podobnie, jak w jakiejś popularnej metodyce?

🌀 Nasza praca nad projektem w największym stopniu przypominała programowanie zwinne. W trakcie pracy staraliśmy się przez cały czas być elastyczni w postawionych wymaganiach i przez cały czas je dostosowaliśmy w miarę napotkania przeszkód. Nie trzymaliśmy się kurczowo opracowanego wcześniej planu, tylko na bieżąco go dostosowywaliśmy poznając realia praktycznej pracy nad projektem, co oszczędziło nam wielu problemów technicznych w implementacji.

🌀 W naszej pracy można też znaleźć nawiązanie do programowania ekstremalnego, ponieważ stosowaliśmy kilka z jego złotych reguł, a były to:

> Programowanie w parach – z uwagi na nasze wspólne sesje programowania, podczas których pracowaliśmy nad implementacją bardziej skomplikowanych funkcjonalności; w naszym przypadku było to programowanie także w trójkę, dzięki czemu byliśmy w stanie bardzo szybko debuggować kod oraz organizować dłuższe sesje programowania (gdy jedna z 3 osób potrzebowała chwili rozluźnienia, włączała się inna, dzięki czemu praca mogła trwać dłużej)

> Przyjęcie standardu kodowania – Z uwagi na przyjęty standard kodowania opisany na 1 stronie sprawozdania

> Wspólna odpowiedzialność – często w przypadku błędów wspólnie, ale także osobno poprawialiśmy kod po sobie nawzajem

> Ciągłe łączenie – ponieważ po dodawaniu prawie każdej funkcjonalności scalaliśmy nasze gałęzie z gałęzią główną „masterem”, aby każdy miał zaktualizowaną wersję programu.