**Dział I: Dodanie gitignore i praca na gałęziach**

1. **Dodanie pliku .gitignore**

Dodano plik .gitignore do repozytorium, który zabezpiecza projekt przed nieumyślnym dodaniem niepożądanych plików i folderów generowanych automatycznie przez narzędzia i IDE (np. bin, target, .idea). Wprowadzono zmiany do repozytorium lokalnego za pomocą poleceń git add i git commit z odpowiednim opisem.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. **Poprawa formatowania kodu**

Poprawiono błędy związane z formatowaniem kodu źródłowego Java w klasie HanoiRhymer. Wprowadzono zmiany do repozytorium lokalnego, a następnie do repozytorium zdalnego za pomocą polecenia git push.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. **Praca z gałęzią format**

Na platformie GitHub przełączono się na gałąź format i wyświetlono listę commitów.

**Ostatni commit w gałęzi format nazywał się [ostatnie szlify formatowania (puste linie)]. W tym commicie usunięto następujące linie: 10 i 16.**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający zrzut ekranu, tekst, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Porównano plik HanoiRhymer.java w gałęzi master i format za pomocą polecenia git diff origin/format \*Hanoi\*. Nie było żadnych różnic w pliku HanoiRhymer.java.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Przełączono się na gałąź master i scalono zmiany z gałęzi format do gałęzi master przy użyciu poleceń git merge --squash i git commit. Sprawdzono, czy wszystkie błędy formatowania zostały usunięte we wszystkich klasach, uruchamiając automatyczne formatowanie.

Kombinacja klawiszy alt + ← oraz alt + → umożliwiają nawigację między ostatnio edytowanymi miejscami w kodzie - lewo cofa do poprzedniego, prawo przechodzi do następnego miejsca edycji.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. **Konwencja nazewnicza klas i metod**

Utworzono gałąź class-method-naming i przełączono się na nią. Na tej gałęzi poprawiono błędy konwencji nazewniczej klas i metod kodu źródłowego Java we wszystkich klasach. Wykonano osobne commity dla zmian nazw klas i osobne dla zmian nazw metod. Do refaktoryzacji użyto opcji Refactor → Rename.

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

1. **Konwencja nazewnicza pól klas**

Przełączono się na gałąź master i utworzono gałąź fields-naming. Na tej gałęzi poprawiono błędy konwencji nazewniczej pól klas we wszystkich klasach i wykonano commit.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Przełączono się na gałąź master i scalono zmiany z gałęzi fields-naming do gałęzi master przy użyciu polecenia git merge --ff-only.

Obraz zawierający tekst, zrzut ekranu, Czcionka, oprogramowanie

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Przełączono się na gałąź class-method-naming i użyto polecenia git rebase, aby przenieść ją na szczyt gałęzi master. Następnie przełączono się na gałąź master i scalono zmiany z gałęzi class-method-naming do gałęzi master przy użyciu polecenia git merge --no-ff.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

**Odpowiedź na pytanie dotyczące różnicy działania opcji polecenia merge:**

**--squash** - łączy wszystkie zmiany z gałęzi źródłowej w jeden commit na gałęzi docelowej, bez zachowania historii commitów

**--ff-only** - wykonuje merge tylko jeśli możliwe jest proste przesunięcie wskaźnika (fast-forward), zachowując liniową historię

**--no-ff** - zawsze tworzy nowy commit scalający, nawet gdy możliwy jest fast-forward, co zachowuje informację o istnieniu gałęzi

**Wyjaśnienie zachowania polecenia rebase:**

Rebase przenosi commity z jednej gałęzi na szczyt innej, przepisując historię. Tworzy to liniową historię commitów zamiast rozgałęzionej, jakby zmiany były wprowadzane sekwencyjnie na najnowszej wersji kodu bazowego. Ułatwia to czytanie historii, ale zmienia identyfikatory commitów.

***NIEKTÓRE KOMENDY NIE DZIAŁAJĄ NA PODANEJ STRONIE WIĘC ZROBIŁEM WIZUALIZACJE W ASCII* 😊**

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, Oprogramowanie multimedialne

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Obraz zawierający tekst, zrzut ekranu, multimedia, gadżet

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Obraz zawierający tekst, zrzut ekranu, multimedia, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.Obraz zawierający tekst, zrzut ekranu, multimedia, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

Wprowadzono zmiany do zdalnego repozytorium.

Obraz zawierający tekst, zrzut ekranu, Czcionka

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.

GRAF COMMITÓW

Obraz zawierający tekst, zrzut ekranu, oprogramowanie, System operacyjny

Zawartość wygenerowana przez sztuczną inteligencję może być niepoprawna.