|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| *Федеральное государственное бюджетное образовательное учреждение высшего образования* ***«МИРЭА – Российский технологический университет»***  **РТУ МИРЭА** |

Институт Информационных технологий (ИТ)

Кафедра Математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ПРАКТИЧЕСКАЯ РАБОТА №3**

по дисциплине

**«Тестирование и верификация программного обеспечения»**

Тема: **«Разработка через тестирование»**

**Студенты группы** ИКБО-06-19 Остапенко Д. Д., Остапенко Н. Д.,  
 Тугушев Д. Р., Тугушев Т. Р.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись студента)

**Принял руководитель работы** Миронов А. И.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись руководителя)

Практические работы выполнены «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

Зачтено «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2021 г.

2021

# **Содержание**

[**1. Выбор и описание методологии разработки** 3](#_Toc85441243)

[**2. Наименование ПО и его описание** 4](#_Toc85441244)

[**2.1** Наименование ПО 4](#_Toc85441245)

[**2.2** Описание игры 4](#_Toc85441246)

[**2.3** Описание ПО 5](#_Toc85441247)

[**3. Демонстрация работоспособности ПО** 6](#_Toc85441248)

[**4. Результат работы в системе контроля версий** 11](#_Toc85441249)

[**5. Заключение** 18](#_Toc85441250)

[**6. Список использованных источников** 19](#_Toc85441251)

# **1. Выбор и описание методологии разработки**

Для выполнения данной практической работы среди представленных была выбрана методология «Разработка через тестирование» (“Test Driven Development”).

Во время разработки через тестирование, в начале цикла разработки пишутся тесты, которые покрывают небольшое количество вариантов тестирования и основываются на требованиях к ПО. Далее к готовым тестам пишется код, который должен пройти подготовленный тест. После прохождения всех подготовленных тестов необходимо отрефакторить полученный код, и далее перейти к следующему циклу разработки ПО.

Эта методология позволяет добиться создания пригодного для автоматического тестирования приложения и очень хорошего покрытия кода тестами, так как ТЗ переводится на язык автоматических тестов, то есть всё, что программа должна делать, проверяется. Также TDD часто упрощает программную реализацию: так как исключается избыточность – если компонент проходит тест, то он считается готовым. Если же существующие тесты проходят, но работает компонент не так, как ожидается, то это значит, что тесты пока не отражают всех требований и это повод добавить новые тесты.

В отличие от метода разработки, при котором сначала пишется код, а потом тест к нему, в методологии TDD сначала необходимо написать тест, а потом код к нему.

# **2. Наименование ПО и его описание**

## **2.1** Наименование ПО

Программное обеспечение, разработанное в рамках данной практической работы, называется “Durak Card Game”, или «Карточная игра Дурак».

## **2.2** Описание игры

В «Дурака» играют, используя колоду из 36 карт (числа от 2 до 5, которые обычно встречаются в колоде из 52 карт, не включены). После перетасовки целой колоды каждый из игроков получает по 6 карт.

Далее из верхней части колоды тянется карта для того, чтобы определить масть козыря, затем помещается в нижнюю часть колоды так, чтобы её было видно всем игрокам, причем эту карту всё ещё можно вытянуть в процессе игры.

В представленной реализации данной игры присутствуют 2 игрока, которые по очереди атакуют и защищаются. Атакующий игрок начинает свой ход с любой из своих (коих в среднем шесть) карт лицевой стороной вверх для того, чтобы защищающийся понимал, чем ему отбивать атаку.

Защита происходит следующим образом:

* Карты козырной масти могут быть отбиты только картами той же козырной масти, но более высокого ранга.
* Карту не козырной масти можно отбить любой картой, чья масть – козырная.
* Все карты могут быть отбиты любой картой той же масти, но более высокого ранга.

После хотя бы одной успешной защиты атакующий игрок может или объявить свою атаку оконченной (отбитой), или продолжить атаковать своими картами. Единственной правило проведения атаки новыми картами состоит в том, что они должны иметь тот же ранг, что и какая-либо карта, уже находящаяся на столе начатого раунда.

По окончанию раунда, атакующий игрок вытягивает карты для того, чтобы восполнить свою колоду, до тех пор, пока карт в колоде не станет шесть, после чего по тому же принципу карты тянет защищающийся игрок.

Нападающим в новом раунде будет тот, кто одержал победу в предыдущем – либо атакующий, вынудивший защищающегося забрать карты, либо защищающийся, успешно отбивший все карты атакующего.

Игра продолжается до тех пор, пока колода не закончится, и игроки не сбросят все свои карты.

## **2.3** Описание ПО

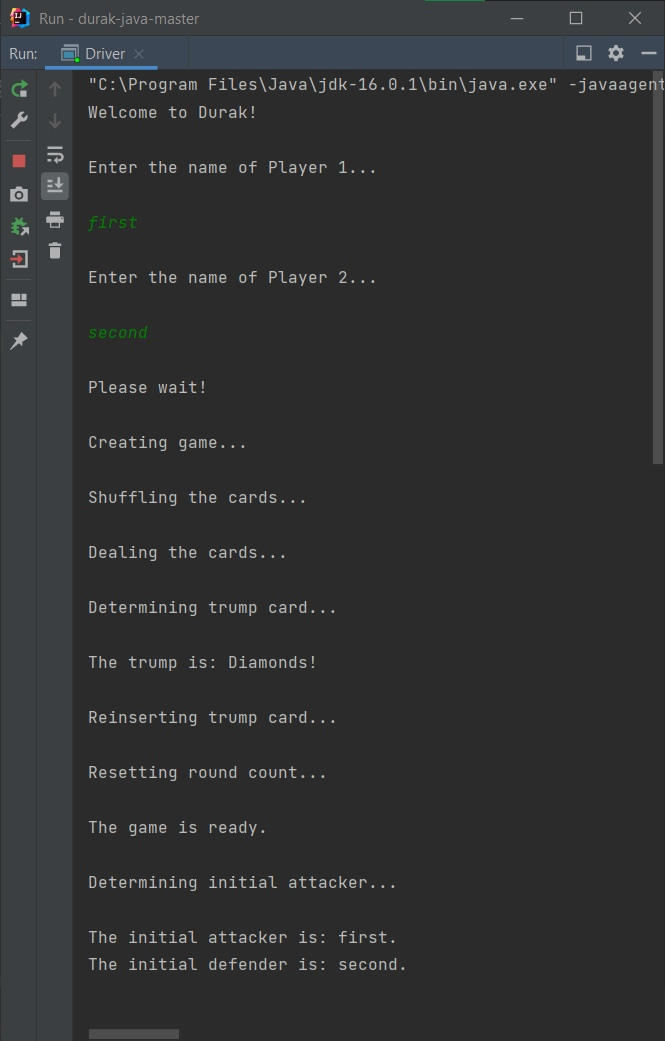
Данное программное обеспечение представляет собой версию карточной игры «Дурак» для консоли.

Классами реализованного приложения являются “Card”, “Deck”, “Hand”, “Player”, “Pair”, “Field”. Также, в пакете “cardstatic” есть класс “Static”, который управляет поведением карт.

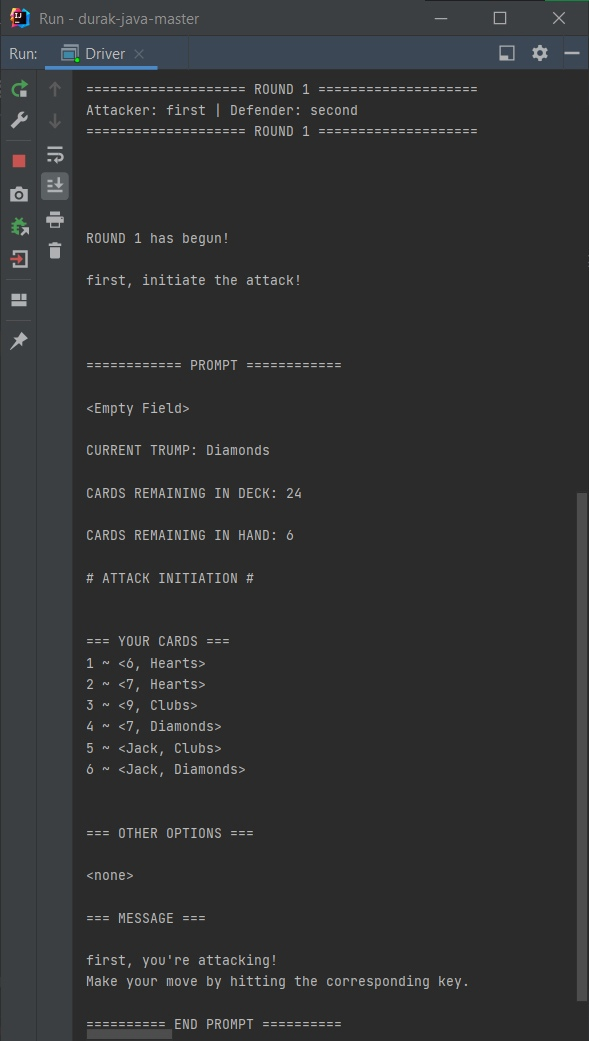
«Дурак» также содержит компоненты, регулирующие игровой процесс. Так, “Driver” запускает игру путем создания экземпляра класса “Durak”, непосредственно инициирующего запуск игры.

# **3. Демонстрация работоспособности ПО**

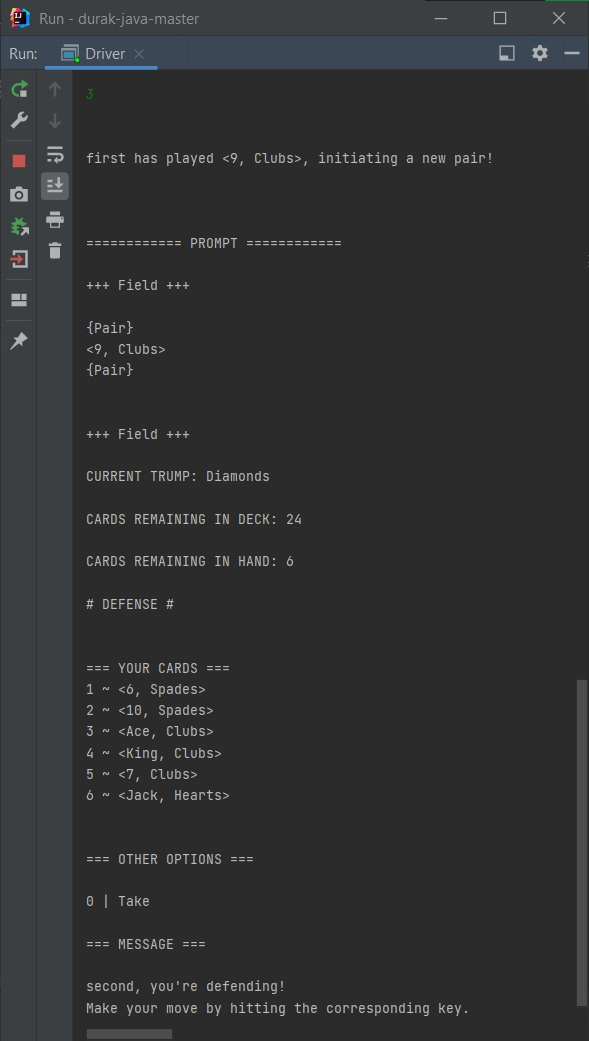
При запуске приложения необходимо ввести имя первого и второго игрока, после чего происходит начальная инициализация игры, перемешивание карт и их перетасовка, определение козырной масти и вставка вытащенной карты обратно в колоду. После один из игроков назначается атакующим, а второй – защищающимся:

  
Рисунок 1 – Ввод имени первого и второго игрока

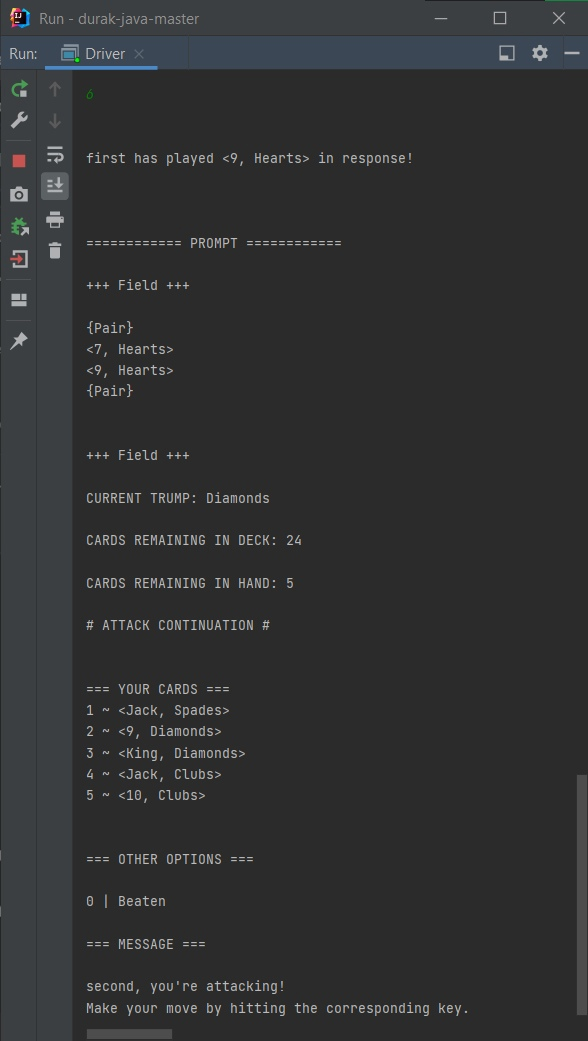
После подготовки к запуску игры, начинается первый раунд: атакующему игроку отображаются число карт в колоде и в наборе собственных карт, карты игрока с их детальным описанием, дополнительными опциями (на данном этапе отсутствуют) и сообщением о необходимости выбора карты атакующим игроком путем ввода соответствующего карте числа:

  
Рисунок 2 – Ожидание пользовательского ввода для начала атаки

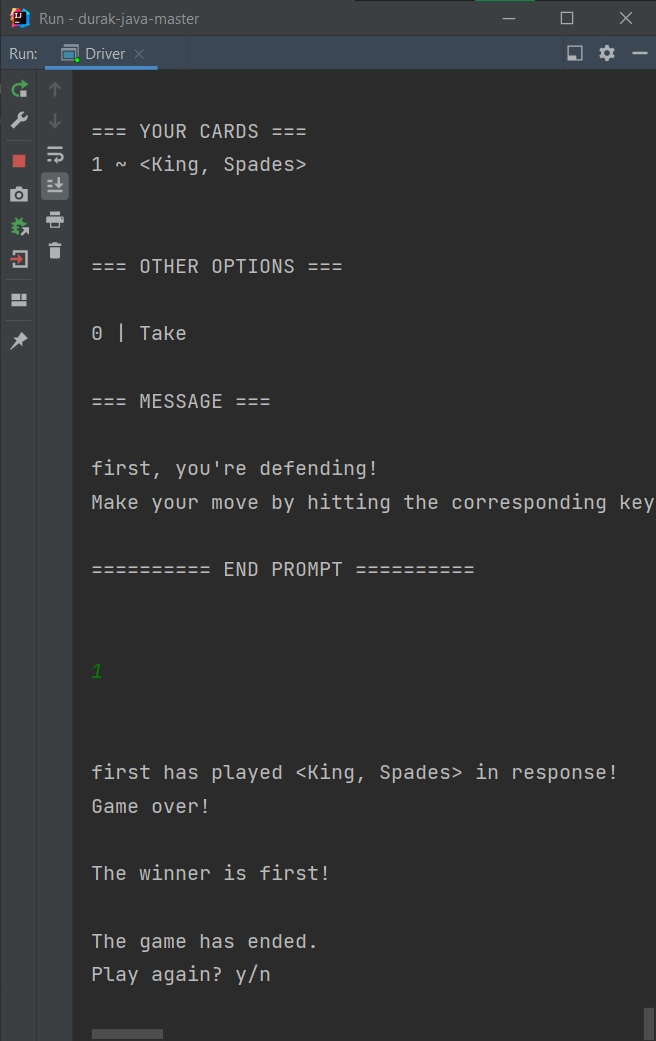
После того, как атака была совершена, ход переходит к защищающемуся игроку: ему отображаются число карт в колоде и в наборе собственных карт, карты игрока с их детальным описанием, дополнительными опциями (взять карты, выброшенные атакующим, при вводе числа 0) и сообщением о необходимости выбора карты защищающимся игроком путем ввода соответствующего карте числа:

  
Рисунок 3 – Ожидание пользовательского ввода для инициирования защиты

После ответа защищающегося игрока, ход снова переходит к атакующему, который теперь может, кроме как атаковать новой картой, еще и объявить свою атаку оконченной по вводу числа 0:

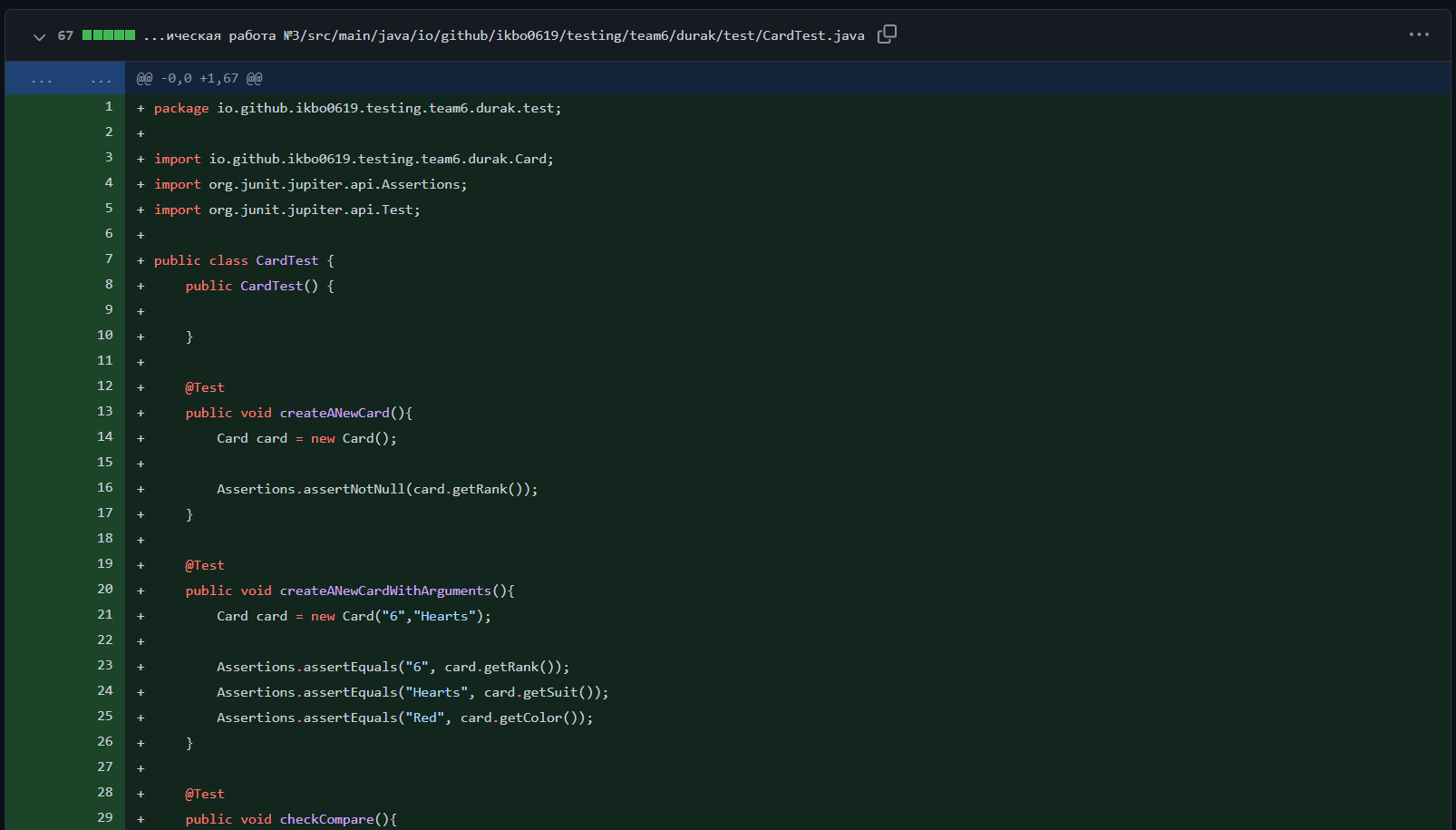
  
Рисунок 4 – Ожидание пользовательского ввода для продолжения атаки

Окончанием текущей игровой сессии является победа одного из игроков, о которой сообщает консоль. Далее она предоставляет на выбор два действия – либо начать новую игру путем ввода символа “y”, либо завершить исполнение программного кода путем ввода символа “n”:

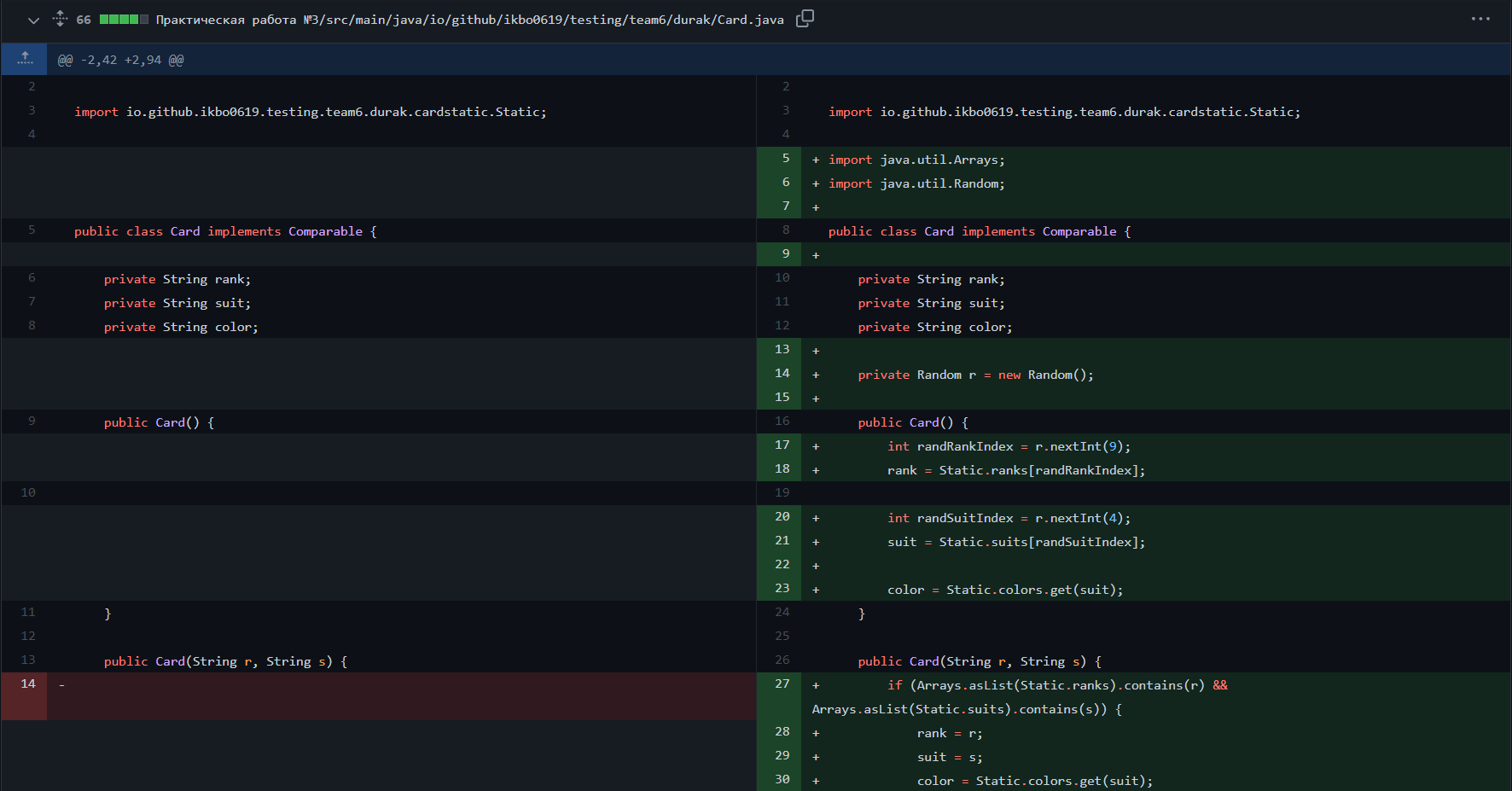
  
Рисунок 5 – Победа одного из игроков в игре «Дурак»

# **4. Результат работы в системе контроля версий**

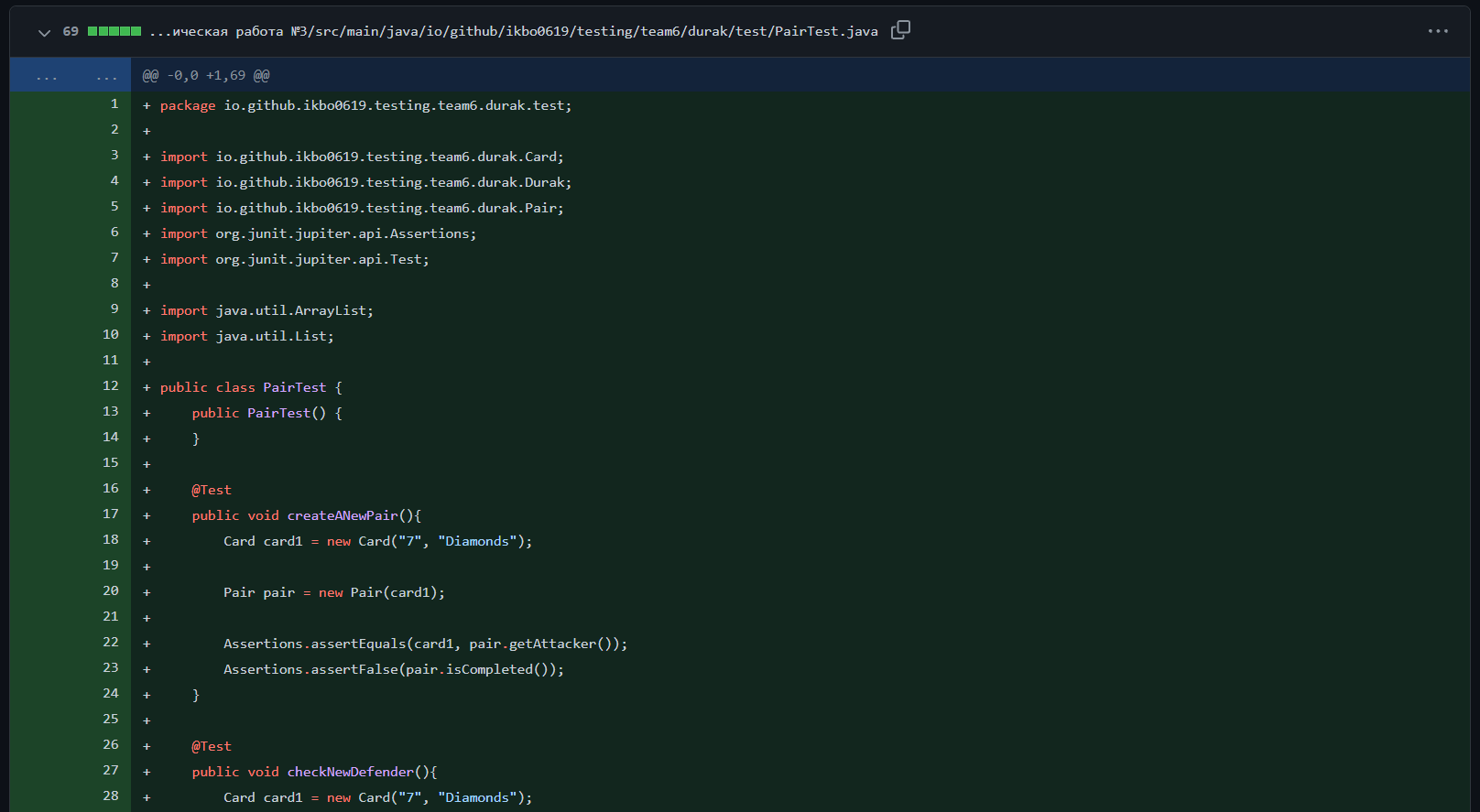
Сначала были составлены тесты для класса карты игрока Card с определенной мастью, где тестируются определенные, но не реализованные методы класса:

  
Рисунок 1 – Подготовленные тесты для класса карты игрока

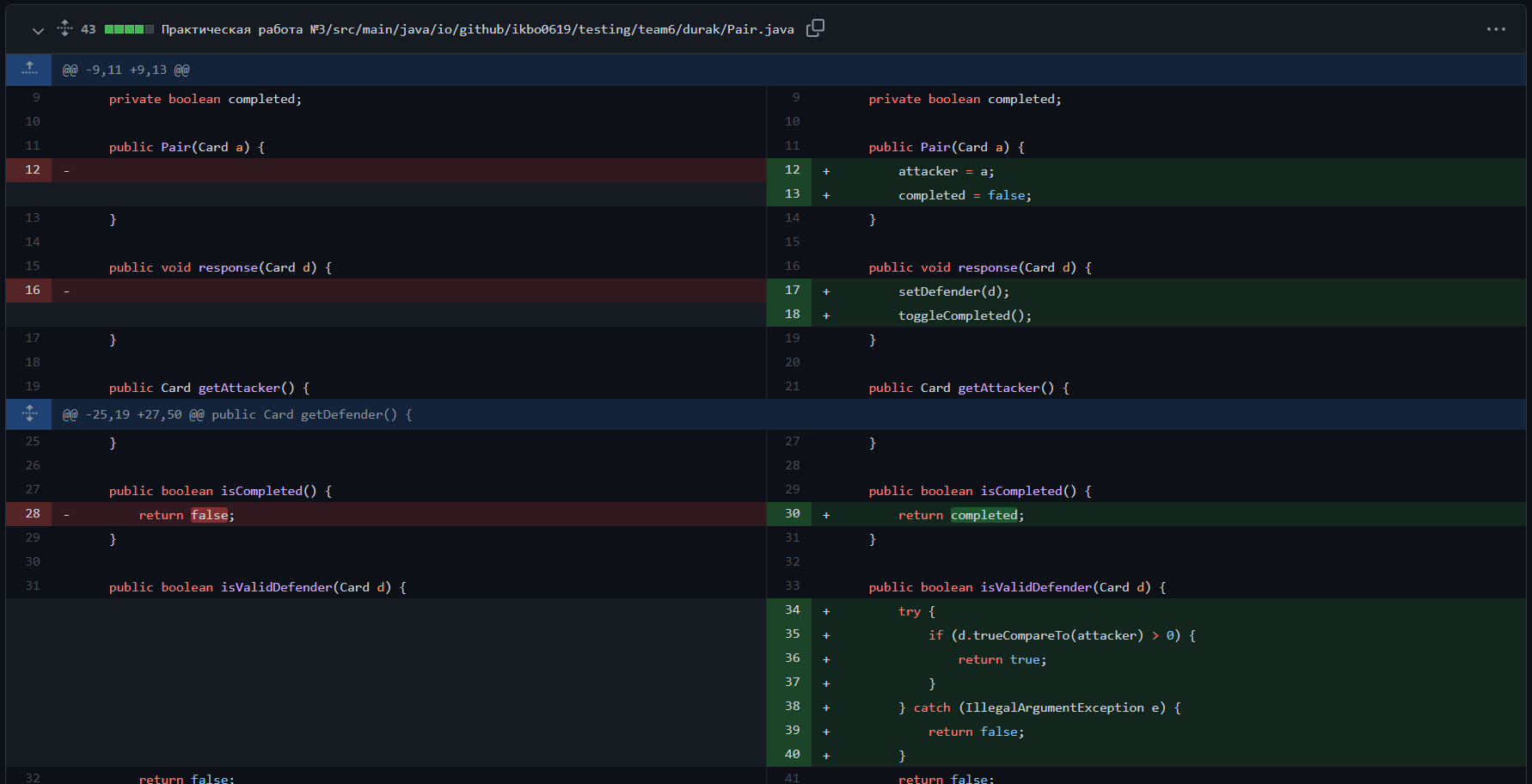
Далее, по составленным тестам был написан код для методов класса:

  
Рисунок 2 – Изменения в классе карты игрока

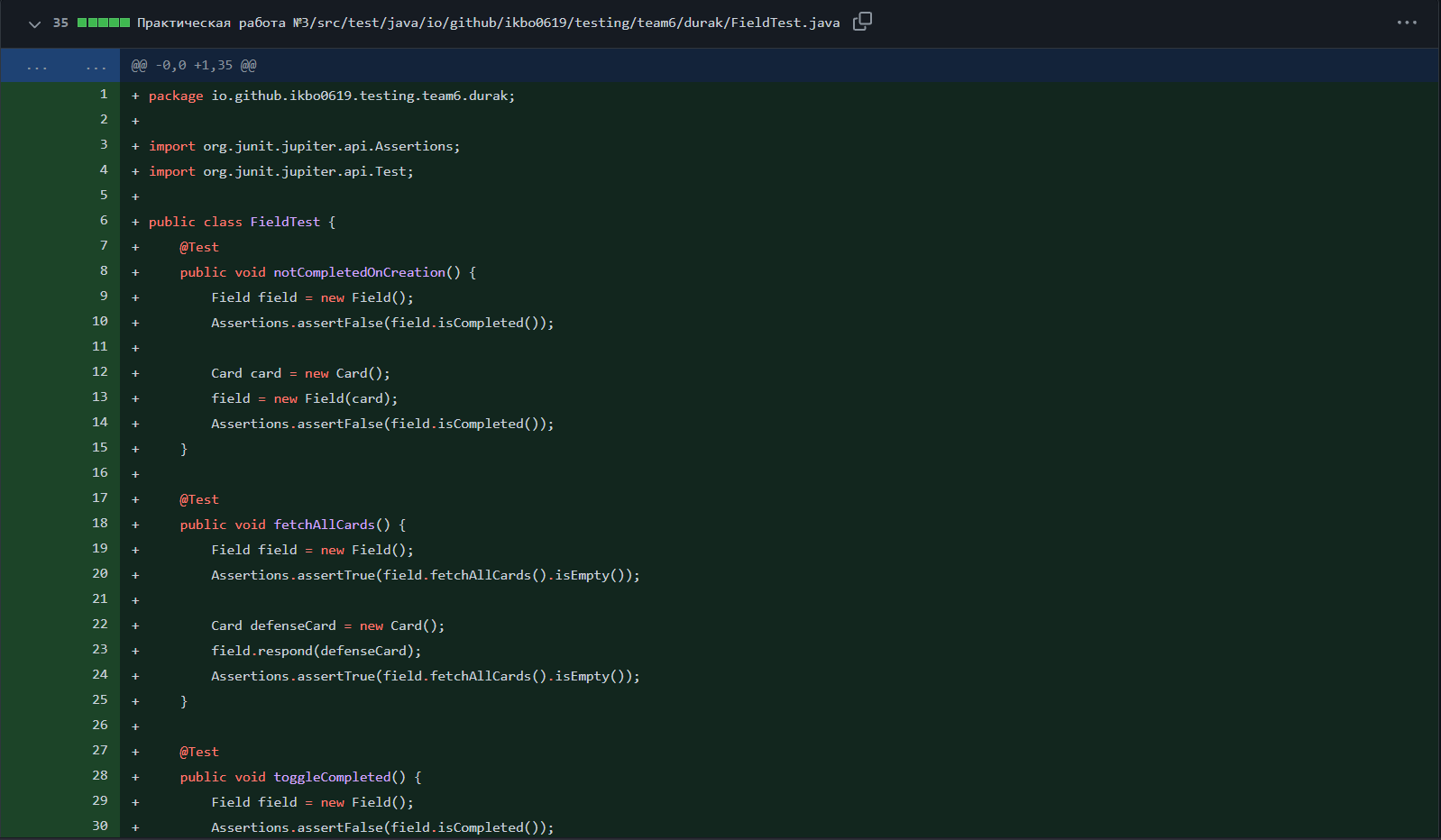
Затем, были составлены тесты для класса пары карт Pair с определенными для него методами:

  
Рисунок 3 – Подготовленные тесты для класса пары карт

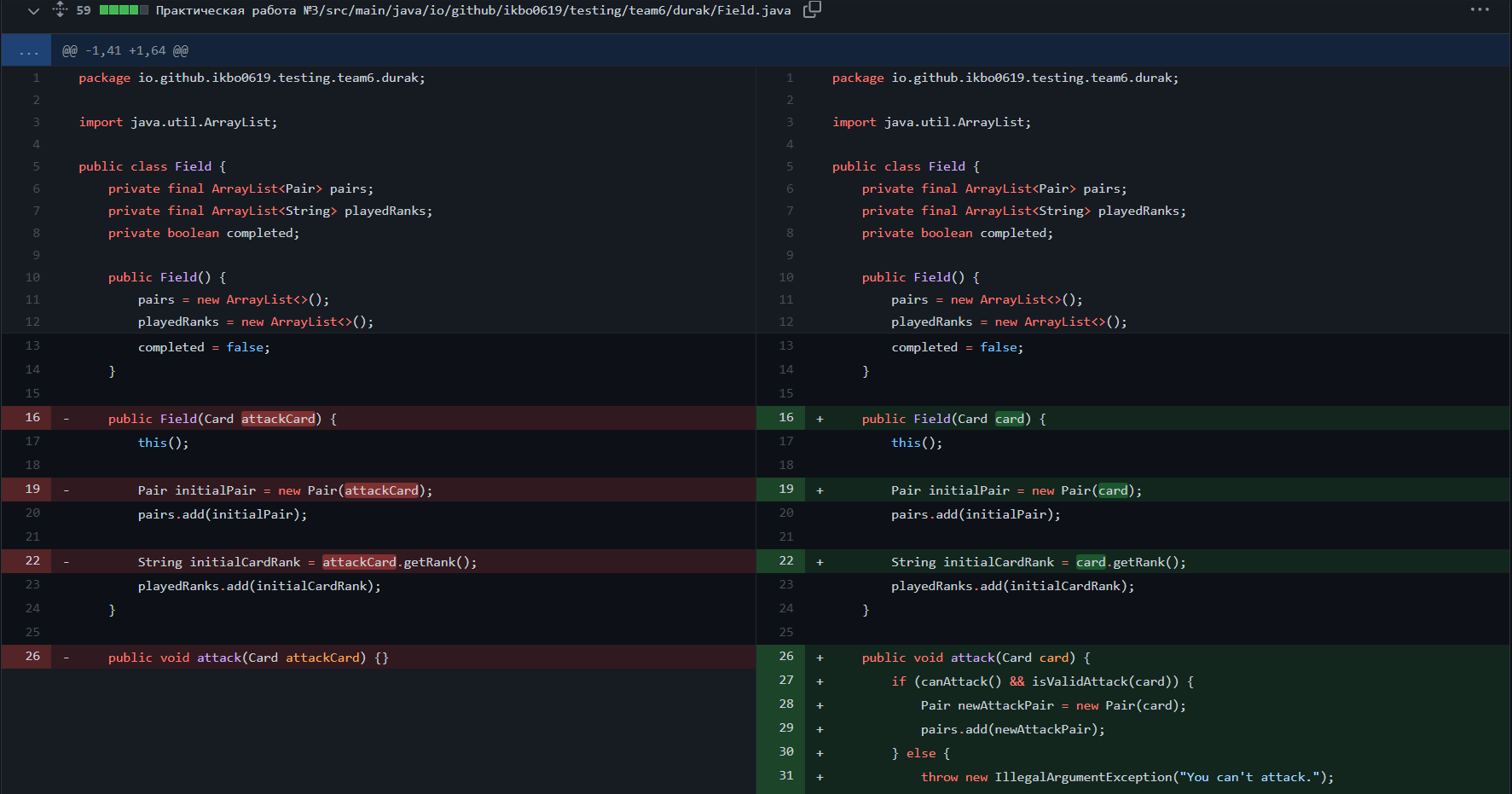
Далее, по составленным тестам был написан код для методов класса:

  
Рисунок 4 – Изменения в классе пары карт

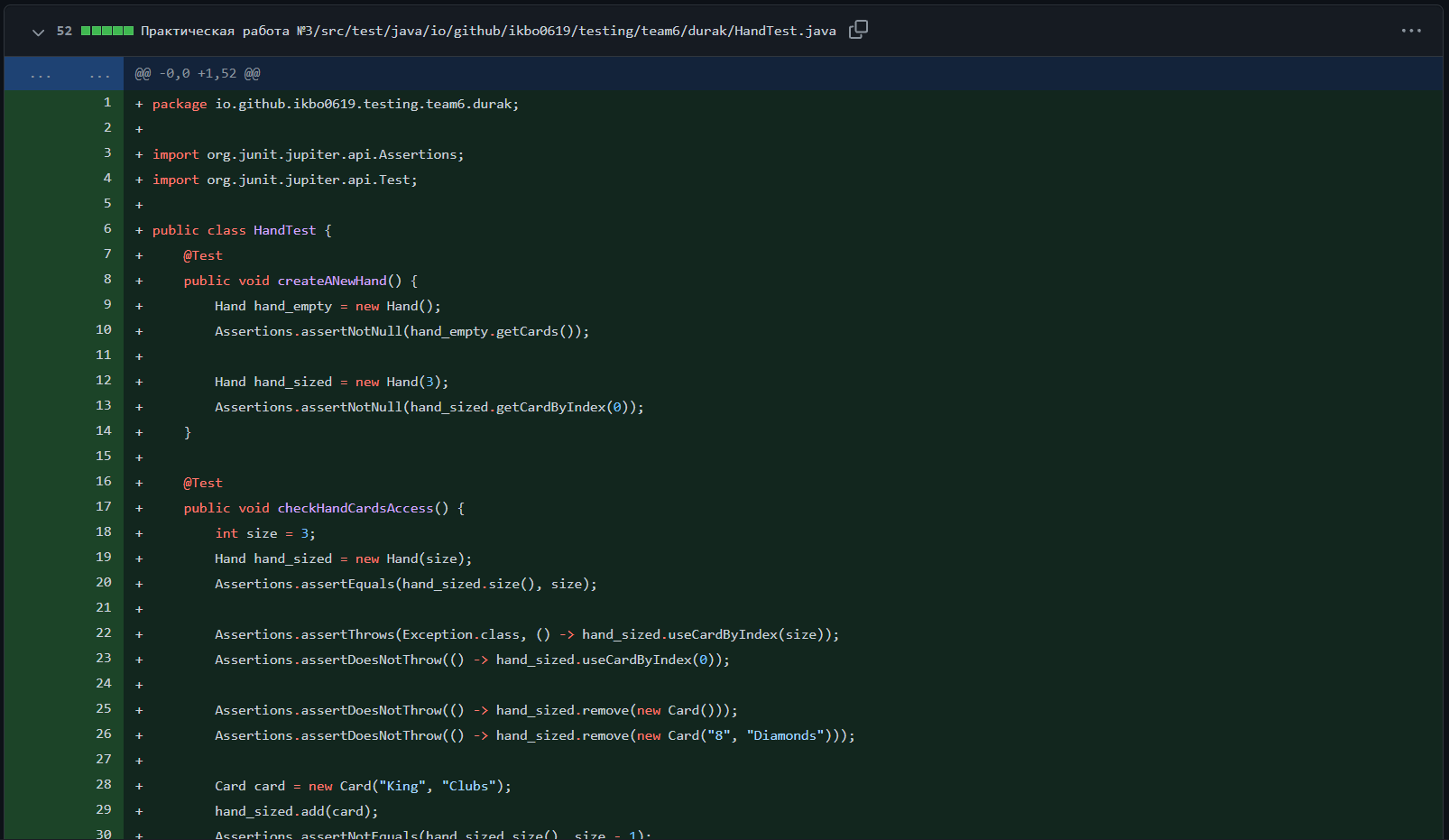
Затем, были составлены тесты для класса Field с определенными для него методами:

  
Рисунок 5 – Подготовленные тесты для класса Field

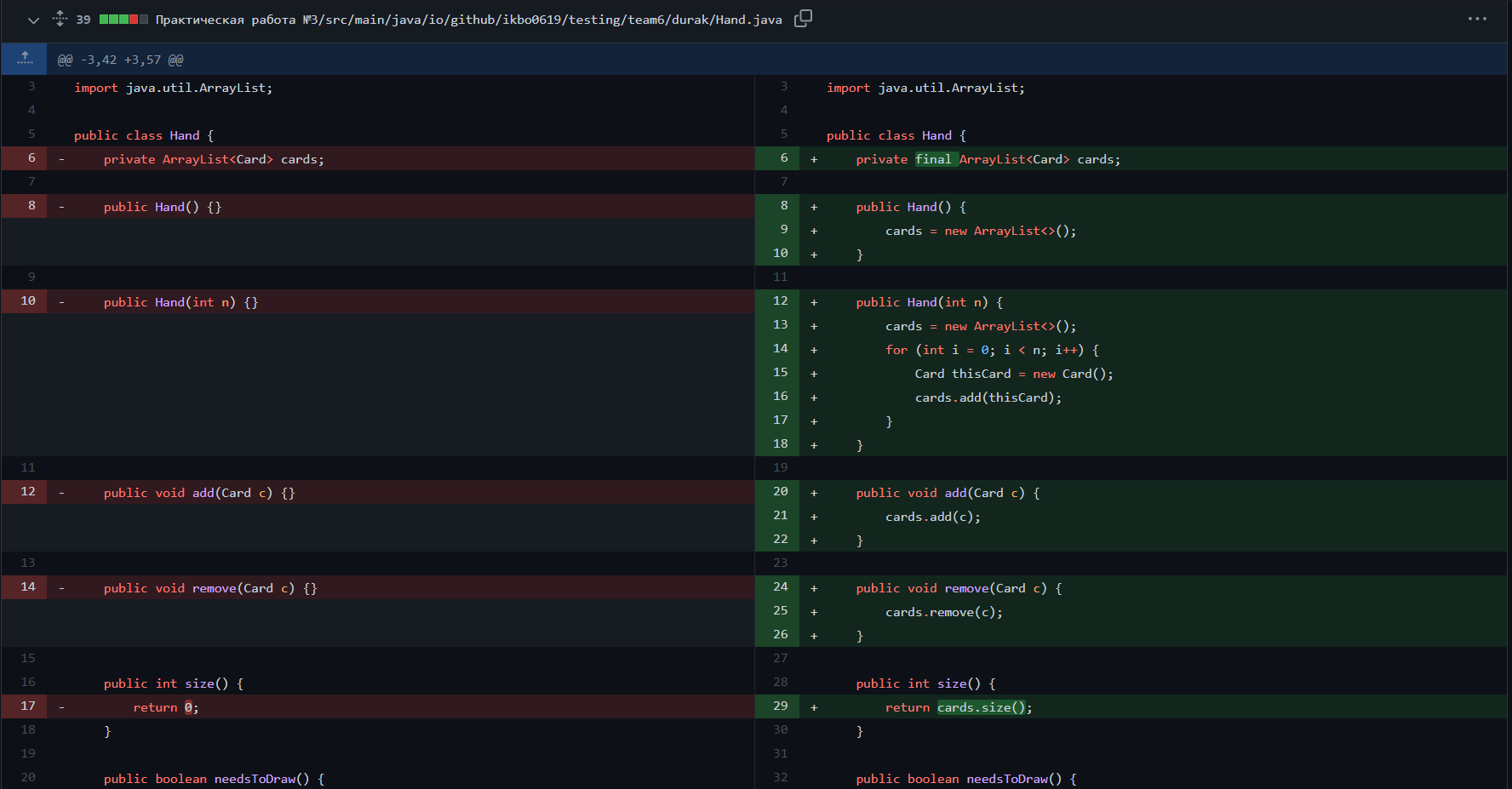
Далее, по составленным тестам был написан код для методов класса:

  
Рисунок 6 – Изменения в классе Field

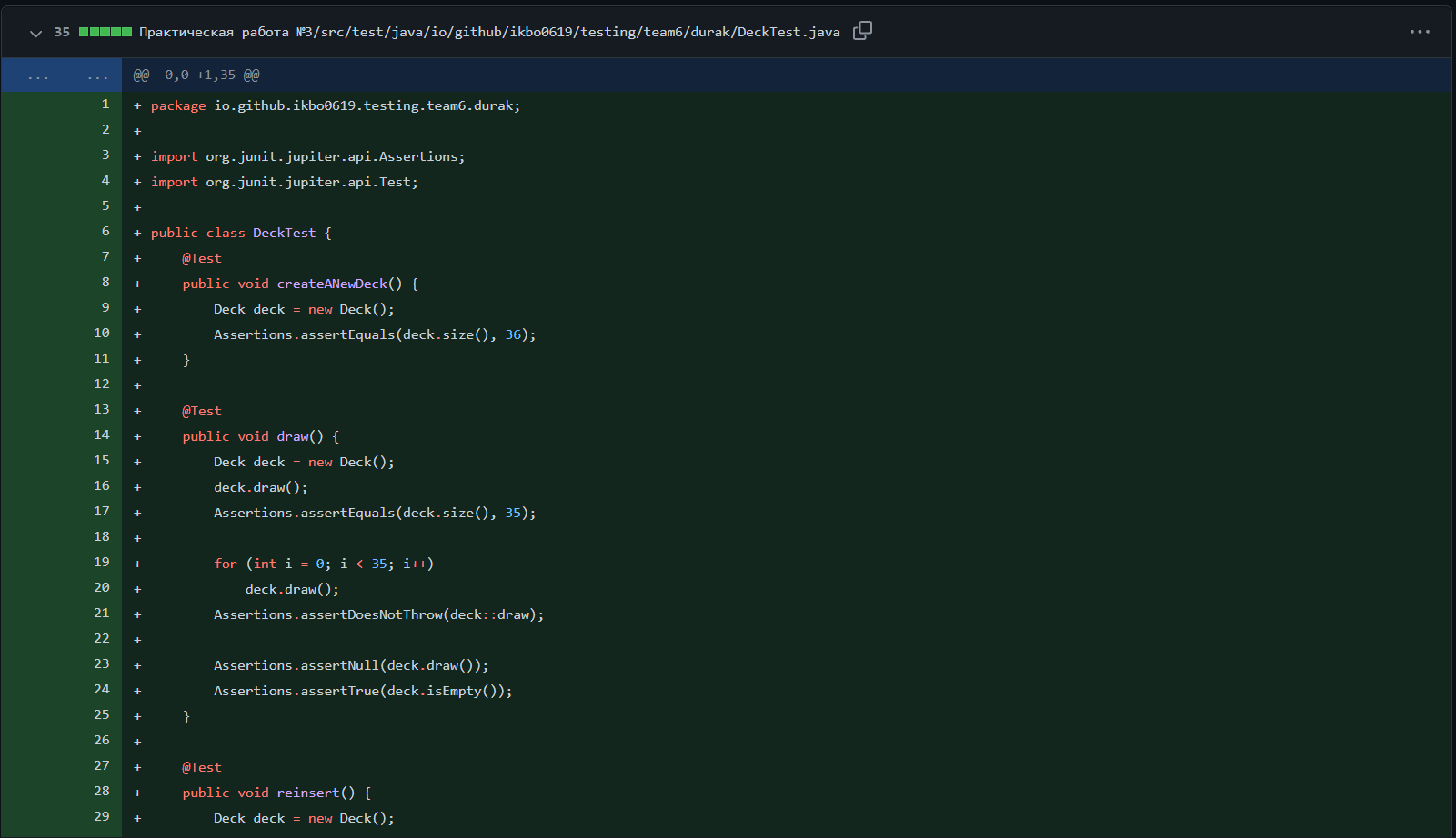
Затем, были составлены тесты для класса набора карт игрока Hand с определенными для него методами:

  
Рисунок 7 – Подготовленные тесты для класса набора карт игрока Hand

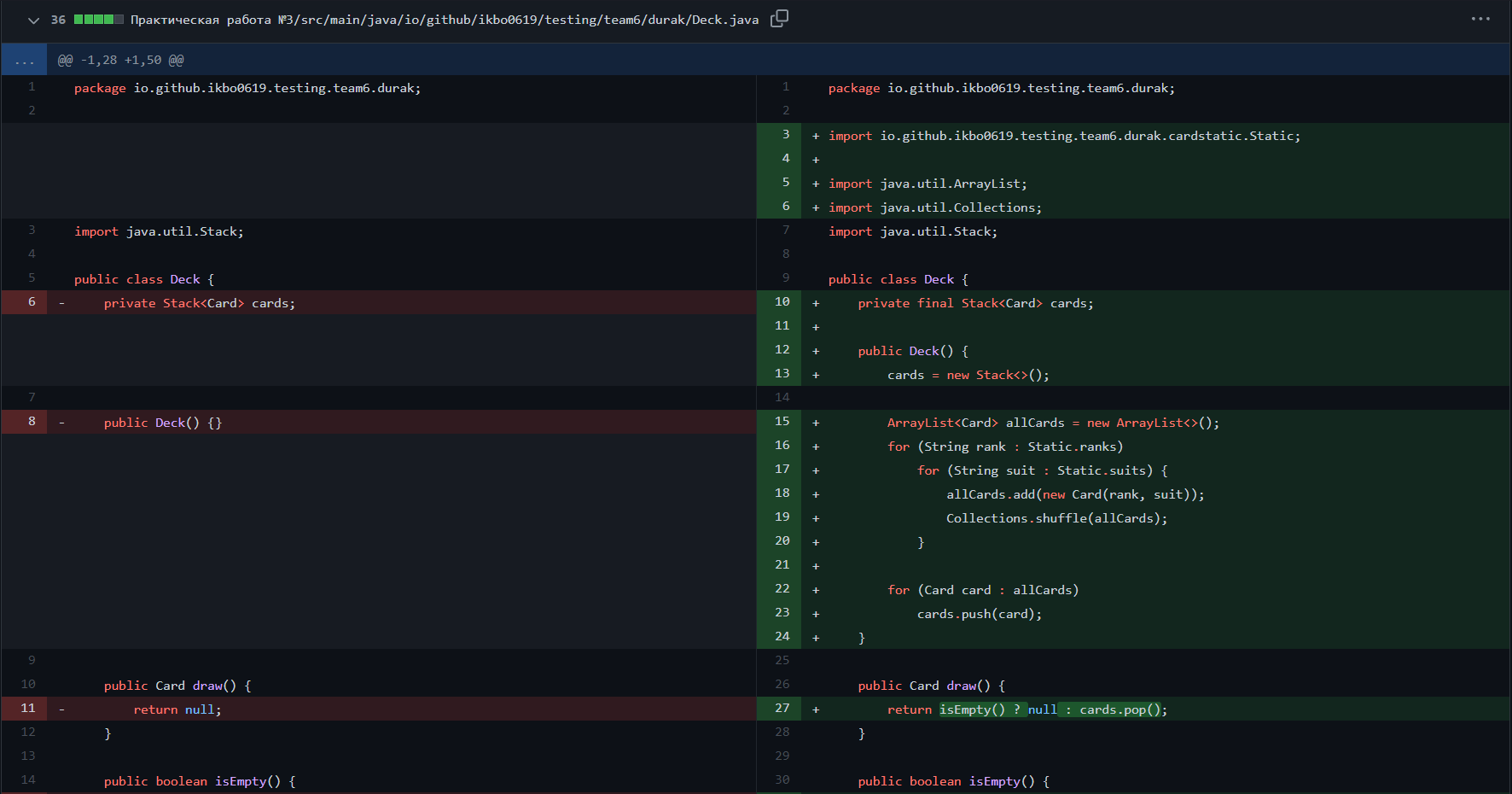
Далее, по составленным тестам был написан код для методов класса:

  
Рисунок 8 – Изменения в классе набора карт игрока

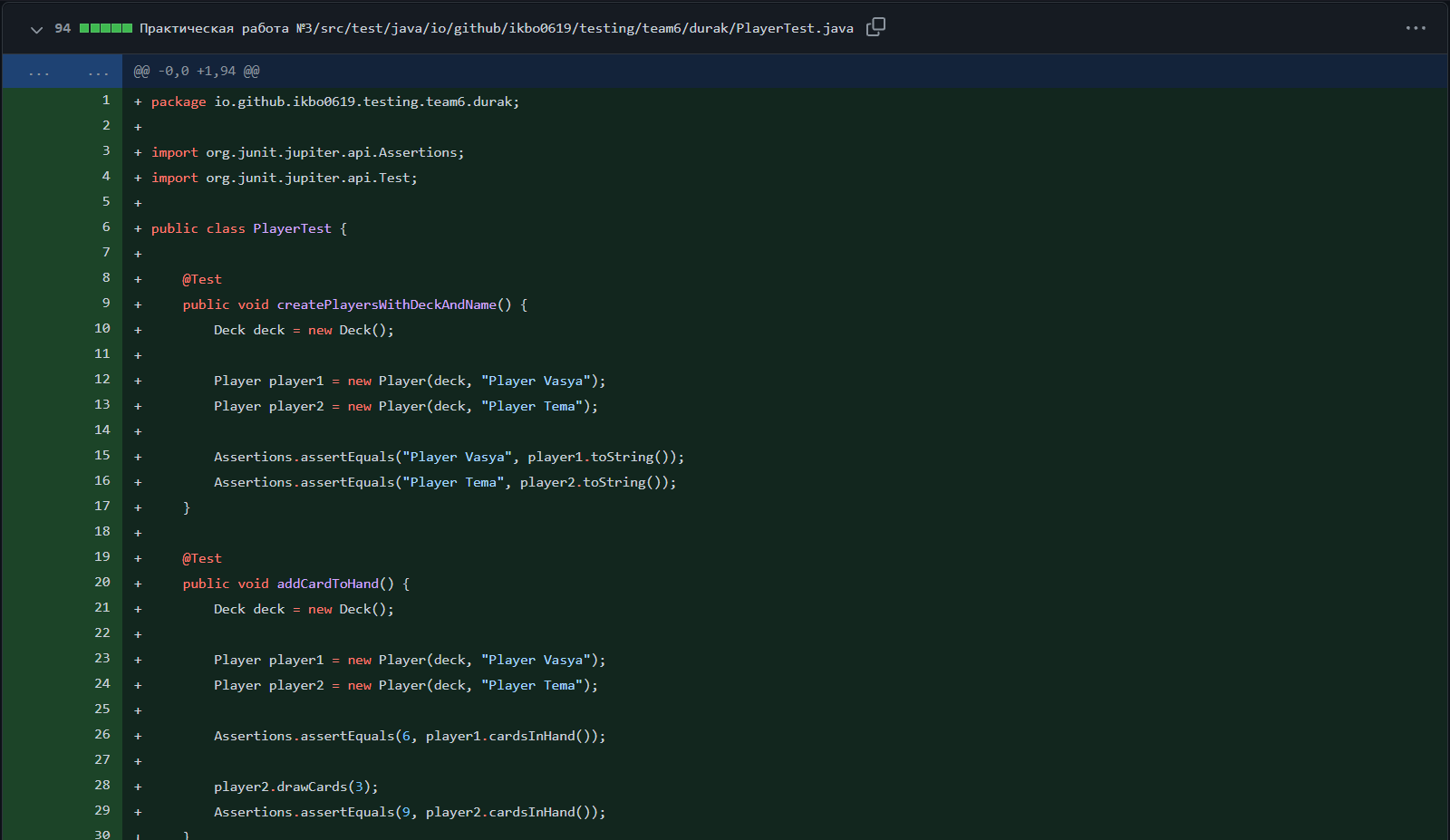
Затем, были составлены тесты для класса колоды карт Deck с определенными для него методами:

  
Рисунок 9 – Подготовленные тесты для класса колоды карт Deck

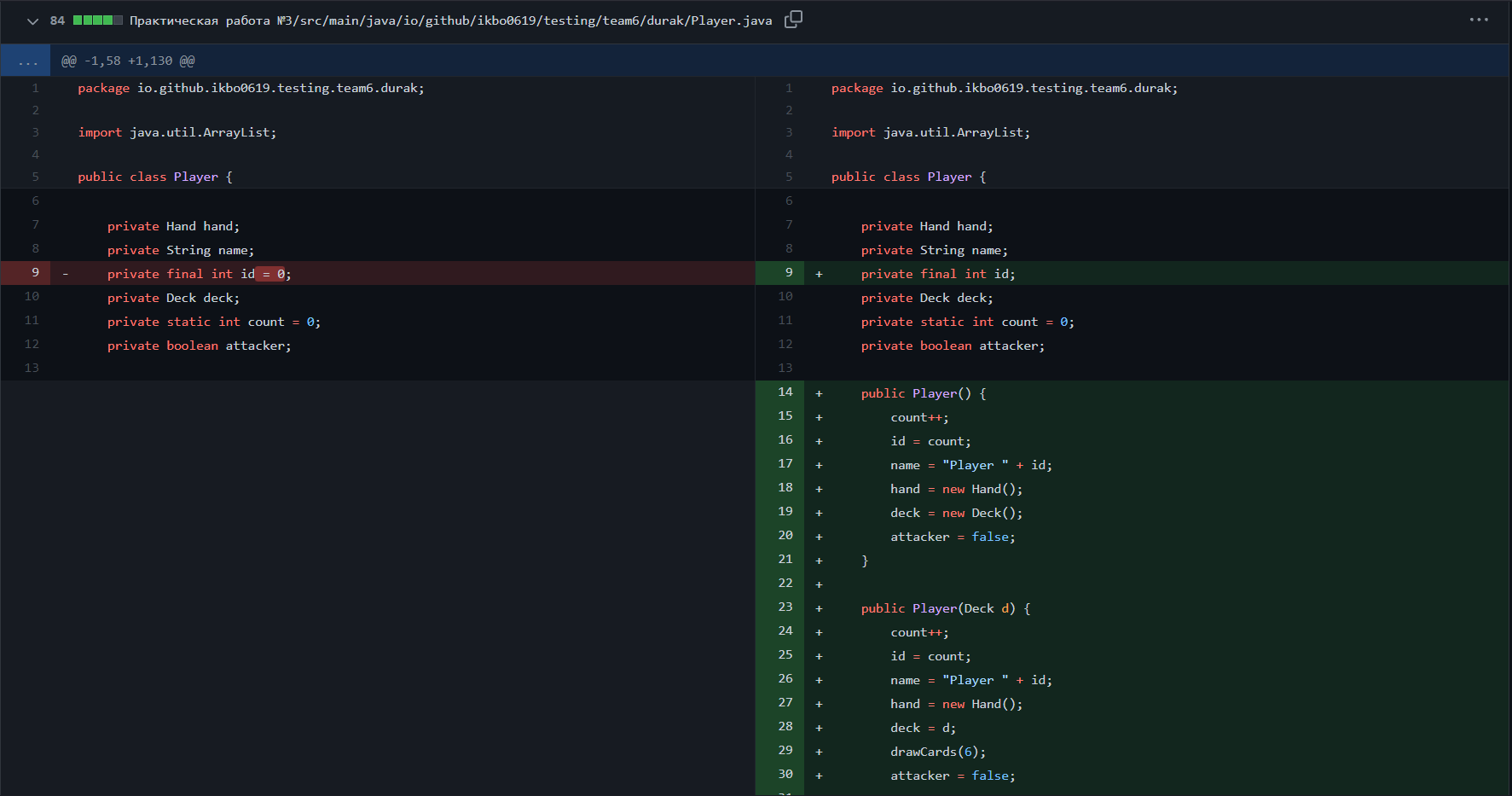
Далее, по составленным тестам был написан код для методов класса:

  
Рисунок 10 – Изменения в классе колоды карт Deck

Затем, были составлены тесты для класса игрока Player с определенными для него методами:

  
Рисунок 11 – Подготовленные тесты для класса игрока Player

Далее, по составленным тестам был написан код для методов класса:

  
Рисунок 12 – Изменения в классе игрока Player

В конце концов, был написан код для методов класса игровой логики Durak, от тестирования которого пришлось отказаться в силу сложности тестирования игровой логики карточной игры в целом:

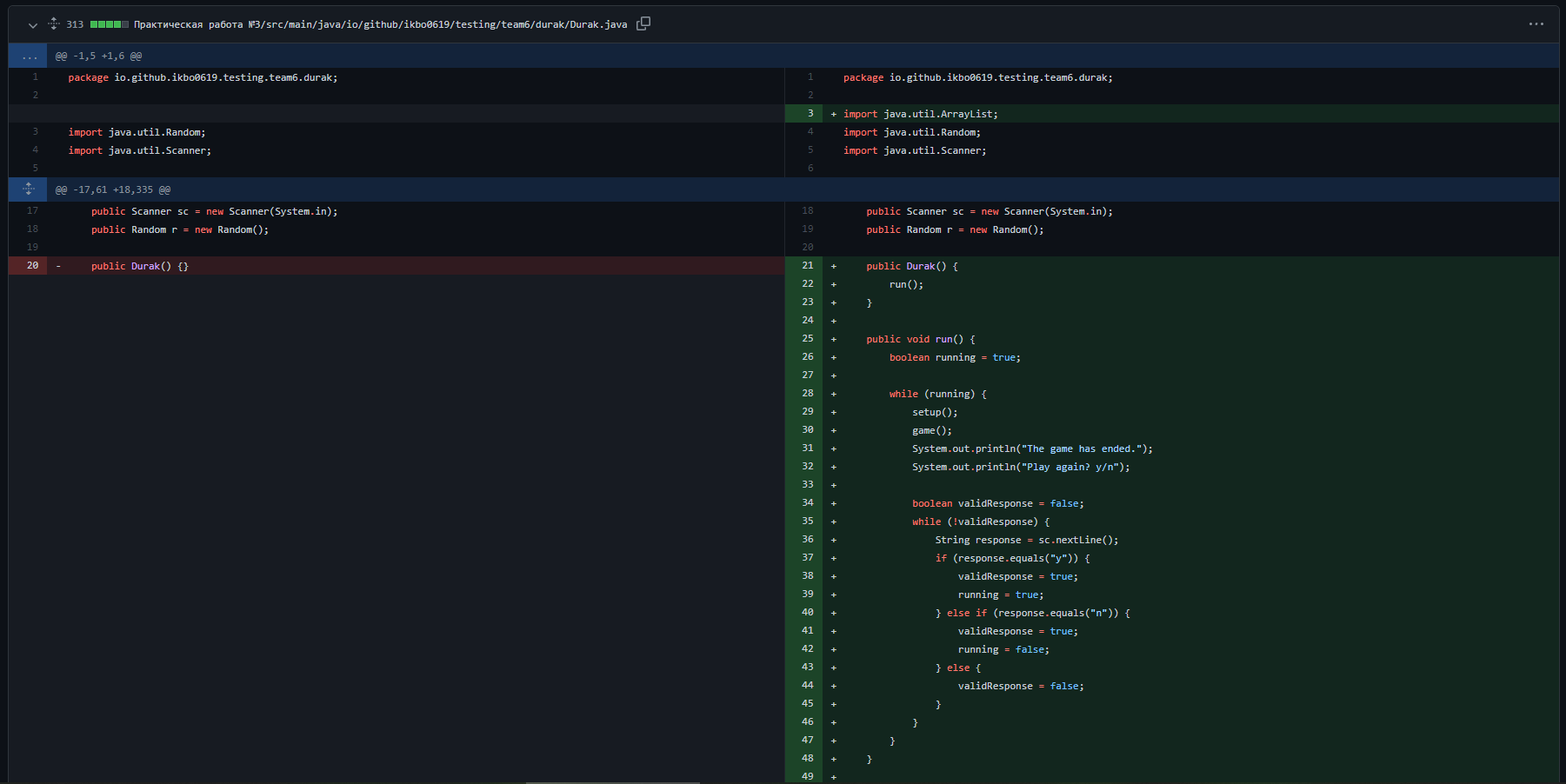


Рисунок 13 – Изменения в классе игрока Durak

# **5. Заключение**

В рамках данной практической работы были изучены:

* Методология «разработка через тестирование» (“Test Driven Development”) и её основные принципы;
* Методология «разработка через поведение» (“Behavior Driven Development”) и её основные принципы;
* Основные принципы использования модульного тестирования.

Во время выполнения практической работы был получен опыт:

* разработки карточной игры «Дурак» для консоли;
* написания Unit-тестов и после разработка ПО согласно методологии «разработки через тестирование» (TDD).

Во время выполнения практической работы был сделан вывод о том, что, прежде чем тратить время на тестирование, нужно сначала проверить, можно ли оценить саму функцию с помощью обычных игровых элементов управления.

Также, если легко (по предварительной оценке) тестируется функция, которая нестабильна и взаимосвязана с другими, что при ошибке в ней создаст очень много проблем при отладке, то применение методологии «разработки через тестирование» для разработки данной функции более чем оправдано.

# **6. Список использованных источников**

Кент Бек. Экстремальное программирование: разработка через тестирование – г. Санкт-Петербург, Питер, 2003 – 225 стр.

Правильное TDD [Электронный ресурс] – 2021 – Режим доступа: <https://habr.com/ru/post/573016/>, свободный

Что такое TDD и BDD на пальцах [Электронный ресурс] – 2021 – Режим доступа: <https://clck.ru/MJVuz>, свободный

TDD и BDD с примерами [Электронный ресурс] – 2021 – Режим доступа: <https://ekaterinagoltsova.github.io/posts/tdd-and-bdd/>, свободный

TDD in Game Dev [Электронный ресурс] – 2021 – Режим доступа: <https://medium.com/game-dev-channel/tdd-in-game-dev-b4684a0a49db>, свободный

Юнит-тестирование для чайников [Электронный ресурс] – 2021 – Режим доступа: <https://habr.com/ru/post/169381/>, свободный

Зачем нужны юнит-тесты [Электронный ресурс] – 2021 – Режим доступа: <https://tproger.ru/translations/unit-tests-purposes/>, свободный