

МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«МИРЭА – Российский технологический университет»**

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

Отчет по практической работе №3

по дисциплине «Тестирование и верификация ПО»

**Выполнили:**

Студенты группы ИКБО-15-22 Оганнисян Г.А.

**Проверил:**

Доцент Чернов Е.А.

Содержание

[Определение персональное варианта 3](#_Toc178801956)

[Реализация по TDD 3](#_Toc178801957)

[1. Пример тестов TDD для угадывания мелодий: 3](#_Toc178801958)

[2. Запуск теста. 3](#_Toc178801959)

[3. Реализация функции. 3](#_Toc178801960)

[4. Повторный запуск теста: 4](#_Toc178801961)

[Реализация по BDD 4](#_Toc178801962)

[1. Пример сценариев BDD 4](#_Toc178801963)

[2. Автоматизация сценариев BDD 4](#_Toc178801964)

[ЗАКЛЮЧЕНИЕ 6](#_Toc178801965)

[ПРИЛОЖЕНИЯ 7](#_Toc178801966)

Определение персональное варианта

Моё задание по номеру студенческого: 11 - Игра «Угадай мелодию» (3 игрока, по очереди проигрывается мелодия и спрашивается, что это за мелодия).

Реализация по TDD

1. Пример тестов TDD для угадывания мелодий:

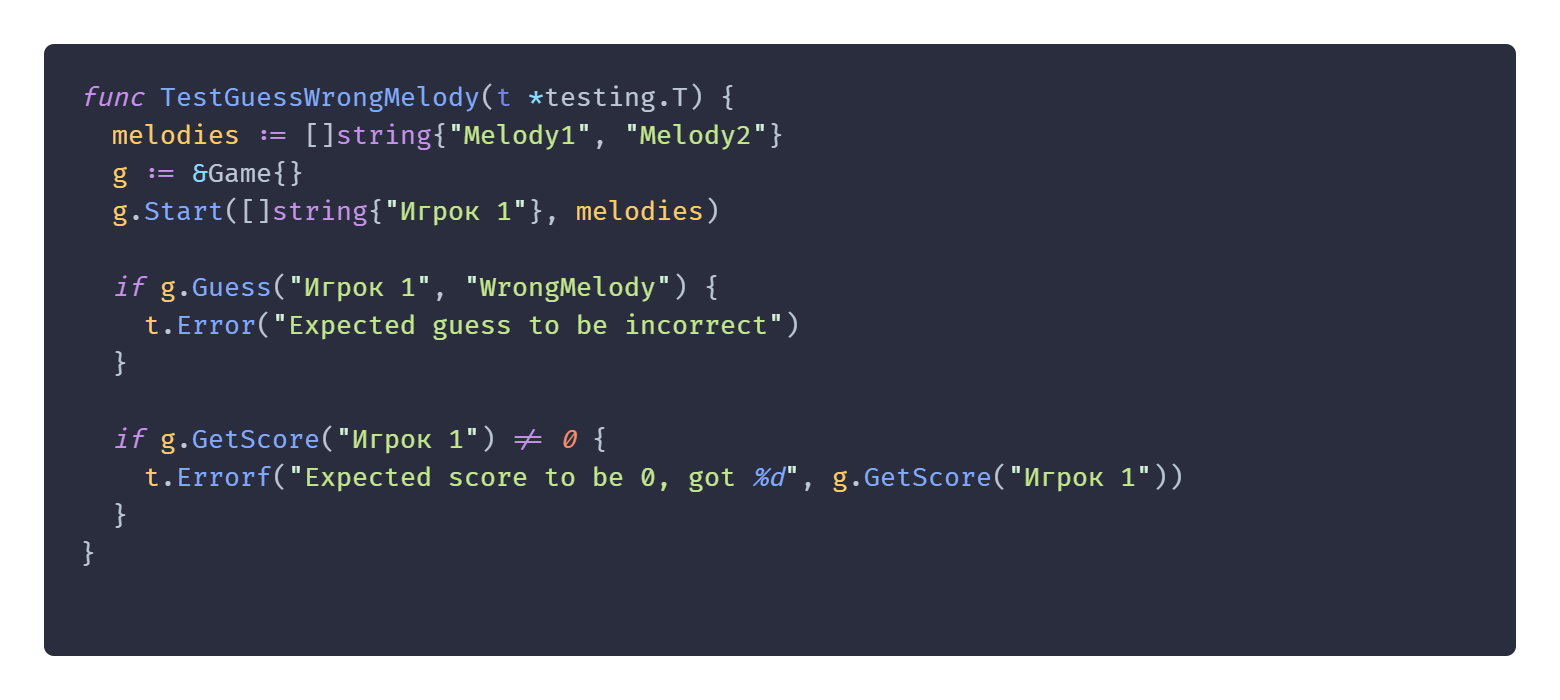


Рисунок 1 – Пример одного теста TDD.

Остальные тесты представлены в приложении А.

1. Запуск теста.

При запуске теста выводиться ошибка, т.к функция ещё не реализована.

1. ****Реализация функции****.

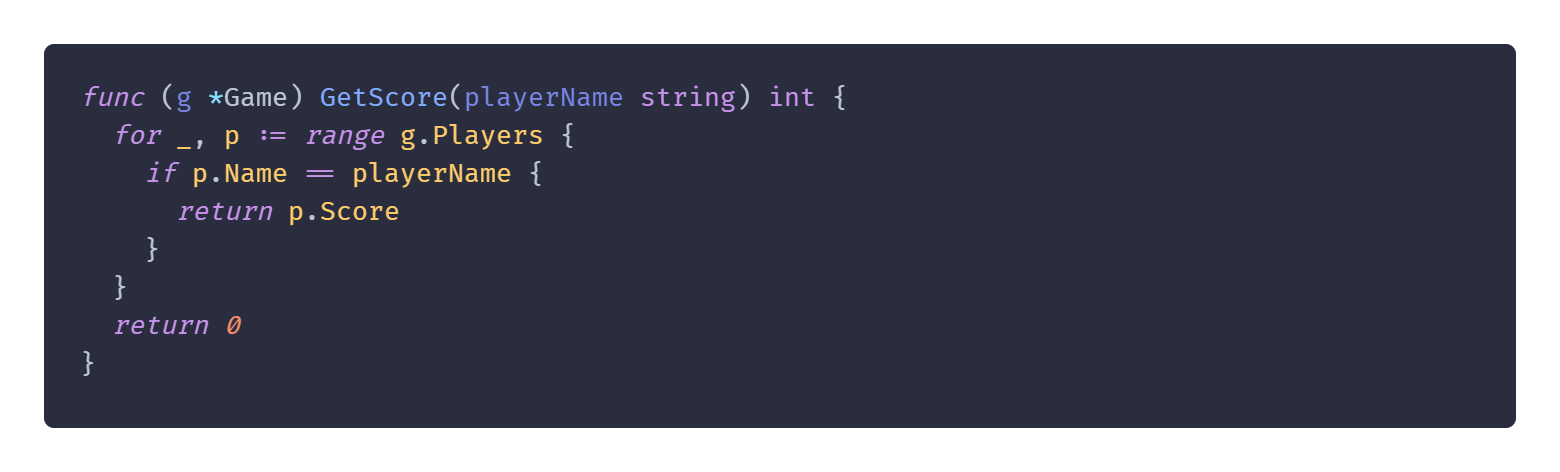


Рисунок 2 – Реализованная функция

1. Повторный запуск теста:

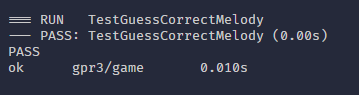


Рисунок 3 – Успешно пройденный тест

Реализация по BDD

1. Пример сценариев BDD

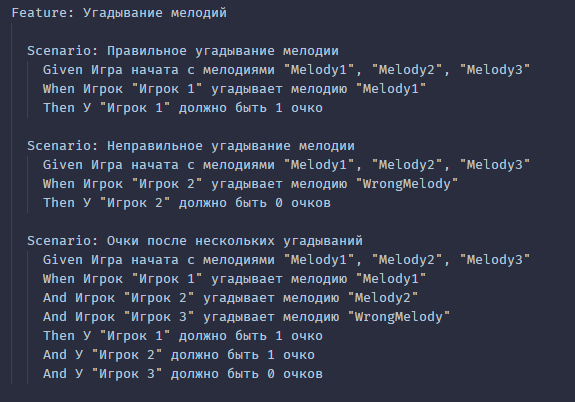


Рисунок 4 – Сценарий BDD

1. Автоматизация сценариев BDD

Для автоматизации сценария BDD была использована библиотека Ginkgo с Gomega. Код с реализацией представлен в приложении Б.

Вывод тестов:

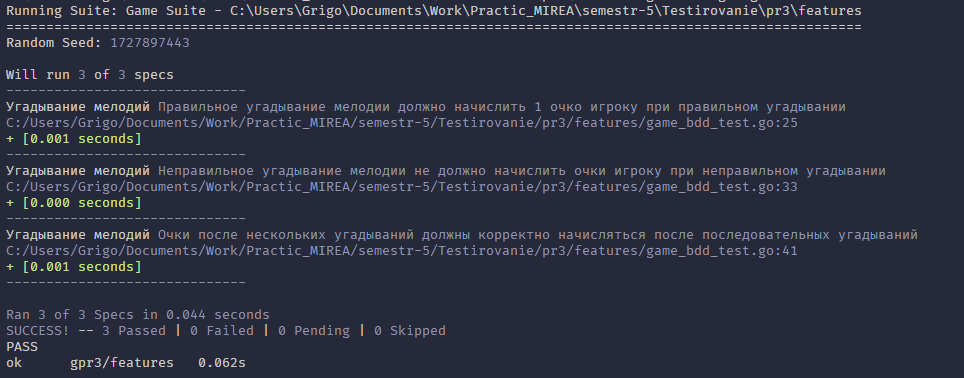


Рисунок 5 – Результат вывода тестов.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения задания была разработана игра «Угадай мелодию», использован полный цикл методологий TDD и BDD. Программа успешно протестирована и поддерживает основные функции, такие как угадывание мелодии, подсчёт очков и очередность игроков.

Методологии TDD и BDD помогли структурировать процесс разработки и обеспечить высокое качество кода с минимальным количеством ошибок.

# ПРИЛОЖЕНИЯ

Приложение А – Примеры TDD тестов кода игры GoLang.

Приложение Б – Примеры BDD тестов кода игры GoLang.

Приложение B – Код игры GoLang.

Приложение Г – Код запуска GoLang.

*Приложение А – Примеры TDD тестов кода игры GoLang.*

package game

import (

  "testing"

)

func TestGuessCorrectMelody(t \*testing.T) {

  melodies := []string{"Melody1", "Melody2"}

  g := &Game{}

  g.Start([]string{"Игрок 1"}, melodies)

  if !g.Guess("Игрок 1", "Melody1") {

    t.Error("Expected guess to be correct")

  }

  if g.GetScore("Игрок 1") != 1 {

    t.Errorf("Expected score to be 1, got %d", g.GetScore("Игрок 1"))

  }

}

func TestGuessWrongMelody(t \*testing.T) {

  melodies := []string{"Melody1", "Melody2"}

  g := &Game{}

  g.Start([]string{"Игрок 1"}, melodies)

  if g.Guess("Игрок 1", "WrongMelody") {

    t.Error("Expected guess to be incorrect")

  }

  if g.GetScore("Игрок 1") != 0 {

    t.Errorf("Expected score to be 0, got %d", g.GetScore("Игрок 1"))

  }

}

func TestNextMelodyAndPlayer(t \*testing.T) {

  melodies := []string{"Melody1", "Melody2", "Melody3"}

  g := &Game{}

  g.Start([]string{"Игрок 1", "Игрок 2"}, melodies)

  // Первый игрок угадывает правильно

  g.Guess("Игрок 1", "Melody1")

  // Проверка счёта первого игрока

  if g.GetScore("Игрок 1") != 1 {

    t.Errorf("Expected score to be 1 for Игрок 1, got %d", g.GetScore("Игрок 1"))

  }

  if g.CurrentMelody != 1 {

    t.Errorf("Expected CurrentMelody to be 1, got %d", g.CurrentMelody)

  }

  // Второй игрок угадывает

  g.Guess("Игрок 2", "Melody2")

  if g.GetScore("Игрок 2") != 1 {

    t.Errorf("Expected score to be 1 for Игрок 2, got %d", g.GetScore("Игрок 2"))

  }

  if g.CurrentMelody != 2 {

    t.Errorf("Expected CurrentMelody to be 2, got %d", g.CurrentMelody)

  }

}

*Приложение Б – Примеры BDD тестов кода игры GoLang.*

package game\_test

import (

  . "gpr3/game"

  "testing"

  . "github.com/onsi/ginkgo/v2"

  . "github.com/onsi/gomega"

)

// Точка входа для обычного запуска через go test

func TestGame(t \*testing.T) {

  RegisterFailHandler(Fail)

  RunSpecs(t, "Game Suite")

}

var \_ = Describe("Угадывание мелодий", func() {

  var g \*Game

  BeforeEach(func() {

    g = &Game{}

  })

  Describe("Правильное угадывание мелодии", func() {

    It("должно начислить 1 очко игроку при правильном угадывании", func() {

      g.Start([]string{"Игрок 1"}, []string{"Melody1", "Melody2", "Melody3"})

      Expect(g.Guess("Игрок 1", "Melody1")).To(BeTrue())

      Expect(g.GetScore("Игрок 1")).To(Equal(1))

    })

  })

  Describe("Неправильное угадывание мелодии", func() {

    It("не должно начислить очки игроку при неправильном угадывании", func() {

      g.Start([]string{"Игрок 2"}, []string{"Melody1", "Melody2", "Melody3"})

      Expect(g.Guess("Игрок 2", "WrongMelody")).To(BeFalse())

      Expect(g.GetScore("Игрок 2")).To(Equal(0))

    })

  })

  Describe("Очки после нескольких угадываний", func() {

    It("должны корректно начисляться после последовательных угадываний", func() {

      g.Start([]string{"Игрок 1", "Игрок 2", "Игрок 3"}, []string{"Melody1", "Melody2", "Melody3"})

      Expect(g.Guess("Игрок 1", "Melody1")).To(BeTrue())

      Expect(g.Guess("Игрок 2", "Melody2")).To(BeTrue())

      Expect(g.Guess("Игрок 3", "WrongMelody")).To(BeFalse())

      Expect(g.GetScore("Игрок 1")).To(Equal(1))

      Expect(g.GetScore("Игрок 2")).To(Equal(1))

      Expect(g.GetScore("Игрок 3")).To(Equal(0))

    })

  })

})

*Приложение B –Код игры GoLang.*

package game

import (

  "strings"

)

type Player struct {

  Name  string

  Score int

}

type Game struct {

  Players       []\*Player

  Melodies      []string

  CurrentPlayer int

  CurrentMelody int

}

func (g \*Game) Start(players []string, melodies []string) {

  g.Players = make([]\*Player, len(players))

  for i, name := range players {

    g.Players[i] = &Player{Name: name, Score: 0}

  }

  g.Melodies = melodies

  g.CurrentPlayer = 0

  g.CurrentMelody = 0

}

func (g \*Game) Guess(playerName, guess string) bool {

  // Найти игрока

  var player \*Player

  for \_, p := range g.Players {

    if p.Name == playerName {

      player = p

      break

    }

  }

  if player == nil {

    return false

  }

  // Проверить угадывание

  correct := strings.ToLower(g.Melodies[g.CurrentMelody]) == strings.ToLower(guess)

  if correct {

    player.Score++

  }

  // Переход к следующей мелодии и игроку

  g.CurrentMelody = (g.CurrentMelody + 1) % len(g.Melodies)

  g.CurrentPlayer = (g.CurrentPlayer + 1) % len(g.Players)

  return correct

}

func (g \*Game) GetScore(playerName string) int {

  for \_, p := range g.Players {

    if p.Name == playerName {

      return p.Score

    }

  }

  return 0

}

*Приложение Г – Код запуска GoLang.*

package main

import (

  "bufio"

  "fmt"

  "gpr3/game"

  "os"

)

func main() {

  // Список игроков

  players := []string{"Игрок 1", "Игрок 2", "Игрок 3"}

  // Список мелодий

  melodies := []string{"Melody1", "Melody2", "Melody3"}

  // Инициализация игры

  game := &game.Game{}

  game.Start(players, melodies)

  // Основная игровая логика

  scanner := bufio.NewScanner(os.Stdin)

  for {

    player := game.Players[game.CurrentPlayer]

    fmt.Printf("Ход игрока %s. Угадайте мелодию: ", player.Name)

    scanner.Scan()

    guess := scanner.Text()

    if game.Guess(player.Name, guess) {

      fmt.Println("Правильно!")

    } else {

      correctMelody := game.Melodies[(game.CurrentMelody-1+len(game.Melodies))%len(game.Melodies)]

      fmt.Printf("Неправильно! Правильный ответ: %s\n", correctMelody)

    }

    fmt.Printf("Очки игрока %s: %d\n", player.Name, player.Score)

  }

}