

|  |  |  |  |
| --- | --- | --- | --- |
| |  | | --- | | МИНОБРНАУКИ РОССИИ | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |
|  | Институт информационных технологий (ИТ) |
|  | Кафедра инструментального и прикладного программного обеспечения (ИиППО) |

|  |  |  |  |
| --- | --- | --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1** | | | |
| **по дисциплине** | | | |
| **«Тестирование и верификация программного обеспечения» на тему**  **«Командная работа по специфицированию и тестированию»** | | | |
| Выполнили студенты группы ИКБО-20-19 | | Анваржонов Ж.Т. | |
| Принял | | Мельников Д.А. | |
| Практические работы выполнены | «31» октября 2021 г. | | Анваржонов Ж Т.    (подписи студентов) | |
| «Зачтено» | « » 2021 г. | | Мельников Д.А.  (подпись руководителя) | |
|  |  | |  | |

Москва 2021

**Практическая работа №3 «Разработка через тестирование»**

**Цель работы:**

1. На основе изучения материала лекций по дисциплине «Тестирова- ние и верификация программного обеспечения» требуется выполнить следующее.

Реализовать приложение c нуля согласно заданию с использованием методологии разработки через тестирование TDD (Test Driven Development) или разработки, управляемой поведением, BDD (Behavior Driven Development) по выбору студента.

На каждом промежуточном шаге разработки (рабочий код и тесты) осуществлять commit в систему контроля версий.

Сложный графический интерфейс для разрабатываемого программного продукта необязателен. Вся функциональность должна быть реализована в классах, допустимо использование консольного интерфейса, где явно не нужен графический интерфейс.

1. Продемонстрировать функционал приложения и код, так, чтобы были видны diff (различия по строкам) между commit, начиная с первого, и можно было отследить последовательность разработки (в средстве Git или на GitHub).
2. Оформить титульный лист отчета.
3. Произвести выбор и описать методологию разработки ПО.
4. Описать принцип работы ПО.
5. Продемонстрировать работоспособность, разрабатываемого программного продукта, вставить в отчет скриншоты ПО
6. Вставить в отчет графическое отображение Branch из GitHub.
7. Защитить практическую работу.

**Вариант задания: «Электронный терапевт»**

**Выполнение задания:**

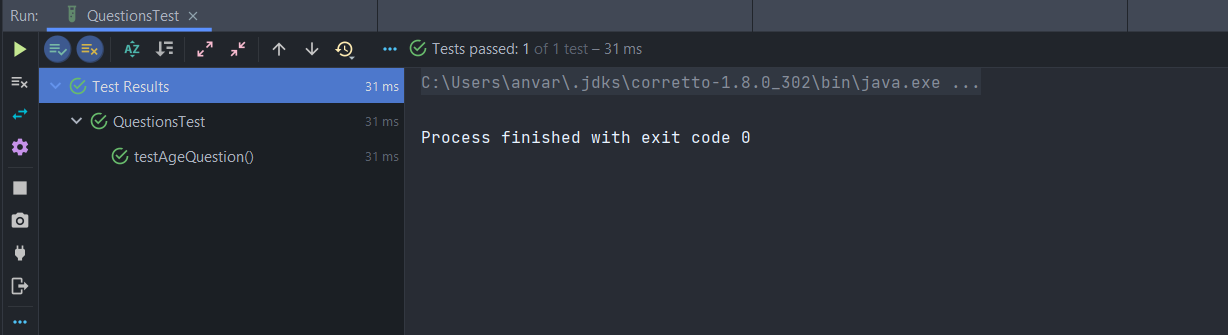
1. Cоздадим базу вопросов терапевта, его конструктор и геттер:

*package* anvarzhonov;  
  
*public enum* Questions {  
 NAME("Укажите своё имя?"),  
 SEX("Укажите свой пол? (М или Ж)"),  
 AGE("Сколько вам лет?"),  
 WHAT\_PAIN("Что у вас болит?"),  
  
 HEAD\_ACHE("Если у вас болит голова - нажмите 1"),  
 HEAD\_FIVE\_DAYS("Болит больше пяти дней?"),  
 HEAD\_THROBBING\_PAIN("Боль невыносимая?"),  
 HIGH\_PRESSURE("Давление повышенное?"),  
 HEAD\_NERVOUS("У вас был стресс в течении 5 дней?"),  
  
 TEMPERATURE\_ACHE("Если у вас температура - нажмите 2"),  
 TEMPERATURE\_COVID("Вы переболели ковидом?"),  
 SMELLS("Вы чувствуете запахи?"),  
 COUGH("У вас есть кашель?"),  
  
 COLD("Если у вас насморк или болит горло - нажмите 3"),  
 DRY\_COUGH("У вас сухой кашель?"),  
  
 BELLY\_ACHE("Если у вас болит живот - нажмите 4"),  
 BELLY\_DIARRHEA("У вас есть диарея?"),  
 BELLY\_URINA\_BLOOD("Есть кровь при мочеиспускании?"),  
 BELLY\_ALCOHOL("Вы часто употребляете алкоголь?");  
  
  
  
  
 *private final* String question;  
  
 Questions(String question) {  
 *this*.question = question;  
 }  
  
 *public* String getQuestion() {  
 *return* question;  
 }  
}

1. Сделаем тест нашего класса:

*package* anvarzhonov;  
  
*import* org.junit.jupiter.api.*Test*;  
  
*public class* QuestionsTest {  
  
 *@Test  
 void* testAgeQuestion() {  
 String AgeQuestion = "Сколько вам лет?";  
 *assert*(AgeQuestion.equals(Questions.AGE.getQuestion()));  
  
 String sexQuestion = "Укажите свой пол? (М или Ж)";  
 *assert*(sexQuestion.equals(Questions.SEX.getQuestion()));  
 }  
}

И покажем успешное выполнение тестов:



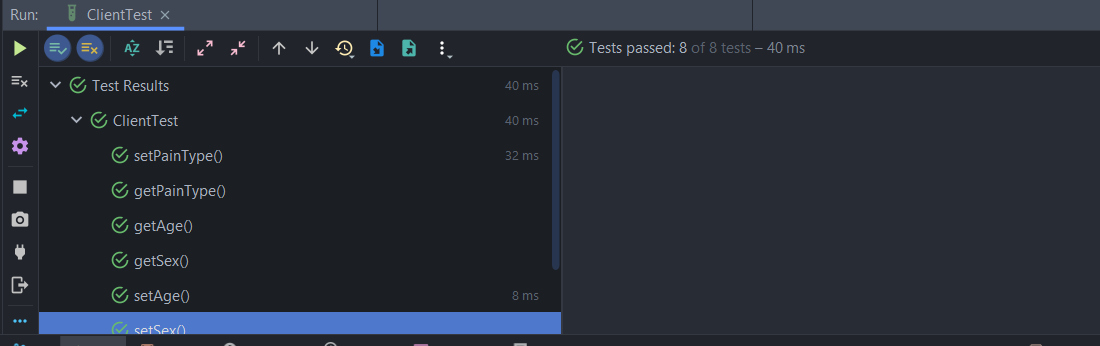
1. Создадим класс самого клиента:

*package* anvarzhonov;  
  
*public class* Client {  
 *private* String name;  
 *private int* age;  
 *private* String sex;  
 *private int* painType;  
  
  
  
 *public* String getName() {  
 *return* name;  
 }  
  
 *public void* setName(String name) {  
 *this*.name = name;  
 }  
  
 *public int* getAge() {  
 *return* age;  
 }  
  
 *public void* setAge(*int* age) {  
 *this*.age = age;  
 }  
  
 *public* String getSex() {  
 *return* sex;  
 }  
  
 *public void* setSex(String sex) {  
 *this*.sex = sex;  
 }  
  
 *public int* getPainType() {  
 *return* painType;  
 }  
  
 *public void* setPainType(*int* painType) {  
 *this*.painType = painType;  
 }  
}

И сделаем тест для этого класса:

*package* anvarzhonov;  
  
*import* org.junit.jupiter.api.Assertions;  
*import* org.junit.jupiter.api.*BeforeEach*;  
*import* org.junit.jupiter.api.*Test*;  
  
*public class* ClientTest {  
 Client client;  
  
  
 *@BeforeEach  
 public void* setUp(){  
 client =*new* Client();  
 }  
  
 *@Test  
 void* getName() {  
 client.setName("lflflggl");  
 Assertions.assertEquals("lflflggl", client.getName());  
 }  
  
 *@Test  
 void* setName() {  
 client.setName("lflflggl");  
 Assertions.assertEquals("lflflggl", client.getName());  
 }  
  
 *@Test  
 void* getAge() {  
 client.setAge(21);  
 Assertions.assertEquals(21, client.getAge());  
 }  
  
 *@Test  
 void* setAge() {  
 client.setAge(21);  
 Assertions.assertEquals(21, client.getAge());  
 }  
  
 *@Test  
 void* getSex() {  
 client.setSex("М");  
 Assertions.assertEquals("М", client.getSex());  
 }  
  
 *@Test  
 void* setSex() {  
 client.setSex("М");  
 Assertions.assertEquals("М", client.getSex());  
  
 }  
  
 *@Test  
 void* getPainType() {  
 client.setPainType(4);  
 Assertions.assertEquals(4, client.getPainType());  
 }  
  
 *@Test  
 void* setPainType() {  
 client.setPainType(4);  
 Assertions.assertEquals(4, client.getPainType());  
 }  
}

Покажем результат тестов:



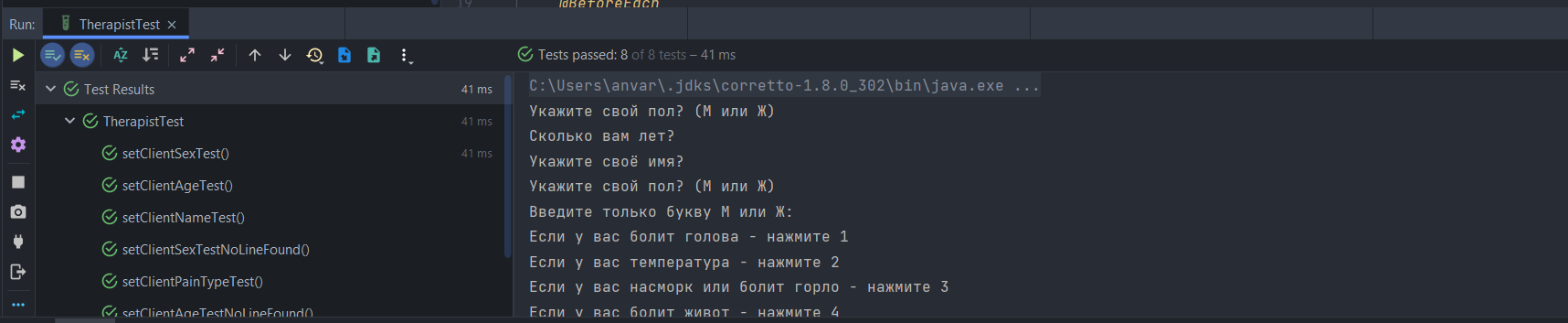
1. Создадим класс Терапевта, который будет задавать вопросы из базы enum.

*package* anvarzhonov;  
  
*import* java.sql.SQLOutput;  
*import* java.util.Objects;  
*import* java.util.Scanner;  
  
*public class* Therapist {  
 *private* Client currentClient;  
 *private* Scanner scanner;  
  
  
 *public void* setClient(Client client) {  
 *this*.currentClient = client;  
 }  
  
 *public* Client getClient() {  
 *return* currentClient;  
 }  
  
 *public void* setClientName() {  
 scanner = *new* Scanner(System.in);  
 System.out.println(Questions.NAME.getQuestion());  
 String name = scanner.nextLine();  
 *while* (name.matches("[0-9]+") || name.equals("")) {  
 System.out.println("Введите свое имя корректно: ");  
 name = scanner.nextLine();  
 }  
 currentClient.setName(name);  
 }  
  
 *public void* setClientSex() {  
 scanner = *new* Scanner(System.in);  
 System.out.println(Questions.SEX.getQuestion());  
 String sex = scanner.nextLine();  
 *while* (!Objects.equals(sex, "М") && !Objects.equals(sex, "Ж")) {  
 System.out.println("Введите только букву М или Ж:");  
 sex = scanner.nextLine();  
  
 }  
 currentClient.setSex(sex);  
 }  
  
 *public void* setClientAge() {  
 scanner = *new* Scanner(System.in);  
 System.out.println(Questions.AGE.getQuestion());  
 String age = scanner.nextLine();  
 *while* (!age.matches("\\d+")) {  
 System.out.println("Введите возраст как целое число:");  
 age = scanner.nextLine();  
 }  
 currentClient.setAge(Integer.parseInt(age));  
 }  
  
 *public void* setClientPainType() {  
 scanner = *new* Scanner(System.in);  
 System.out.println(Questions.HEAD\_ACHE.getQuestion());  
 System.out.println(Questions.TEMPERATURE\_ACHE.getQuestion());  
 System.out.println(Questions.COLD.getQuestion());  
 System.out.println(Questions.BELLY\_ACHE.getQuestion());  
  
 String painType = scanner.nextLine();  
 *while* (!painType.matches("[1-4]")) {  
 System.out.println("Введите цифру от 1 до 4");  
 painType = scanner.nextLine();  
 }  
 currentClient.setPainType(Integer.parseInt(painType));  
 }  
  
 *public void* healing() {  
 *switch* (currentClient.getPainType()) {  
 *case* 1:  
 headHealing();  
 *break*;  
 *case* 2:  
 temperatureHealing();  
 *break*;  
 *case* 3:  
 coldHealing();  
 *break*;  
 *case* 4:  
 bellyHealing();  
 *break*;  
 }  
 }  
  
 *public void* headHealing() {  
  
 System.out.println(Questions.HEAD\_FIVE\_DAYS.getQuestion());  
 *boolean* headFiveDays = getClientHealingAnswer();  
  
 System.out.println(Questions.HEAD\_THROBBING\_PAIN.getQuestion());  
 *final boolean* headThrobbingPain = getClientHealingAnswer();  
  
 *if* (headThrobbingPain) {  
 System.out.println(Questions.HIGH\_PRESSURE.getQuestion());  
 *boolean* highPressure = getClientHealingAnswer();  
 *if* (highPressure) {  
 System.out.println("Ваш диагноз: Артериальная гипертония. Вам следует принять Каптоприл. " +  
 "Если не поможет, приходите ещё");  
 }  
 }  
 System.out.println(Questions.HEAD\_NERVOUS.getQuestion());  
 *boolean* headNervous = getClientHealingAnswer();  
 *if* (!headFiveDays) {  
 System.out.println("Если нет - примите парацетомол.");  
 } *else* {  
 System.out.println("Вы скорее всего переутомились, отдохните от всего на природе");  
 }  
  
 }  
  
 *public void* temperatureHealing() {  
 System.out.println(Questions.TEMPERATURE\_COVID.getQuestion());  
 *boolean* temperature = getClientHealingAnswer();  
 *if* (!temperature) {  
  
 System.out.println(Questions.SMELLS.getQuestion());  
 *boolean* covidSmells = getClientHealingAnswer();  
  
 System.out.println(Questions.COUGH.getQuestion());  
 *boolean* covidCough = getClientHealingAnswer();  
 *if* (covidCough && !covidSmells) {  
 System.out.println("Ваш диагноз: У вас Ковид. Вам придется лечь в больницу.");  
 }  
 }*else* {  
 System.out.println("У вас обычная простуда");  
 }  
  
  
 }  
  
 *public void* coldHealing() {  
  
 System.out.println(Questions.DRY\_COUGH.getQuestion());  
 *boolean* dryCough = getClientHealingAnswer();  
  
 *if* (dryCough) {  
 System.out.println("У вас ОРВИ");  
 } *else* {  
 System.out.println("У вас гайморит");  
 }  
 }  
  
 *public void* bellyHealing() {  
 System.out.println(Questions.BELLY\_DIARRHEA.getQuestion());  
 *boolean* bellyDiarhea = getClientHealingAnswer();  
  
 System.out.println(Questions.BELLY\_URINA\_BLOOD.getQuestion());  
 *boolean* bellyUrinaBlood = getClientHealingAnswer();  
  
 System.out.println(Questions.BELLY\_ALCOHOL.getQuestion());  
 *boolean* bellyAlcohol = getClientHealingAnswer();  
  
 *if* (bellyDiarhea)  
 System.out.println("Ваш диагноз: кишечное отравление. Пейте 2 дня Лоперамид, а потом приходите снова");  
 *else if*(bellyUrinaBlood) System.out.println("У вас камни в почках. Вам нужно лечь в больницу");  
 *else if* (bellyAlcohol) System.out.println("У вас язва желудка");  
 *else* System.out.println("У вас гастрит");  
 }  
  
 *private boolean* getClientHealingAnswer() {  
 scanner = *new* Scanner(System.in);  
 String answer = scanner.nextLine();  
 *while* (!answer.equals("Да") && !answer.equals("Нет") &&  
 !answer.equals("да") && !answer.equals("нет")) {  
 System.out.println("Введите только \"Да\" или \"Нет\"");  
 answer = scanner.nextLine();  
 }  
 *return* answer.equals("Да") || answer.equals("да");  
 }  
}

Покажем тест класс:

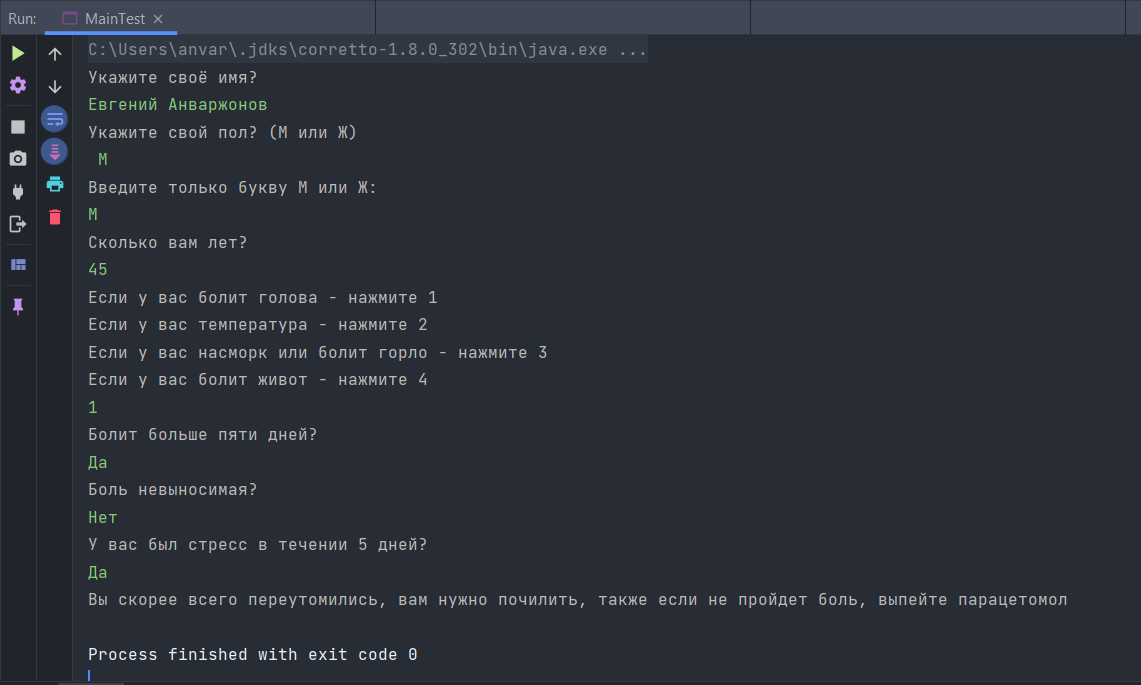
*package* anvarzhonov;  
  
*import* org.junit.jupiter.api.Assertions;  
*import* org.junit.jupiter.api.*BeforeEach*;  
*import* org.junit.jupiter.api.*Test*;  
*import* org.mockito.*Mock*;  
  
*import* java.io.ByteArrayInputStream;  
*import* java.io.InputStream;  
*import* java.util.NoSuchElementException;  
  
*public class* TherapistTest {  
  
 *@Mock* Client client;  
  
 Therapist therapist;  
  
 *@BeforeEach  
 void* setUp(){  
 *this*.therapist = *new* Therapist();  
 therapist.setClient(*new* Client());  
 }  
  
 *@Test  
 public void* setClientNameTest(){  
  
 String input = "Аристотель";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 therapist.setClientName();  
  
 Assertions.assertEquals("Аристотель", therapist.getClient().getName());  
 }  
  
  
 *@Test  
 public void* setClientNameTestNoLineFound(){  
  
 String input = "1234";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 Assertions.assertThrows(NoSuchElementException.*class*, () -> therapist.setClientName());  
 }  
  
 *@Test  
 public void* setClientSexTest(){  
  
 String input = "М";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 therapist.setClientSex();  
  
 Assertions.assertEquals("М", therapist.getClient().getSex());  
 }  
  
 *@Test  
 public void* setClientSexTestNoLineFound(){  
  
 String input = "af";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 Assertions.assertThrows(NoSuchElementException.*class*, () -> therapist.setClientSex());  
 }  
  
 *@Test  
 public void* setClientAgeTest(){  
  
 String input = "15";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 therapist.setClientAge();  
  
 Assertions.assertEquals(15, therapist.getClient().getAge());  
 }  
  
  
 *@Test  
 public void* setClientAgeTestNoLineFound(){  
  
 String input = "0.3";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 Assertions.assertThrows(NoSuchElementException.*class*, () -> therapist.setClientAge());  
  
 }  
  
 *@Test  
 public void* setClientPainTypeTest(){  
  
 String input = "3";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 therapist.setClientPainType();  
  
 Assertions.assertEquals(3, therapist.getClient().getPainType());  
 }  
  
  
 *@Test  
 public void* setClientPainTypeTestNoLineFound(){  
  
 String input = "6";  
 InputStream in = *new* ByteArrayInputStream(input.getBytes());  
 System.setIn(in);  
  
 Assertions.assertThrows(NoSuchElementException.*class*, () -> therapist.setClientPainType());  
  
 }  
}

Результат выполнения:



1. Cделаем проверку на работоспособность нашего приложения:
2. *package* anvarzhonov;  
     
   *public class* MainTest {  
    *public static void* main(String[] args) {  
    Therapist therapist = *new* Therapist();  
    therapist.setClient(*new* Client());  
    therapist.setClientName();  
    therapist.setClientSex();  
    therapist.setClientAge();  
    therapist.setClientPainType();  
    therapist.healing();  
    }  
   }

Результат выполнения:



**Вывод:**

В ходе выполнения практической работы мы научились реализовывать приложение c нуля согласно заданию с использованием методологии разработки через тестирование TDD

Список литературы

1)Алпатов, А. Н. Тестирование и отладка программного обеспечения : методические указания / А. Н. Алпатов. — Москва : РТУ МИРЭА, 2020. — 40 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/167578 (дата обращения: 16.09.2021). — Режим доступа: для авториз. пользователей.

2)Карпович, Е. Е. Методы тестирования и отладки программного обеспечения : учебник / Е. Е. Карпович. — Москва : МИСИС, 2020. — 136 с. — ISBN 978-5-907226-64-7. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/147965 (дата обращения: 16.09.2021). — Режим доступа: для авториз. пользователей.

**Приложение**

Ссылка на репозиторий GitHub: https://github.com/anvarzhonov/pr3