**РТУ МИРЭА**

**КАФЕДРА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ И СТАНДАРТИЗАЦИИ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ**

**УТВЕРЖДЕН**

**XXX.51-ЛУ**

ПРИЛОЖЕНИЕ-ИМИТАЦИЯ ИГРЫ «КАЗИНО»

**Программа и методика испытаний**

**XXX.51**

2020

содержание

[1 Объект испытаний 3](#_Toc55630648)

[1.1 Наименование испытуемой программы 3](#_Toc55630649)

[1.2 Область применения испытуемой программы 3](#_Toc55630650)

[1.3 Обозначение испытуемой программы 3](#_Toc55630651)

[2 Цель испытаний 4](#_Toc55630652)

[3 Требования к программе 5](#_Toc55630653)

[4 Требования к программной документации 6](#_Toc55630654)

[5 Средства и порядок испытаний 7](#_Toc55630655)

[5.1 Технические средства, используемые во время испытаний 7](#_Toc55630656)

[5.2 Программные средства, используемые во время испытаний 7](#_Toc55630657)

[5.3 Порядок проведения испытаний 7](#_Toc55630658)

[6 Условия проведения испытаний 8](#_Toc55630659)

[6.1 Условия начала и завершения отдельных этапов испытаний 8](#_Toc55630660)

[6.2 Условия начала и завершения отдельных этапов испытаний 8](#_Toc55630661)

[6.3 Ограничения в условиях проведения испытаний 8](#_Toc55630662)

[6.4 Требования к техническому обслуживанию 8](#_Toc55630663)

[6.5 Меры, обеспечивающие безопасность и безаварийность проведения испытаний 8](#_Toc55630664)

[6.6 Область применения испытуемой программы 9](#_Toc55630665)

[6.7 Требования к персоналу, проводящему испытания 9](#_Toc55630666)

[7 Методы испытаний 10](#_Toc55630667)

# Объект испытаний

## Наименование испытуемой программы

Наименование испытуемой программы — «Приложение-имитация игры “Казино”».

## Область применения испытуемой программы

Программа предназначена к применению пользователем в развлекательных целях без привлечения реально существующих финансовых средств.

## Обозначение испытуемой программы

Тема разработки — «Разработка приложения-имитации игры “Казино” в рамках совместной групповой работы».

# Цель испытаний

Целью проведения испытаний являются проверка соответствия характеристик разработанной программы функциональным и отдельным иным видам требований, изложенным в документе Техническое задание.

# Требования к программе

При проведении испытаний функциональные характеристики (возможности) программы подлежат проверке на соответствие требованиям, изложенным в пункте «Требования к составу выполняемых функций» Технического задания.

# Требования к программной документации

Состав программной документации, предъявляемой на испытания, должен включать в себя:

1. техническое задание (ГОСТ 19.201-78);
2. матрицу требований;
3. описание бизнес-процессов испытуемой программы (в любой нотации);
4. матрицу соответствия требованиям;
5. руководство программиста на языке Java.

# Средства и порядок испытаний

## Технические средства, используемые во время испытаний

В состав технических средств должен входить персональный компьютер, включающий в себя:

* процессор серии Intel Celeron, AMD Ryzen 3 или новее;
* материнскую плату с FSB;
* оперативную память не менее 64 Мбайт;
* жесткий диск со свободным объёмом не менее 1 Гбайт;
* монитор;
* клавиатуру;
* манипулятор типа «мышь».

## Программные средства, используемые во время испытаний

Системные программные средства, используемые программой, должны быть представлены лицензионной локализованной версией операционной системы.

Допускается использование пакета обновления для ОС Windows v. 20H2.

Для проведения испытаний предоставляется версия разработанной программы в виде .jar пакета.

## Порядок проведения испытаний

Испытания должны проводится поэтапно согласно разделу «Методы испытаний» настоящего документа.

# Условия проведения испытаний

## Условия начала и завершения отдельных этапов испытаний

Испытания должны проводиться в нормальных климатических условиях по ГОСТ 22261-94. Условия проведения испытаний приведены ниже:

* температура окружающего воздуха, °С — 20 ± 5;
* относительная влажность, % — от 30 до 80;
* атмосферное давление, кПа — от 84 до 106;
* частота питающей электросети, Гц — 50 ± 0,5;
* напряжение питающей сети переменного тока, В — 220 ± 4,4.

## Условия начала и завершения отдельных этапов испытаний

Необходимым и достаточным условием завершения 1 этапа испытаний и начала 2 этапа испытаний и т. д. вплоть до последнего пункта раздела «Методы испытаний» настоящего документа, является успешное завершение проверок, проводимых на 1 этапе, далее втором и т. д. до последнего пункта.

## Ограничения в условиях проведения испытаний

Климатические условия эксплуатации, при которых должны обеспечиваться заданные характеристики, должны удовлетворять требованиям, предъявляемым к техническим средствам в части условий их эксплуатации.

## Требования к техническому обслуживанию

Требования к техническому обслуживанию не предъявляются.

## Меры, обеспечивающие безопасность и безаварийность проведения испытаний

При проведении испытаний заказчик должен обеспечить соблюдение требований безопасности, установленных ГОСТ 12.2.007.0–75, ГОСТ 12.2.007.3-75, «Правилами техники безопасности при эксплуатации электроустановок потребителей», и «Правилами технической эксплуатации электроустановок потребителей».

## Область применения испытуемой программы

Заказчик совместно с исполнителем проводят все подготовительные мероприятия для проведения испытаний на объекте заказчика, а также проводят испытания в соответствии с настоящей программой и методиками.

Заказчик осуществляет контроль проведения испытаний.

## Требования к персоналу, проводящему испытания

Персонал, проводящий испытания, должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы, знаниями языка программирования Java, а также способностью понимать английский язык (или иметь средства для перевода).

# Методы испытаний

1. Тестовый сценарий testIsLibraryInit
   1. Дескриптор теста

ADM–1001.

* 1. Описание теста

Тест нацелен проверить работоспособность библиотеки чтения/записи.

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. init() – функция инициализации библиотеки чтения/записи.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Вызываем функцию модуля — init().
     3. Вызываем функцию модуля — free().
     4. Компилируем проект.
     5. Запускаем приложение.
  4. Постусловия
     1. После выполнения п. 1.5.5, убеждаемся в наличии файла profile.prf в корневом каталоге приложения.

1. Тестовый сценарий testIsLibraryFree
   1. Дескриптор теста

ADM–1002.

* 1. Описание теста

Тест нацелен выявить утечку памяти при многократной инициализации и освобождения ресурсов библиотеки чтения/записи файла.

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. free() — функция деинициализации библиотеки чтения/записи.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Создаем цикл на 1000 итераций с наличием функций инициализации библиотеки(init()) и деинициализации(free()).
     3. Компилируем проект.
     4. Запускаем приложение.
  4. Постусловия
     1. На фазе выполнения п. 2.5.4 используя интегрированное или стороннее программное обеспечение, убеждаемся в отсутствии/наличии прироста незначительного объема памяти к зарезервированному приложением.

1. Тестовый сценарий testIsNotEmptyAfterWriting
   1. Дескриптор теста

ADM–1003.

* 1. Описание теста

Тест нацелен проверить наличие данных в файле после записи в него данных при вызове функции free().

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. nWriteDataToFile(ArrayList<Account>) — нативная библиотечная функция, отвечающая за запись данных в файл.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Вызываем функцию модуля — init().
     3. Вызываем функцию модуля — free().
     4. Компилируем проект.
     5. Запускаем приложение.
     6. Завершаем работу приложения.
  4. Постусловия
     1. После выполнения п. 3.5.6 проверяем наличие данных в файле profile.prf.

1. Тестовый сценарий testIsNotEmptyAfterReading
   1. Дескриптор теста

ADM–1004.

* 1. Описание теста

Тест нацелен проверить наличие данных в файле после чтения данных, при вызове функции init().

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. nReadDataFromFile() — нативная библиотечная функция, отвечающая за чтение данных из файла в программу.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Вызываем функцию модуля — init().
     3. Вызываем функцию модуля — free().
     4. Компилируем проект.
     5. Запускаем приложение.
  4. Постусловия
     1. На фазе выполнения п. 4.5.5 проверяем наличие файла и данных в файле profile.prf.

1. Тестовый сценарий testIsLogFileOpen
   1. Дескриптор теста

ADM–1005.

* 1. Описание теста

Тест нацелен проверить работоспособность библиотечной функции ведения лог-файла.

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. option\_nWritingLogFile() — нативная библиотечная функция, отвечающая за ведение лог-файла.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Вызываем функцию модуля — init().
     3. Вызываем функцию модуля — free().
     4. Компилируем проект.
     5. Запускаем приложение.
  4. Постусловия
     1. На фазе выполнения п. 5.5.5 проверяем наличие файла system.log.

1. Тестовый сценарий testIsLogFileWriting
   1. Дескриптор теста

ADM–1006.

* 1. Описание теста

Тест нацелен проверить работоспособность функционала, отвечающего за ведение лог файла.

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. option\_nWritingLogFile() — нативная библиотечная функция, отвечающая за чтение данных из файла в программу.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Вызываем функцию модуля — option\_nWritingLogFile();
     3. Вызываем функцию модуля — init().
     4. Вызываем функцию модуля — free().
     5. Компилируем проект.
     6. Запускаем приложение.
  4. Постусловия
     1. На фазе выполнения п. 6.5.6 проверяем наличие файла и данных в файле system.log.

1. Тестовый сценарий testIsWritingAlways
   1. Дескриптор теста

ADM–1007.

* 1. Описание теста

Тест нацелен проверить работоспособность системы авто-сохранений.

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. option\_saveAlways() — функция модуля, отвечающая за активацию режима автосохранения.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Вызываем функцию модуля — option\_saveAlways().
     3. Вызываем функцию модуля — init().
     4. Вызываем функцию модуля — add(Account).
     5. Вызываем функцию модуля — free().
     6. Компилируем проект.
     7. Запускаем приложение.
     8. Завершаем приложение.
     9. Удаляем вызов функции add(Account) в теле функции main().
     10. Вызываем функцию модуля — get(Account) в теле функции main().
     11. Выводим данные полученного аккаунта в консоль.
     12. Компилируем проект.
     13. Запускаем приложение.
  4. Постусловия
     1. До выполнения фазы тестирования по п. 7 необходимо успешное завершение сценария по п. 3.
     2. После выполнения п. 7.5.13 проверяем наличие данных созданного на фазе 7.5.4 аккаунта пользователя.

1. Тестовый сценарий testIsPrfFileNameChanged
   1. Дескриптор теста

ADM–1008.

* 1. Описание теста

Тест нацелен проверить возможность изменять название выходного файла с данными пользователей.

* 1. Тестируемый модуль
     1. AccountsDataManager — файловый менеджер приложения, чтение/запись данных аккаунтов пользователей в файл.
  2. Тестируемые функции
     1. option\_setNameProfileFile(String) — функция модуля, отвечающая за смену стандартного имени файла с данными пользователей.
  3. Шаги сценария
     1. В функции main() инициализируем модуль AccountsDataManager.
     2. Вызываем функцию модуля — option\_setNameProfileFile(String).
     3. Вызываем функцию модуля — init().
     4. Вызываем функцию модуля — free().
     5. Компилируем проект.
     6. Запускаем приложение.
  4. Постусловия
     1. На фазе выполнения п. 8.5.6 проверяем наличие файла и данных в файле с произвольно установленным на п. 8.5.2 именем файла.

1. Тестовый сценарий testIsAuthSuccess
   1. Дескриптор теста

AU–1001.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод неверных данных аутентификации.

* 1. Тестируемый модуль
     1. PageAuthorization — модуль, отвечающий за проверку корректности введенных пользователем данных авторизации, окна авторизации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageAuthorization, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Вводим различные данные в поля Login и Password.
     3. Жмем кнопку Enter.
  4. Постусловия
     1. На фазе выполнения п. 9.5.3 регистрируем поведение приложения. Удачно — осуществлен вход в систему. Неудачно — сгенерирована ошибка входа.

1. Тестовый сценарий testIsValidRegLogin
   1. Дескриптор теста

REG–1001.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Login окна регистрации.

* 1. Тестируемый модуль
     1. PageRegistration — модуль, отвечающий за проверку корректности введенных пользователем данных, регистрации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageAuthorization, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к регистрации пользователя.
     3. Вводим различные данные в поле Login.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 10.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidRegPassword
   1. Дескриптор теста

REG–1002.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Password окна регистрации.

* 1. Тестируемый модуль
     1. PageRegistration — модуль, отвечающий за проверку корректности введенных пользователем данных, регистрации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageAuthorization, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к регистрации пользователя.
     3. Вводим различные данные в поле Password.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 11.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidRegCardNumber
   1. Дескриптор теста

REG–1003.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Card number окна регистрации.

* 1. Тестируемый модуль
     1. PageRegistration — модуль, отвечающий за проверку корректности введенных пользователем данных, регистрации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageAuthorization, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к регистрации пользователя.
     3. Вводим различные данные в поле Card number.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 12.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidRegCardHolderFullName
   1. Дескриптор теста

REG–1004.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Full name of the card holder окна регистрации.

* 1. Тестируемый модуль
     1. PageRegistration — модуль, отвечающий за проверку корректности введенных пользователем данных, регистрации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageAuthorization, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к регистрации пользователя.
     3. Вводим различные данные в поле Full name of the card holder.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 13.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidRegCardExpDate
   1. Дескриптор теста

REG–1005.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Card expiration date окна регистрации.

* 1. Тестируемый модуль
     1. PageRegistration — модуль, отвечающий за проверку корректности введенных пользователем данных, регистрации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageAuthorization, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к регистрации пользователя.
     3. Вводим различные данные в поле Card expiration date.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 14.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidRegCVV
   1. Дескриптор теста

REG–1006.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле CVV окна регистрации.

* 1. Тестируемый модуль
     1. PageRegistration — модуль, отвечающий за проверку корректности введенных пользователем данных, регистрации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageAuthorization, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к регистрации пользователя.
     3. Вводим различные данные в поле CVV.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 15.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidChangedLogin
   1. Дескриптор теста

CHDT–1001.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Login окна изменения данных пользователя.

* 1. Тестируемый модуль
     1. PageChangeData — модуль, отвечающий за проверку корректности введенных пользователем новых данных информации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageChangeData, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к смене данных пользователя.
     3. Вводим различные данные в поле Login.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 16.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidChangedPassword
   1. Дескриптор теста

CHDT–1002.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Password окна изменения данных пользователя.

* 1. Тестируемый модул:
     1. PageChangeData — модуль, отвечающий за проверку корректности введенных пользователем новых данных информации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageChangeData, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к смене данных пользователя.
     3. Вводим различные данные в поле Password.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 17.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidChangedCardNumber
   1. Дескриптор теста

CHDT–1003.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Card number окна изменения данных пользователя.

* 1. Тестируемый модуль
     1. PageChangeData — модуль, отвечающий за проверку корректности введенных пользователем новых данных информации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageChangeData, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к смене данных пользователя.
     3. Вводим различные данные в поле Card number.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 18.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidChangedFullName
   1. Дескриптор теста

CHDT–1004.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Full name окна изменения данных пользователя.

* 1. Тестируемый модуль
     1. PageChangeData — модуль, отвечающий за проверку корректности введенных пользователем новых данных информации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageChangeData, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к смене данных пользователя.
     3. Вводим различные данные в поле Full name.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 19.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidChangedCardExpDate
   1. Дескриптор теста

CHDT–1005.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле Date expiration card окна изменения данных пользователя.

* 1. Тестируемый модуль
     1. PageChangeData — модуль, отвечающий за проверку корректности введенных пользователем новых данных информации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageChangeData, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к смене данных пользователя.
     3. Вводим различные данные в поле Date expiration card.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 20.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidChangedCVV
   1. Дескриптор теста

CHDT–1006.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле CVV окна изменения данных пользователя.

* 1. Тестируемый модуль
     1. PageChangeData — модуль, отвечающий за проверку корректности введенных пользователем новых данных информации пользователя.
  2. Тестируемые функции
     1. DataValidation() — функция модуля PageChangeData, отвечает за проверку введенных пользователем данных.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к смене данных пользователя.
     3. Вводим различные данные в поле CVV.
     4. Заполняем все оставшиеся поля корректными данными.
     5. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 17.5.5 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной регистрации. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidSlotMachineBet
   1. Дескриптор теста

SMB–1001.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле ставок игры Slot machine, а также на допустимые значения.

* 1. Тестируемый модуль
     1. GameRules — модуль, отвечает за игровую логику.
  2. Тестируемые функции
     1. SlotMachineBetValidation() — функция модуля GameRules, отвечает за проверку ставки.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к игровому режиму Slot machine.
     3. Вводим различные данные в поле ставок.
     4. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 22.5.4 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешной ставке. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidNewBalance
   1. Дескриптор теста

NB–1001.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле пополнения баланса, а также на допустимые значения пополнения.

* 1. Тестируемый модуль
     1. PageChangeData — модуль, отвечающий за проверку корректности введенной суммы пополнения баланса.
  2. Тестируемые функции
     1. NewBalanceValidation() — функция модуля PageChangeData, отвечает за проверку суммы пополнения.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к смене данных пользователя.
     3. Вводим различные данные в поле пополнения баланса.
     4. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 22.5.4 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешном пополнении баланса. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidRussianRouletteBet
   1. Дескриптор теста

RR–1001.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле ставок игрового режима Russian Rullet, а также на допустимые значения.

* 1. Тестируемый модуль
     1. GameRules — модуль, отвечающий за проверку корректности введенной суммы пополнения баланса.
  2. Тестируемые функции
     1. RRBetValidation() — функция модуля GameRules, отвечает за проверку совершенной ставки.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к игровому режиму Russian Roulette.
     3. Вводим различные данные в поле выбора ставки.
     4. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 24.5.4 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешно совершенной ставке. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidDrumRoomBet
   1. Дескриптор теста

DRB–1001.

* 1. Описание теста

Тест нацелен проверить реакцию приложения на ввод различных данных в поле ставок игрового режима Drum room, а также на допустимые значения.

* 1. Тестируемый модуль
     1. GameRules — модуль, отвечающий за проверку корректности введенной суммы пополнения баланса.
  2. Тестируемые функции
     1. DrumRoomBetValidation() — функция модуля GameRules, отвечает за проверку совершенной ставки.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Переходим к игровому режиму Drum Room.
     3. Вводим различные данные в поле выбора ставки.
     4. Жмем кнопку Enter.
  4. Постусловия
     1. После выполнения п. 25.5.4 регистрируем поведение приложения. Удачно — сообщение-уведомление об успешно совершенной ставке. Неудачно — сгенерирована ошибка ввода некорректных данных.

1. Тестовый сценарий testIsValidGUIBuild
   1. Дескриптор теста

GUI–1001.

* 1. Описание теста

Тест нацелен проверить работоспособность всех составляющих пользовательского графического интерфейса.

* 1. Тестируемый модуль
     1. BetRoomController — модуль, отвечающий построение графического пользовательского интерфейса – комната ставок.
     2. Controller — модуль, отвечающий построение графического пользовательского интерфейса — взаимодействие.
     3. DrumController — модуль, отвечающий построение графического пользовательского интерфейса, — взаимодействие (комната режима Drum).
     4. EditingDataController — модуль, отвечающий построение графического пользовательского интерфейса — модификация пользовательских данных.
     5. ErrorNum — модуль, отвечающий построение графического пользовательского интерфейса — вывод ошибок.
     6. MessageWinSlotMacnines — модуль, отвечающий построение графического пользовательского интерфейса — вывод сообщений о победных ставках режима SlotMachine.
     7. MLose — модуль, отвечающий построение графического пользовательского интерфейса — вывод сообщений о поражении.
     8. NotBalance — модуль, отвечающий построение графического пользовательского интерфейса – вывод сообщений об отсутствии баланса.
     9. PaymentDataController — модуль, отвечающий построение графического пользовательского интерфейса – взаимодействие с платежными данными.
     10. PlusBalanceController — модуль, отвечающий построение графического пользовательского интерфейса — пополнение баланса.
     11. RegUserController — модуль, отвечающий построение графического пользовательского интерфейса — регистрация пользователя.
     12. RRRoommControleer — модуль, отвечающий построение графического пользовательского интерфейса — игровой режим русской рулетки.
     13. sampleUserAccountController — модуль, отвечающий построение графического пользовательского интерфейса — пример аккаунта пользователя.
  2. Тестируемые функции
     1. Все функции тестируемых модулей.
  3. Шаги сценария
     1. Запускаем приложение.
     2. Произвольно перемещаемся по различным окнам с произвольными нажатиями по клавишам.
     3. Вводим различные данные в различные поля.
     4. Жмем кнопку Enter.
     5. Завершаем работу приложения.
  4. Постусловия
     1. На стадии выполнения п. 26.5.2 регистрируем поведение приложения. При неудачном завершении теста, зарегистрировать попытку тестирования. И перейти к пункту 26.5.1. Удачно — сбои в работе приложения отсутствуют. Неудачно — сгенерирована ошибка.

Лист регистрации изменений

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Изм. | Номера листов (страниц) | | | | Всего листов (стр.) в документе | № документа | Входящий  № сопроводи-тельного  документа | Под-пись | Дата |
| изме­нен­ных | Заме-нен-ных | новых | Аннулиро-ванных |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |

СОСТАВИЛИ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование организации, предприятия | Должность исполнителя | Фамилия, имя, отчество | Подпись | Дата |
|
| Подгруппа студентов группы ИКБО-17-18 ИИТ РТУ МИРЭА | Разработчик | Евпатов  Михаил Васильевич |  | 05.11. 2020 г. |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

СОГЛАСОВАНО

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование организации, предприятия | Должность исполнителя | Фамилия, имя, отчество | Подпись | Дата |
|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |