**Проект по курсу «Интерпретируемый язык программирования высокого уровня»**

**Основная часть**

**Название проекта:** Редактор персонажей Dungeons and Dragons

**Автор проекта:** Синодов Александр Станиславович, группа БСБО-06-22

**Идея проекта:** Идея проекта заключается в создании вспомогательного приложения для игроков настольно-ролевой игры Dungeons and Dragons. В обычной ситуации пользователю пришлось бы самостоятельно разбираться с основными параметрами, выбором расы, класса и снаряжения. Однако данное приложение должно уменьшить порог вхождения в игру для новых игроков, предоставляя краткое описание для каждого из аспектов.

**Описание реализации:** В программе основным является класс главного окна (class MainWindow), из которого пользователь может либо создать нового персонажа, открыть список существующих, либо просто выйти из программы. Если пользователь решит создать нового персонажа, то будет запущена цепочка классов, последовательно появляющихся на главном окне в виде виджетов. Пользователю придётся последовательно пройти через:

* Выбор расы персонажа (class RaceSelectionWidget)
* Выбор класса (class ClassSelectionWidget)
* Распределение характеристик (class StatSelectionWidget)
* Описание (class CharacterDescriptionWidget)
* Выбор снаряжения (class EquipmentSelectionWidget)

После чего, собрав информацию о выбранных пользователем пунктах, программа сохранит информацию о персонаже в текстовом документе и отправит файл в отдельную папку персонажей. Эта информация будет показана во вкладке «Список персонажей» (class CharacterListWidget) на том же главном окне.

Для создания атмосферы фэнтези были добавлены соответствующие задний фон и фоновая музыка, которые случайным образом меняются при запуске приложения или при окончании произведения соответственно.

**Описание технологий:** Основные функции программы были выполнены с использованием библиотеки PyQt5. В основном она использовалась для создания пользовательского интерфейса. Для сохранения персонажей и работы с файлами использовалась библиотека OS. Для добавления фоновой музыки немного использовалась библиотека pygame.

**Приложение**

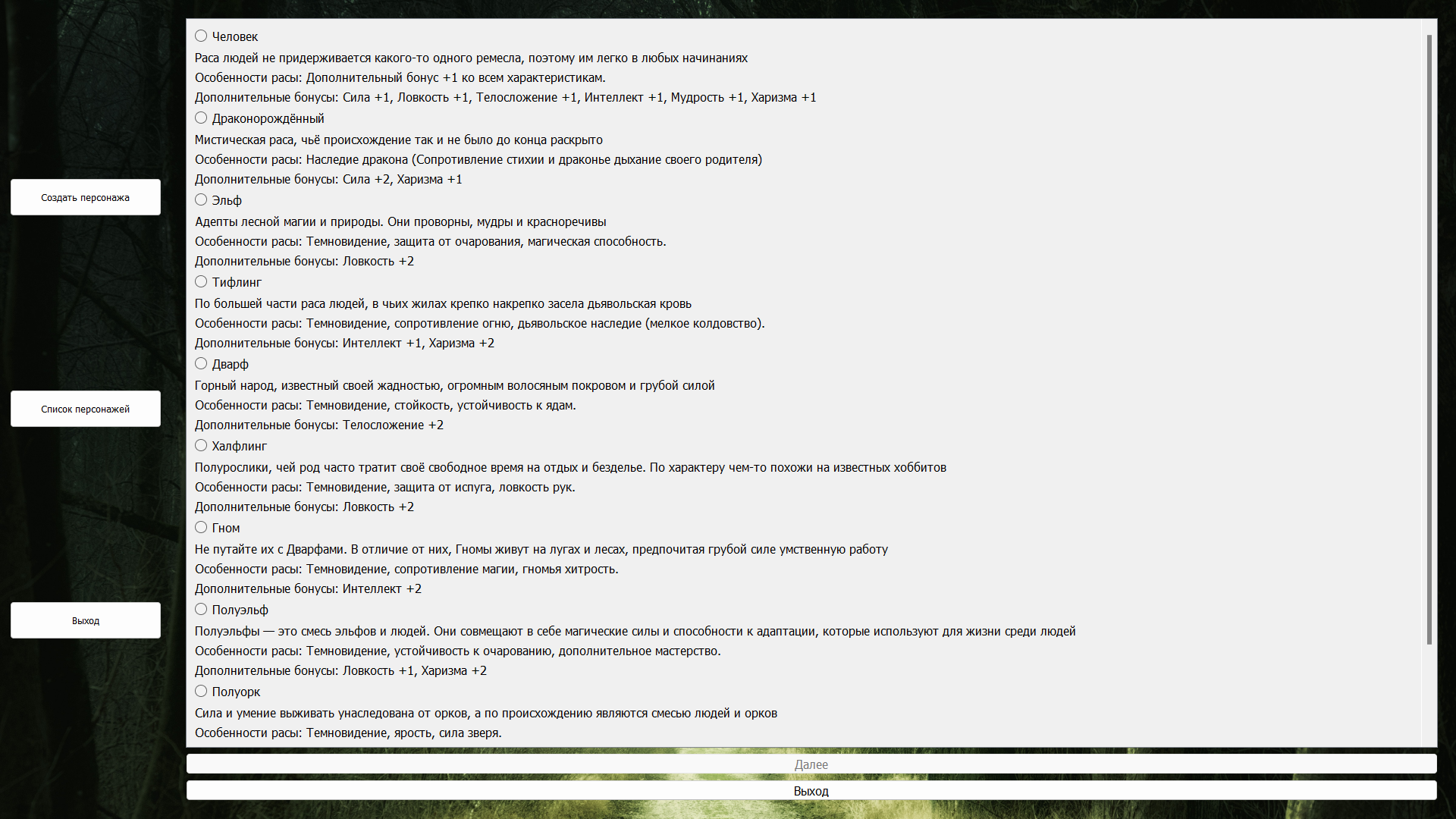


**Рисунок 1. Главное окно программы**

На данном изображении показано главное окно программы. Здесь пользователь может выбрать одно из следующих действий: создать нового персонажа, открыть список существующих персонажей или выйти из программы.

**Рисунок 2. Виджет просмотра персонажей**

Данное изображение показывает вызов виджета, отвечающего за показ списка персонажей, собранных в отдельной папке, их статистики и возможность удалить ненужных персонажей пользователя.



**Рисунок 3. Начало создания персонажа**

Последнее изображение показывает начало работы с персонажем пользователя, а именно – выбор расы. Последующие стадии создания представлены подобным образом.

**Листинг кода**

*import* math  
*import* sys  
*import* os  
*import* random  
*import* pygame  
*from* PyQt5.QtWidgets *import* (QApplication, QWidget, QVBoxLayout, QPushButton, QLabel, QScrollArea, QDialog,  
 QHBoxLayout, QRadioButton, QLineEdit, QTextEdit, QComboBox, QCheckBox,  
 QListWidget, QGroupBox, QFormLayout, QMessageBox)  
*from* PyQt5.QtGui *import* QPixmap, QPalette, QBrush, QIcon  
*from* PyQt5.QtCore *import* Qt, QTimer  
  
*# Главное окно приложения  
class* MainWindow(QWidget):  
 *def \_\_init\_\_*(*self*, image\_folder, icon\_path, music\_folder):  
 *super*().*\_\_init\_\_*()  
 *self*.image\_folder = image\_folder  
 *self*.icon\_path = icon\_path  
 *self*.music\_folder = music\_folder  
  
 *self*.current\_background = *None  
 self*.is\_fullscreen = *True  
 self*.is\_creating\_character = *False  
 self*.is\_viewing\_characters = *False  
  
 self*.init\_ui()  
 *self*.init\_music()  
  
 *def* init\_ui(*self*):  
 *self*.setWindowIcon(QIcon(*self*.icon\_path))  
 *self*.set\_random\_background()  
 *self*.create\_buttons()  
 *self*.showFullScreen()  
 *self*.setWindowTitle("D&D character creator")  
  
 *def* create\_buttons(*self*):  
 *self*.create\_character\_button = *self*.create\_button("Создать персонажа", *self*.show\_race\_selection)  
 *self*.view\_characters\_button = *self*.create\_button("Список персонажей", *self*.show\_character\_list)  
 *self*.exit\_button = *self*.create\_button("Выход", *self*.close)  
  
 button\_layout = QVBoxLayout()  
 button\_layout.addWidget(*self*.create\_character\_button)  
 button\_layout.addWidget(*self*.view\_characters\_button)  
 button\_layout.addWidget(*self*.exit\_button)  
  
 main\_layout = QHBoxLayout()  
 main\_layout.addLayout(button\_layout)  
  
 *self*.race\_selection\_widget = QWidget()  
 *self*.class\_selection\_widget = QWidget()  
  
 main\_layout.addWidget(*self*.race\_selection\_widget)  
 main\_layout.addWidget(*self*.class\_selection\_widget)  
  
 *self*.setLayout(main\_layout)  
  
 *def* create\_button(*self*, text, callback):  
 button = QPushButton(text, *self*)  
 button.setFixedSize(200, 50)  
 button.clicked.connect(callback)  
 *return* button  
  
 *def* set\_random\_background(*self*):  
 images = [f *for* f *in* os.listdir(*self*.image\_folder) *if* os.path.isfile(os.path.join(*self*.image\_folder, f))]  
 *if not* images:  
 *raise Exception*("No images found in the specified directory.")  
 random\_image\_path = os.path.join(*self*.image\_folder, random.choice(images))  
 *self*.current\_background = QPixmap(random\_image\_path)  
 *self*.update\_background()  
  
 *def* update\_background(*self*):  
 *if self*.current\_background:  
 scaled\_pixmap = *self*.current\_background.scaled(*self*.size(), Qt.KeepAspectRatioByExpanding, Qt.SmoothTransformation)  
 palette = QPalette()  
 palette.setBrush(QPalette.Background, QBrush(scaled\_pixmap))  
 *self*.setPalette(palette)  
  
 *def* show\_race\_selection(*self*):  
 *if not self*.is\_creating\_character *and not self*.is\_viewing\_characters:  
 *self*.is\_creating\_character = *True  
 self*.is\_viewing\_characters = *False  
 self*.hide\_widget\_if\_exists("character\_list\_widget")  
 *self*.race\_selection\_widget = RaceSelectionWidget(*self*)  
 *self*.layout().addWidget(*self*.race\_selection\_widget)  
 *self*.race\_selection\_widget.show()  
  
 *def* show\_character\_list(*self*):  
 *if not self*.is\_viewing\_characters *and not self*.is\_creating\_character:  
 *self*.is\_viewing\_characters = *True  
 self*.is\_creating\_character = *False  
 self*.hide\_widget\_if\_exists("race\_selection\_widget")  
 *self*.character\_list\_widget = CharacterListWidget(*self*)  
 *self*.layout().addWidget(*self*.character\_list\_widget)  
 *self*.character\_list\_widget.show()  
  
 *def* hide\_widget\_if\_exists(*self*, widget\_name):  
 *if hasattr*(*self*, widget\_name):  
 *getattr*(*self*, widget\_name).hide()  
  
 *def* show\_equipment\_selection(*self*, description\_window):  
 *self*.equipment\_selection\_widget = EquipmentSelectionWidget(description\_window)  
 *self*.layout().addWidget(*self*.equipment\_selection\_widget)  
 *self*.equipment\_selection\_widget.show()  
  
 *def* show\_stat\_selection(*self*, class\_selection\_widget, selected\_race, race\_bonuses):  
 *self*.stat\_selection\_widget = StatSelectionWidget(*self*, class\_selection\_widget, selected\_race, race\_bonuses)  
 *self*.layout().addWidget(*self*.stat\_selection\_widget)  
 *self*.stat\_selection\_widget.show()  
  
 *def* show\_character\_description(*self*, selected\_race, stat\_selection\_widget):  
 *self*.character\_description\_widget = CharacterDescriptionWidget(*self*, selected\_race, stat\_selection\_widget)  
 *self*.layout().addWidget(*self*.character\_description\_widget)  
 *self*.character\_description\_widget.show()  
  
 *def* keyPressEvent(*self*, event):  
 *if* event.key() == Qt.Key\_F10:  
 *if self*.is\_fullscreen:  
 *self*.showNormal()  
 *self*.is\_fullscreen = *False  
 else*:  
 *self*.showFullScreen()  
 *self*.is\_fullscreen = *True  
  
 def* resizeEvent(*self*, event):  
 *self*.update\_background()  
 *super*().resizeEvent(event)  
  
 *def* init\_music(*self*):  
 pygame.display.init()  
 pygame.mixer.init()  
  
 *self*.music\_files = [f *for* f *in* os.listdir(*self*.music\_folder)*if* os.path.isfile(os.path.join(*self*.music\_folder, f))]  
 *if not self*.music\_files:  
 *raise Exception*("No music files found in the specified directory.")  
 *self*.play\_random\_music()  
  
 *def* play\_random\_music(*self*):  
 random\_music\_path = os.path.join(  
 *self*.music\_folder, random.choice(*self*.music\_files)  
 )  
 pygame.mixer.music.load(random\_music\_path)  
 pygame.mixer.music.play()  
 pygame.mixer.music.set\_volume(0.1)  
 pygame.mixer.music.set\_endevent(pygame.USEREVENT)  
 *self*.music\_timer = QTimer(*self*)  
 *self*.music\_timer.timeout.connect(*self*.check\_music\_end)  
 *self*.music\_timer.start(1000)  
  
 *def* check\_music\_end(*self*):  
 *for* event *in* pygame.event.get():  
 *if* event.type == pygame.USEREVENT:  
 *self*.play\_random\_music()  
  
  
*# Окно выбора расы  
class* RaceSelectionWidget(QWidget):  
 *def \_\_init\_\_*(*self*, parent):  
 *super*().*\_\_init\_\_*(parent)  
 *self*.parent = parent  
 *self*.selected\_race = *None  
 self*.race\_bonuses = {}  
 *self*.races = {  
 "Человек": {  
 "description": "Раса людей не придерживается какого-то одного ремесла, поэтому им легко в любых начинаниях",  
 "features": "Дополнительный бонус +1 ко всем характеристикам.",  
 "bonuses": {"Сила": 1,"Ловкость": 1,"Телосложение": 1,"Интеллект": 1,"Мудрость": 1,"Харизма": 1,},  
 },  
 "Драконорождённый": {  
 "description": "Мистическая раса, чьё происхождение так и не было до конца раскрыто",  
 "features": "Наследие дракона (Сопротивление стихии и драконье дыхание своего родителя)",  
 "bonuses": {"Сила": 2, "Харизма": 1},  
 },  
 "Эльф": {  
 "description": "Адепты лесной магии и природы. Они проворны, мудры и красноречивы",  
 "features": "Темновидение, защита от очарования, магическая способность.",  
 "bonuses": {"Ловкость": 2},  
 },  
 "Тифлинг": {  
 "description": "По большей части раса людей, в чьих жилах крепко накрепко засела дьявольская кровь",  
 "features": "Темновидение, сопротивление огню, дьявольское наследие (мелкое колдовство).",  
 "bonuses": {"Интеллект": 1, "Харизма": 2},  
 },  
 "Дварф": {  
 "description": "Горный народ, известный своей жадностью, огромным волосяным покровом и грубой силой",  
 "features": "Темновидение, стойкость, устойчивость к ядам.",  
 "bonuses": {"Телосложение": 2},  
 },  
 "Халфлинг": {  
 "description": "Полурослики, чей род часто тратит своё свободное время на отдых и безделье. По характеру чем-то похожи на известных хоббитов",  
 "features": "Темновидение, защита от испуга, ловкость рук.",  
 "bonuses": {"Ловкость": 2},  
 },  
 "Гном": {  
 "description": "Не путайте их с Дварфами. В отличие от них, Гномы живут на лугах и лесах, предпочитая грубой силе умственную работу",  
 "features": "Темновидение, сопротивление магии, гномья хитрость.",  
 "bonuses": {"Интеллект": 2},  
 },  
 "Полуэльф": {  
 "description": "Полуэльфы — это смесь эльфов и людей. Они совмещают в себе магические силы и способности к адаптации, которые используют для жизни среди людей",  
 "features": "Темновидение, устойчивость к очарованию, дополнительное мастерство.",  
 "bonuses": {"Ловкость": 1, "Харизма": 2},  
 },  
 "Полуорк": {  
 "description": "Сила и умение выживать унаследована от орков, а по происхождению являются смесью людей и орков",  
 "features": "Темновидение, ярость, сила зверя.",  
 "bonuses": {"Сила": 2, "Телосложение": 1},  
 },  
 "Тёмный эльф": {  
 "description": "Эльфы, чьё происхождение отличается от лесных. Их далекие предки были тесно связаны с силами зла",  
 "features": "Темновидение, устойчивость к огню, магическая способность.",  
 "bonuses": {"Ловкость": 1, "Интеллект": 1, "Харизма": 1},  
 },  
 }  
 *self*.init\_ui()  
  
 *def* init\_ui(*self*):  
 *self*.scroll\_area = QScrollArea(*self*)  
 *self*.scroll\_area.setWidgetResizable(*True*)  
 *self*.widget = QWidget()  
 *self*.layout = QVBoxLayout()  
 *self*.display\_races()  
  
 *self*.next\_button = QPushButton("Далее", *self*)  
 *self*.exit\_button = QPushButton("Выход", *self*)  
 *self*.next\_button.setEnabled(*False*)  
 *self*.next\_button.clicked.connect(*self*.proceed\_to\_next\_step)  
 *self*.exit\_button.clicked.connect(*self*.close)  
  
 main\_layout = QVBoxLayout()  
 main\_layout.addWidget(*self*.scroll\_area)  
 main\_layout.addWidget(*self*.next\_button)  
 main\_layout.addWidget(*self*.exit\_button)  
 *self*.setStyleSheet("background-style:white;")  
 *self*.setLayout(main\_layout)  
 QMessageBox.information(*self*, "Раса", "Выберите расу персонажа")  
  
 *def* update\_scroll\_area(*self*):  
 *self*.widget.setLayout(*self*.layout)  
 *self*.scroll\_area.setWidget(*self*.widget)  
  
 *def* select\_race(*self*, race\_name, bonuses):  
 *self*.selected\_race = race\_name  
 *self*.race\_bonuses = bonuses  
 *self*.next\_button.setEnabled(*True*)  
  
 *def* close(*self*):  
 *self*.parent.is\_creating\_character = *False  
 super*().close()  
  
 *def* display\_races(*self*):  
 *for* race\_name, race\_info *in self*.races.items():  
 *self*.create\_race\_option(race\_name, race\_info)  
 *self*.setStyleSheet("font-size: 16px;")  
 *self*.update\_scroll\_area()  
  
 *def* create\_race\_option(*self*, race\_name, race\_info):  
 race\_radio\_button = QRadioButton(race\_name)  
 race\_radio\_button.clicked.connect(  
 *lambda* \_, rn=race\_name, rb=race\_info["bonuses"]: *self*.select\_race(rn, rb)  
 )  
 *self*.layout.addWidget(race\_radio\_button)  
  
 description\_label = QLabel(race\_info["description"])  
 features\_label = QLabel("Особенности расы: " + race\_info["features"])  
 bonuses\_label = QLabel(  
 "Дополнительные бонусы: "  
 + ", ".join(  
 f"{key} +{value}" *for* key, value *in* race\_info["bonuses"].items()  
 )  
 )  
  
 *self*.layout.addWidget(description\_label)  
 *self*.layout.addWidget(features\_label)  
 *self*.layout.addWidget(bonuses\_label)  
  
 *def* proceed\_to\_next\_step(*self*):  
 *self*.class\_selection\_widget = ClassSelectionWidget(  
 *self*.parent, *self*.selected\_race, *self*.race\_bonuses  
 )  
 *self*.class\_selection\_widget.show()  
 *self*.parent.layout().addWidget(*self*.class\_selection\_widget)  
 *self*.hide()  
  
  
*# Окно выбора класса  
class* ClassSelectionWidget(QWidget):  
 *def \_\_init\_\_*(*self*, parent, selected\_race, race\_bonuses):  
 *super*().*\_\_init\_\_*(parent)  
 *self*.parent = parent  
 *self*.selected\_class = *None  
 self*.selected\_race = selected\_race  
 *self*.race\_bonuses = race\_bonuses  
 *self*.init\_ui()  
  
 *def* init\_ui(*self*):  
 *self*.scroll\_area = QScrollArea(*self*)  
 *self*.scroll\_area.setWidgetResizable(*True*)  
 *self*.widget = QWidget()  
 *self*.layout = QVBoxLayout()  
 *self*.display\_classes()  
  
 *self*.next\_button = QPushButton("Далее", *self*)  
 *self*.back\_button = QPushButton("Назад", *self*)  
 *self*.next\_button.setEnabled(*False*)  
 *self*.next\_button.clicked.connect(*self*.proceed\_to\_next\_step)  
 *self*.back\_button.clicked.connect(*self*.return\_to\_previous\_step)  
  
 main\_layout = QVBoxLayout()  
 main\_layout.addWidget(*self*.scroll\_area)  
 main\_layout.addWidget(*self*.next\_button)  
 main\_layout.addWidget(*self*.back\_button)  
 *self*.setStyleSheet("background-style:white;")  
 *self*.setLayout(main\_layout)  
 QMessageBox.information(*self*, "Класс", "Выберите класс персонажа")  
  
 *def* update\_scroll\_area(*self*):  
 *self*.widget.setLayout(*self*.layout)  
 *self*.scroll\_area.setWidget(*self*.widget)  
  
 *def* select\_class(*self*, class\_name):  
 *self*.selected\_class = class\_name  
 *self*.next\_button.setEnabled(*True*)  
  
 *def* display\_classes(*self*):  
 classes = {  
 "Бард": "Неважно, кем является бард: учёным, скальдом или проходимцем; он плетёт магию из слов и музыки, вдохновляя союзников, деморализуя противников, манипулируя сознанием, создавая иллюзии, и даже исцеляя раны.",  
 "Варвар": "Несмотря на разнообразие, всех варваров объединяет одно — их ярость. Необузданный, неугасимый и бездумный гнев. Не просто эмоция, их ярость как свирепость загнанного в угол хищника, как безжалостный удар урагана, как штормовые валы океана.",  
 "Воин": "Странствующие рыцари, военачальники-завоеватели, королевские чемпионы, элитная пехота, бронированные наёмники и короли разбоя — будучи воинами, все они мастерски владеют оружием, доспехами, и приёмами ведения боя.",  
 "Волшебник": "Волшебники — адепты высшей магии, объединяющиеся по типу своих заклинаний. Опираясь на тонкие плетения магии, пронизывающей вселенную, волшебники способны создавать заклинания взрывного огня, искрящихся молний, тонкого обмана и грубого контроля над сознанием.",  
 "Друид": "Призывая стихии или подражая животным, друиды воплощают незыблемость, приспособляемость и гнев природы. Они ни в коем случае не владыки природы — вместо этого друиды ощущают себя частью её неодолимой воли.",  
 "Жрец": "Жрецы являются посредниками между миром смертных и далёкими мирами богов. Настолько же разные, насколько боги, которым они служат, жрецы воплощают работу своих божеств.",  
 "Изобретатель": "Изобретатели — величайшие мастера пробуждать магию в обычных предметах. Они рассматривают магию как сложную систему, которую следует расшифровать и применять в заклинаниях и изобретениях.",  
 "Колдун": "Колдуны — искатели знаний, что скрываются в ткани мультивселенной. Через договор, заключённый с таинственными существами сверхъестественной силы, колдуны открывают для себя магические эффекты, как едва уловимые, так и впечатляющие воображение.",  
 "Монах": "Вне зависимости от выбранной дисциплины, всех монахов объединяет одно — возможность управлять энергией, текущей в их телах. Вне зависимости от того, проявляется ли она выдающимися боевыми способностями, или чуть заметным усилением защиты и скорости, эта энергия влияет на всё, что делает монах.",  
 "Паладин": "Вне зависимости от происхождения и миссии, паладинов объединяет их клятва противостоять силам зла. Принесённая ли перед алтарём бога и заверенная священником, или же на священной поляне перед духами природы и феями, или в момент отчаяния и горя смерти, присяга паладина — могущественный договор.",  
 "Плут": "Плуты полагаются на мастерство, скрытность и уязвимые места врагов, чтобы взять верх в любой ситуации. У них достаточно сноровки для нахождения решения в любой ситуации, демонстрируя находчивость и гибкость, которые являются краеугольным камнем любой успешной группы искателей приключений.",  
 "Следопыт": "Вдали от суеты городов и посёлков, за изгородями, которые защищают самые далёкие фермы от ужасов дикой природы, среди плотно стоящих деревьев, беспутья лесов и на просторах необъятных равнин следопыты несут свой бесконечный дозор.",  
 "Чародей": "Чародеи являются носителями магии, дарованной им при рождении их экзотической родословной, неким потусторонним влиянием или воздействием неизвестных вселенских сил. Никто не может обучиться чародейству, как, например, выучить язык, так же как никто не может обучить, как прожить легендарную жизнь.",  
 }  
 *for* class\_name, class\_description *in* classes.items():  
 *self*.create\_class\_option(class\_name, class\_description)  
 *self*.setStyleSheet("font-size: 16px;")  
 *self*.update\_scroll\_area()  
  
 *def* create\_class\_option(*self*, class\_name, class\_description):  
 class\_radio\_button = QRadioButton(class\_name)  
 class\_radio\_button.clicked.connect(  
 *lambda* \_, cn=class\_name: *self*.select\_class(cn)  
 )  
 *self*.layout.addWidget(class\_radio\_button)  
 *self*.layout.addWidget(QLabel(class\_description))  
  
 *def* proceed\_to\_next\_step(*self*):  
 *self*.parent.show\_stat\_selection(*self*, *self*.selected\_race, *self*.race\_bonuses)  
 *self*.hide()  
  
 *def* return\_to\_previous\_step(*self*):  
 *self*.parent.race\_selection\_widget.show()  
 *self*.hide()  
  
  
*# Окно распределения характеристик  
class* StatSelectionWidget(QWidget):  
 *def \_\_init\_\_*(*self*, parent, class\_selection\_widget, selected\_race, race\_bonuses):  
 *super*().*\_\_init\_\_*(parent)  
 *self*.parent = parent  
 *self*.class\_selection\_widget = class\_selection\_widget  
 *self*.selected\_race = selected\_race  
 *self*.stats = {  
 "Сила": 0,  
 "Ловкость": 0,  
 "Телосложение": 0,  
 "Интеллект": 0,  
 "Мудрость": 0,  
 "Харизма": 0,  
 }  
 *self*.selected\_values = {}  
 *self*.race\_bonuses = race\_bonuses  
 *self*.init\_ui()  
  
 *def* init\_ui(*self*):  
 *self*.layout = QVBoxLayout()  
 *self*.stat\_comboboxes = {}  
 *self*.final\_stat\_labels = {}  
  
 inner\_widget = QWidget()  
 inner\_layout = QVBoxLayout(inner\_widget)  
 inner\_widget.setLayout(inner\_layout)  
 inner\_widget.setStyleSheet("background-color: white;")  
  
 *for* stat, bonus *in self*.race\_bonuses.items():  
 *self*.stats[stat] += bonus  
  
 stat\_distribution\_label = QLabel(  
 "Стандартное распределение характеристик: 15, 14, 13, 12, 10, 8")  
 stat\_distribution\_label\_2=QLabel("Совет: Чем выше модификатор(значение в скобках), тем лучше вы будете справляться с проверками характеристик")  
 inner\_layout.addWidget(stat\_distribution\_label)  
 inner\_layout.addWidget(stat\_distribution\_label\_2)  
  
 *for* stat *in self*.stats.keys():  
 h\_layout = QHBoxLayout()  
 stat\_label = QLabel(  
 f"{stat} (бонус от расы: +{*self*.race\_bonuses.get(stat, 0)})"  
 )  
 stat\_combobox = QComboBox()  
 stat\_combobox.addItems(["15", "14", "13", "12", "10", "8", "0"])  
 stat\_combobox.setCurrentIndex(6)  
 stat\_combobox.currentIndexChanged.connect(  
 *lambda* \_, s=stat, cb=stat\_combobox: *self*.update\_stat(  
 s, cb.currentText()  
 )  
 )  
 h\_layout.addWidget(stat\_label)  
 h\_layout.addWidget(stat\_combobox)  
 *self*.stat\_comboboxes[stat] = stat\_combobox  
  
 final\_stat\_label = QLabel(  
 f"Итоговое значение и модификатор: {*self*.stats[stat]} ({math.floor((*self*.stats[stat] - 10) / 2)})"  
 )  
 *self*.final\_stat\_labels[stat] = final\_stat\_label  
 h\_layout.addWidget(final\_stat\_label)  
  
 inner\_layout.addLayout(h\_layout)  
  
 *self*.next\_button = QPushButton("Далее", *self*)  
 *self*.back\_button = QPushButton("Назад", *self*)  
 *self*.next\_button.setEnabled(*False*)  
 *self*.next\_button.clicked.connect(*self*.proceed\_to\_next\_step)  
 *self*.back\_button.clicked.connect(*self*.return\_to\_previous\_step)  
  
 inner\_layout.addWidget(*self*.next\_button)  
 inner\_layout.addWidget(*self*.back\_button)  
  
 *self*.scroll\_area = QScrollArea(*self*)  
 *self*.scroll\_area.setWidgetResizable(*True*)  
 *self*.scroll\_area.setWidget(inner\_widget)  
  
 *self*.layout.addWidget(*self*.scroll\_area)  
 *self*.setStyleSheet("font-size: 16px;")  
 *self*.setLayout(*self*.layout)  
 QMessageBox.information(  
 *self*, "Характеристики", "Распределите характеристики персонажа"  
 )  
  
 *def* update\_stat(*self*, stat, value):  
 *if* value *and* value != "0":  
 *self*.selected\_values[stat] = value  
 *else*:  
 *self*.selected\_values.pop(stat, *None*)  
  
 *self*.stats[stat] = (  
 *int*(value) + *self*.race\_bonuses.get(stat, 0) *if* value != "0" *else* 0  
 )  
 modifier = math.floor((*self*.stats[stat] - 10) / 2)  
 *self*.final\_stat\_labels[stat].setText(  
 f"Итоговое значение и модификатор: {*self*.stats[stat]} ({modifier})"  
 )  
 *self*.update\_comboboxes()  
  
 *self*.next\_button.setEnabled(*len*(*self*.selected\_values) == *len*(*self*.stats))  
  
 *def* update\_comboboxes(*self*):  
 all\_values = {"0", "15", "14", "13", "12", "10", "8"}  
 used\_values = *set*(*self*.selected\_values.values())  
  
 *for* stat, combobox *in self*.stat\_comboboxes.items():  
 current\_value = combobox.currentText()  
  
 combobox.blockSignals(*True*)  
 combobox.clear()  
 *for* value *in sorted*(all\_values, key=*int*, reverse=*True*):  
 *if* value == current\_value *or* value *not in* used\_values:  
 combobox.addItem(value)  
  
 combobox.setCurrentText(  
 current\_value *if* current\_value *in* all\_values *else* "0"  
 )  
 combobox.blockSignals(*False*)  
  
 *def* proceed\_to\_next\_step(*self*):  
 *self*.parent.race\_bonuses = *self*.race\_bonuses  
 *self*.parent.show\_character\_description(*self*.selected\_race, *self*)  
 *self*.hide()  
  
 *def* return\_to\_previous\_step(*self*):  
 *self*.class\_selection\_widget.show()  
 *self*.hide()  
  
  
*# Окно описания персонажа  
class* CharacterDescriptionWidget(QDialog):  
 *def \_\_init\_\_*(*self*, parent, selected\_race, stat\_selection\_widget):  
 *super*().*\_\_init\_\_*(parent)  
 *self*.setWindowFlags(Qt.FramelessWindowHint)  
 *self*.setGeometry(276, 76, 1555, 922)  
 *self*.character\_description = ""  
 *self*.character\_name = ""  
 *self*.selected\_race = selected\_race  
 *self*.stat\_selection\_widget = stat\_selection\_widget  
 *self*.race\_bonuses = parent.race\_bonuses  
 *self*.class\_selection\_widget = (  
 parent.stat\_selection\_widget.class\_selection\_widget  
 )  
 *self*.init\_ui()  
  
 *def* init\_ui(*self*):  
 layout = QVBoxLayout()  
  
 name\_label = QLabel("Введите имя персонажа:")  
 name\_label.setStyleSheet("background-color: white; font-size: 16px;")  
 layout.addWidget(name\_label)  
  
 *self*.name\_edit = QLineEdit()  
 *self*.name\_edit.textChanged.connect(*self*.check\_input)  
 layout.addWidget(*self*.name\_edit)  
  
 description\_label = QLabel("Описание персонажа:")  
 description\_label.setStyleSheet("background-color: white; font-size: 16px;")  
 layout.addWidget(description\_label)  
  
 *self*.description\_edit = QTextEdit()  
 layout.addWidget(*self*.description\_edit)  
  
 *self*.next\_button = QPushButton("Далее", *self*)  
 *self*.next\_button.setEnabled(*False*)  
 *self*.back\_button = QPushButton("Назад", *self*)  
 *self*.next\_button.clicked.connect(*self*.proceed\_to\_next\_step)  
 *self*.back\_button.clicked.connect(*self*.return\_to\_previous\_step)  
  
 layout.addWidget(*self*.next\_button)  
 layout.addWidget(*self*.back\_button)  
 *self*.setStyleSheet("background-style:white;")  
 *self*.setStyleSheet("font-size: 16px;")  
 *self*.setLayout(layout)  
 QMessageBox.information(*self*, "Описание", "Введите Имя и описание персонажа")  
  
 *def* check\_input(*self*):  
 *self*.next\_button.setEnabled(*bool*(*self*.name\_edit.text().strip()))  
  
 *def* proceed\_to\_next\_step(*self*):  
 *self*.character\_name = *self*.name\_edit.text().strip()  
 *self*.character\_description = *self*.description\_edit.toPlainText().strip()  
 *if not self*.character\_name:  
 QMessageBox.warning(*self*, "Ошибка", "Имя персонажа не может быть пустым.")  
 *return  
 self*.parent().show\_equipment\_selection(*self*)  
 *self*.hide()  
  
 *def* return\_to\_previous\_step(*self*):  
 *if self*.stat\_selection\_widget:  
 *self*.stat\_selection\_widget.show()  
 *self*.parent().is\_creating\_character = *False  
 self*.close()  
  
  
*# Окно выбора снаряжения  
class* EquipmentSelectionWidget(QDialog):  
 *def \_\_init\_\_*(*self*, description\_window):  
 *super*().*\_\_init\_\_*(description\_window)  
 *self*.setWindowTitle("Выбор снаряжения")  
 *self*.setWindowFlags(Qt.FramelessWindowHint)  
 *self*.setGeometry(276, 76, 1555, 922)  
 *self*.description\_window = description\_window  
 *self*.selected\_equipment = {  
 "оружие": *None*,  
 "снаряжение": *None*,  
 "инструменты": *None*,  
 "снаряжение класса": *None*,  
 }  
 *self*.init\_ui()  
  
 *def* init\_ui(*self*):  
 layout = QVBoxLayout()  
 class\_name = *self*.description\_window.class\_selection\_widget.selected\_class  
  
 equipment = {  
 "Бард": {  
 "оружие": ["Рапира", "Длинный меч", "Кинжал"],  
 "снаряжение": ["Кожаная броня", "Мантия"],  
 "инструменты": ["Набор артиста", "Набор дипломата", "Набор книг"],  
 "снаряжение класса": ["Лютня", "Флейта", "Скрипка"],  
 },  
 "Варвар": {  
 "оружие": ["Секира", "Молот", "Два ручных топора"],  
 "снаряжение": ["Кожаная броня", "Кольчуга", "Обмотки"],  
 "инструменты": ["Набор путешественника"],  
 "снаряжение класса": ["Боевой рог"],  
 },  
 "Воин": {  
 "оружие": ["Двуручный меч", "Длинный меч", "Длинный лук"],  
 "снаряжение": ["Кожаная броня", "Кольчуга"],  
 "инструменты": [  
 "Набор путешественника",  
 "Набор исследователя подземелий",  
 ],  
 "снаряжение класса": [],  
 },  
 "Волшебник": {  
 "оружие": ["Посох", "Кинжал"],  
 "снаряжение": ["Мантия"],  
 "инструменты": ["Набор учёного", "Мешочек с компонентами"],  
 "снаряжение класса": ["Книга заклинаний"],  
 },  
 "Друид": {  
 "оружие": ["Боевой", "Скимитар"],  
 "снаряжение": ["Кожаная броня", "Тканный доспех"],  
 "инструменты": ["Набор путешественника"],  
 "снаряжение класса": ["Посох друида"],  
 },  
 "Жрец": {  
 "оружие": ["Булава", "Боевой молот"],  
 "снаряжение": ["Чешуйчатый доспех", "Кольчуга", "Кожаный доспех"],  
 "инструменты": ["Набор путешественника", "Набор священника"],  
 "снаряжение класса": ["Священный символ и щит церкви"],  
 },  
 "Изобретатель": {  
 "оружие": ["Меч", "Лёгкий арбалет"],  
 "снаряжение": ["Поклёпанная броня", "Чешуйчатый доспех"],  
 "инструменты": [  
 "Воровские инструменты",  
 "Набор исследователя подземелий",  
 ],  
 "снаряжение класса": ["Однозарядный пистолет"],  
 },  
 "Колдун": {  
 "оружие": ["Короткий меч", "Лёгкий арбалет"],  
 "снаряжение": ["Кожаная броня"],  
 "инструменты": [  
 "Набор учёного",  
 "Набор исследователя подземелий",  
 "Мешочек с компонентами",  
 ],  
 "снаряжение класса": ["Гримуар"],  
 },  
 "Монах": {  
 "оружие": ["Боевой посохч", "Короткий меч"],  
 "снаряжение": ["Обмотки"],  
 "инструменты": [  
 "Набор путешественника",  
 "Набор исследователя подземелий",  
 ],  
 "снаряжение класса": ["Чётки для ци"],  
 },  
 "Паладин": {  
 "оружие": ["Двуручный меч", "Боевой молот"],  
 "снаряжение": ["Кольчуга", "Латные доспехи"],  
 "инструменты": ["Набор путешественника", "Набор священника"],  
 "снаряжение класса": ["Священный символ"],  
 },  
 "Плут": {  
 "оружие": ["Рапира", "Короткий меч"],  
 "снаряжение": ["Кожаный доспех с капюшоном"],  
 "инструменты": [  
 "Набор взломщика",  
 "Набор исследователя подземелий",  
 "Набор путешественника",  
 ],  
 "снаряжение класса": ["Кинжалы"],  
 },  
 "Следопыт": {  
 "оружие": ["Два скимитара", "Два коротких меча"],  
 "снаряжение": ["Кожаная броня", "Чешуйчатый доспех"],  
 "инструменты": [  
 "Набор путешественника",  
 "Набор исследователя подземелий",  
 ],  
 "снаряжение класса": ["Длинный лук"],  
 },  
 "Чародей": {  
 "оружие": ["Два кинжала", "Короткий меч"],  
 "снаряжение": ["Мантия"],  
 "инструменты": [  
 "Набор путешественника",  
 "Набор исследователя подземелий",  
 "Мешочек с компонентами",  
 ],  
 "снаряжение класса": ["Магическая семейная реликвия"],  
 },  
 }  
  
 equipment\_label = QLabel(f"Выберите снаряжение для класса: {class\_name}")  
 equipment\_label.setStyleSheet("background-color: white;font-size:16px;")  
 layout.addWidget(equipment\_label)  
  
 *self*.equipment\_checkboxes = {  
 "оружие": [],  
 "снаряжение": [],  
 "инструменты": [],  
 "снаряжение класса": [],  
 }  
  
 *for* category, items *in* equipment[class\_name].items():  
 layout.addWidget(*self*.create\_equipment\_groupbox(category, items))  
  
 *self*.finish\_button = QPushButton("Закончить", *self*)  
 *self*.back\_button = QPushButton("Назад", *self*)  
 *self*.finish\_button.clicked.connect(*self*.finish\_creation)  
 *self*.back\_button.clicked.connect(*self*.return\_to\_previous\_step)  
  
 layout.addWidget(*self*.finish\_button)  
 layout.addWidget(*self*.back\_button)  
 *self*.setStyleSheet("background-color: white; font-size: 16px;")  
 *self*.setLayout(layout)  
 QMessageBox.information(  
 *self*,  
 "Снаряжение",  
 "Выберите снаряжение персонажа (По одному предмету каждого типа)",  
 )  
  
 *def* create\_equipment\_groupbox(*self*, title, items):  
 groupbox = QGroupBox(title)  
 form\_layout = QFormLayout()  
 *for* item *in* items:  
 checkbox = QCheckBox(item)  
 checkbox.stateChanged.connect(  
 *lambda* state, cb=checkbox, cat=title: *self*.update\_equipment\_selection(  
 cat, cb  
 )  
 )  
 form\_layout.addRow(checkbox)  
 *self*.equipment\_checkboxes[title].append(checkbox)  
 groupbox.setLayout(form\_layout)  
 groupbox.setStyleSheet("background-color: white;")  
 *return* groupbox  
  
 *def* update\_equipment\_selection(*self*, category, checkbox):  
 *if* checkbox.isChecked():  
 *for* cb *in self*.equipment\_checkboxes[category]:  
 *if* cb != checkbox:  
 cb.setEnabled(*False*)  
 *self*.selected\_equipment[category] = checkbox.text()  
 *else*:  
 *for* cb *in self*.equipment\_checkboxes[category]:  
 cb.setEnabled(*True*)  
 *self*.selected\_equipment[category] = *None  
  
 def* finish\_creation(*self*):  
 selected\_race = *self*.description\_window.selected\_race  
 race\_features = *self*.description\_window.parent().race\_selection\_widget.races[  
 selected\_race  
 ]["features"]  
  
 character\_data = {  
 "Имя": *self*.description\_window.character\_name,  
 "Раса": selected\_race,  
 "Класс": *self*.description\_window.class\_selection\_widget.selected\_class,  
 "Описание": *self*.description\_window.character\_description,  
 "Снаряжение": *self*.selected\_equipment,  
 "Особенности расы": race\_features,  
 }  
  
 stats\_and\_modifiers = *self*.description\_window.stat\_selection\_widget.stats  
 *for* stat, value *in* stats\_and\_modifiers.items():  
 character\_data[f"{stat}"] = f"{value} ({math.floor((value - 10) / 2)})"  
  
 *if not* os.path.exists("characters"):  
 os.makedirs("characters")  
 file\_path = os.path.join("characters", f"{character\_data['Имя']}.txt")  
 *with open*(file\_path, "w") *as* file:  
 *for* key, value *in* character\_data.items():  
 file.write(f"{key}: {value}\n")  
 QMessageBox.information(  
 *self*,  
 "Сохранение персонажа",  
 f"Персонаж '{*self*.description\_window.character\_name}' успешно создан и сохранен!",  
 )  
 *self*.description\_window.parent().is\_creating\_character = *False  
 self*.close()  
  
 *def* return\_to\_previous\_step(*self*):  
 *self*.description\_window.show()  
 *self*.close()  
  
  
*# Вызов списка персонажей  
class* CharacterListWidget(QWidget):  
 *def \_\_init\_\_*(*self*, parent):  
 *super*().*\_\_init\_\_*(parent)  
 *self*.setWindowTitle("Список персонажей")  
 *self*.setWindowFlags(Qt.FramelessWindowHint)  
 *self*.setGeometry(276, 76, 1555, 922)  
 *self*.parent = parent  
 *self*.init\_ui()  
  
 *def* init\_ui(*self*):  
 layout = QVBoxLayout()  
 *self*.character\_list = QListWidget()  
 *self*.character\_list.itemClicked.connect(*self*.display\_character\_info)  
  
 *self*.load\_characters()  
  
 layout.addWidget(*self*.character\_list)  
  
 *self*.delete\_button = QPushButton("Удалить", *self*)  
 *self*.delete\_button.clicked.connect(*self*.delete\_character)  
 layout.addWidget(*self*.delete\_button)  
  
 *self*.back\_button = QPushButton("Назад", *self*)  
 *self*.back\_button.clicked.connect(*self*.return\_to\_main\_menu)  
 layout.addWidget(*self*.back\_button)  
  
 *self*.character\_info\_text = QTextEdit()  
 *self*.character\_info\_text.setReadOnly(*True*)  
 layout.addWidget(*self*.character\_info\_text)  
  
 *self*.setLayout(layout)  
  
 *def* load\_characters(*self*):  
 *if not* os.path.exists("characters"):  
 os.makedirs("characters")  
 *for* character\_file *in* os.listdir("characters"):  
 *if* character\_file.endswith(".txt"):  
 *self*.character\_list.addItem(character\_file.replace(".txt", ""))  
  
 *def* display\_character\_info(*self*, item):  
 character\_name = item.text()  
 file\_path = os.path.join("characters", f"{character\_name}.txt")  
 *with open*(file\_path, "r") *as* file:  
 character\_info = file.read()  
 *self*.character\_info\_text.setStyleSheet("font-size: 16px;")  
 *self*.character\_info\_text.setText(character\_info)  
  
 *def* delete\_character(*self*):  
 current\_item = *self*.character\_list.currentItem()  
 *if* current\_item:  
 character\_name = current\_item.text()  
 file\_path = os.path.join("characters", f"{character\_name}.txt")  
 os.remove(file\_path)  
 *self*.character\_list.takeItem(*self*.character\_list.row(current\_item))  
 *self*.character\_info\_text.clear()  
  
 *def* return\_to\_main\_menu(*self*):  
 *self*.hide()  
 *self*.parent.is\_viewing\_characters = *False  
 self*.parent.show()  
  
  
*# Начало программы  
if* \_\_name\_\_ == "\_\_main\_\_":  
 app = QApplication(sys.argv)  
 image\_folder = "Pictures/Background"  
 icon\_path = "Pictures/Icon/D&D.ico"  
 music\_folder = "Music"  
 main\_window = MainWindow(image\_folder, icon\_path, music\_folder)  
 main\_window.show()  
 sys.exit(app.exec\_())