## Практическая работа №12. Проектирование классов

***Проект «Таск-менеджер»***

*Цель проекта*: создание программы, которая позволяет пользователям создавать новые задачи, устанавливать им приоритет, отмечать выполнение, а также просматривать и редактировать список задач.

*Функциональные требования*:

* Создание новой задачи: пользователь должен иметь возможность ввести описание задачи, установить ей приоритет (высокий, средний, низкий) и добавить в список задач.
* Установка выполнения задачи: пользователь должен иметь возможность отметить задачу как выполненную.
* Редактирование задачи: пользователь должен иметь возможность изменить описание задачи или ее приоритет.
* Просмотр списка задач: пользователь должен иметь возможность просматривать список всех задач с указанием их приоритета и статуса выполнения.

*Внешний интерфейс*. Программа должна предоставлять текстовый интерфейс для взаимодействия с пользователем. При запуске программа должна выводить приветствие и предлагать доступные команды (например, «Создать задачу», «Отметить задачу выполненной», «Редактировать задачу» и т.д.). Пользователь должен вводить команды с клавиатуры для выполнения соответствующих действий.

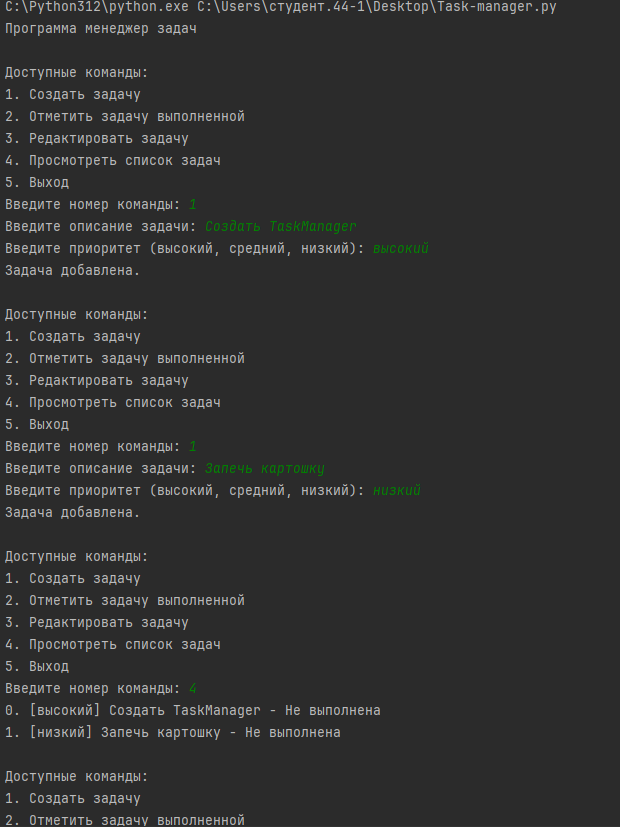
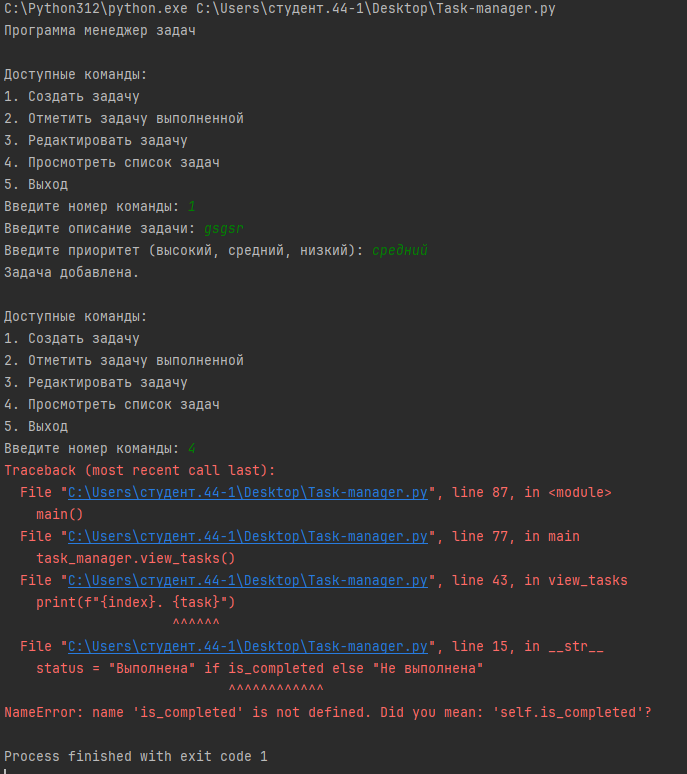
*Предлагаемый дизайн классов*:

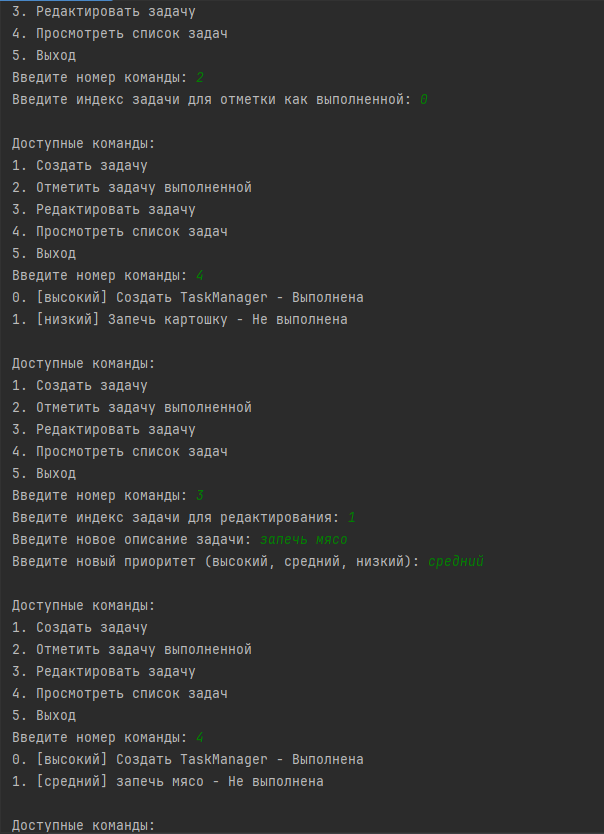
* Класс Task: представляет задачу с полями, такими как описание, приоритет и статус выполнения.
* Класс TaskManager: представляет менеджер задач и содержит список экземпляров класса Task. Методы класса TaskManager должны реализовывать функции создания, установки выполнения, редактирования и просмотра задач.

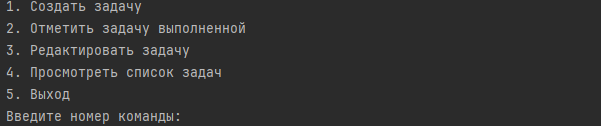
Программный код:

class Task:  
 def \_\_init\_\_(self, description, priority):  
 self.description = description  
 self.priority = priority  
 self.is\_completed = False  
  
 def mark\_completed(self):  
 self.is\_completed = True  
  
 def edit(self, new\_description, new\_priority):  
 self.description = new\_description  
 self.priority = new\_priority  
  
 def \_\_str\_\_(self):  
 status = "Выполнена" if self.is\_completed else "Не выполнена"  
 return f"[{self.priority}] {self.description} - {status}"  
  
  
class TaskManager:  
 def \_\_init\_\_(self):  
 self.tasks = []  
  
 def add\_task(self, description, priority):  
 task = Task(description, priority)  
 self.tasks.append(task)  
  
 def mark\_task\_completed(self, task\_index):  
 if 0 <= task\_index < len(self.tasks):  
 self.tasks[task\_index].mark\_completed()  
 else:  
 print("Задача с таким индексом не найдена.")  
  
 def edit\_task(self, task\_index, new\_description, new\_priority):  
 if 0 <= task\_index < len(self.tasks):  
 self.tasks[task\_index].edit(new\_description, new\_priority)  
 else:  
 print("Задача с таким индексом не найдена.")  
  
 def view\_tasks(self):  
 if not self.tasks:  
 print("Список задач пуст.")  
 for index, task in enumerate(self.tasks):  
 print(f"{index}. {task}")  
  
  
def main():  
 task\_manager = TaskManager()  
 print("Программа менеджер задач")  
  
 while True:  
 print("\nДоступные команды:")  
 print("1. Создать задачу")  
 print("2. Отметить задачу выполненной")  
 print("3. Редактировать задачу")  
 print("4. Просмотреть список задач")  
 print("5. Выход")  
  
 command = input("Введите номер команды: ")  
  
 if command == "1":  
 description = input("Введите описание задачи: ")  
 priority = input("Введите приоритет (высокий, средний, низкий): ")  
 task\_manager.add\_task(description, priority)  
 print("Задача добавлена.")  
  
 elif command == "2":  
 task\_index = int(input("Введите индекс задачи для отметки как выполненной: "))  
 task\_manager.mark\_task\_completed(task\_index)  
  
 elif command == "3":  
 task\_index = int(input("Введите индекс задачи для редактирования: "))  
 new\_description = input("Введите новое описание задачи: ")  
 new\_priority = input("Введите новый приоритет (высокий, средний, низкий): ")  
 task\_manager.edit\_task(task\_index, new\_description, new\_priority)  
  
 elif command == "4":  
 task\_manager.view\_tasks()  
  
 elif command == "5":  
 print("Выход из программы.")  
 break  
  
 else:  
 print("Неверная команда. Пожалуйста, попробуйте снова.")  
  
  
main()

Тестирование:

1. Потому что self.tasks не приняли за enumerate
2. Потому что перед is\_completed не ввели self.





Ошибки:

