Humanoid Robot

Introduction

The aim was to create a Python application that simulates a humanoid robot system by utilising object-oriented programming (OOP) principles along with data structures: a list, a stack, and a queue. The design of the system focuses on three main features: a To-Do List implemented with a list, a Calculator that uses a stack (including undo functionality), and a Support Queue based on a queue. These features showcase the real-world application of OOP principles and data structures.

Instructions to Run the Application

1. Download and install Python on your system if it is not already installed.
2. Clone the GitHub repository:
    a. git clone https://github.com/shraddha-gore/UoEO-Humanoid-Robot.git
3. Open the terminal and navigate to the project directory:
    a. cd <path to Humanoid-Robot-UoEO>
4. Run the application:
    a. python main.py
5. Run unit tests:
    a. python -m unittest discover -s tests -p '*_test.py'

Object-Oriented Programming Principles

Encapsulation

Each feature is encapsulated within its own class, keeping attributes and methods hidden from outside access. For example, the internal list in the To-Do List isn't directly visible to the user; instead, it's managed through specific methods like add_task() and remove_task().

Inheritance

A base class called Feature outlines shared attributes and methods for the various features. The three specific feature classes—ToDoList, StackCalculator, and SupportQueue—inherit from this base class, which helps minimise redundancy and encourages code reuse.

Polymorphism

Polymorphism is demonstrated in methods such as clear() and display(), which are implemented differently across all features. This allows each class to define its own behaviour while still providing a consistent interface for users.

Abstraction

The system makes complex logic easier to handle through user-friendly methods. For instance, in the calculator, users can interact with methods like add() and undo() without needing to grasp the underlying stack operations. Jain (2021) explains that abstraction conceals complexity from the user, which is a fundamental aspect of OOP that enhances system intuitiveness and lessens the cognitive burden on developers.

Development Process

During the design phase, the created class diagrams visualise the structure and relationships between classes. At first, the focus was on CRUD operations for each feature. However, as development moved forward, I recognised the necessity for additional methods, particularly getters and setters, to enable smooth data manipulation within these CRUD operations. This realisation highlights the iterative nature of software development and the need to adjust design according to implementation realities.

In addition to the core features, the menu manager was restructured as a class, which encapsulated operations for managing user interactions and delegating tasks to the appropriate feature classes. This method followed the principles of abstraction and encapsulation, ensuring that each part of the application is modular and easily extendable.

Testing involved unit tests using Python’s unittest framework, along with manual testing for functional validation. A dedicated test suite is designed for all features in the tests directory. Each feature underwent thorough testing to confirm that its methods functioned as intended. Both testing types included positive and negative scenarios to ensure system reliability.

Conclusion

The Humanoid Robot System effectively showcases how OOP principles and data structures can address real-world challenges. Although there were hurdles, like adjusting initial designs to meet implementation requirements, the system successfully met its objectives of modularity, scalability, and functionality. It establishes a solid foundation for future enhancements, including the integration of new features and the implementation of logging, among others.

References

Jain, J. (2021) *A Quick Guide on OOPs Concept in Python*. Available at: https://medium.com/codex/a-quick-guide-on-oops-concept-in-python-3ee6c40cf673 (Accessed: 11 January 2025).