**Implementation Comparison Python and Java: Expense Tracker Application**

Nitish Dhinaharan, Nicolas Ferradas Bertoli, Alexa R. Fisher, Arun Bhaskar Gaddle

Department of Information Technology, University of the Cumberlands

MSCS-632-B01 Advanced Programming Languages

Dr. Dax Bradley

April 6, 2025

**Implementation Comparison Python and Java: Expense Tracker Application**

Python and Java are a few of the most popular programming languages in recent years. The members selected the languages due to specific expertise. The design of the Expense Tracker applications highlighted the key differences between the programming languages. Feature differences underscored in this report include data structures, memory management, and error handling. Each language's characteristics affected the application's design, performance, and readability.

**Java**

**Data Structures**

**Memory Management**

**Error Handling**

**Python**

**Data Structures**

Data structures within Python are built-in (Phillips et al., 2016). The data structures emphasize Python’s ease of use and writability (Phillips et al., 2016). A few highlighted data structures include dictionaries and lists. Lists allow developers to store items dynamically (Phillips et al., 2016). Dictionaries focus on key-value pairing for storage and retrieval (Phillips et al., 2016). The dictionary containers are ideal for mapping objects (Phillips et al., 2016). Developers often utilize dictionaries for searching (Phillips et al., 2016).

**Memory Management**

Python takes a hassle-free approach to memory management (Yegulalp, 2022). Developers do not have to allocate, manage, or dispose of memory manually (Yegulalp, 2022). Within Python, memory management is automatic and completed during runtime (Yegulalp, 2022). It utilizes reference counting, commonly noted as refcount, in which each object tracks the number of other objects referencing it (Yegulalp, 2022). Once the object is no longer referenced and the count reaches zero, Python automatically deallocates its memory (Yegulalp, 2022).

**Error Handling**

Error handling in Python is crucial. Due to Python’s dynamic typing, runtime interpretation can potentially yield unexpected inputs (Lott & Phillips, 2021). Developers can mitigate these inputs by handling exceptions (Lott & Phillips, 2021). A common way to handle the exceptions is a “try…except” clause (Lott & Phillips, 2021). The clause tries the reference code; if invalid input occurs, an exception halts the processing (Lott & Phillips, 2021). Frequently, developers include a print statement describing the error.

**Comparison**

The comparison of Python and Java reveals distinct design characteristics. Python features dynamic typing, which allows for high flexibility. Data structures are built-in, simplifying development (Phillips et al., 2016). Examples of data structures within Python include lists and dictionaries, which allow for effective performance and readability. Dictionaries allow rapid searching based on key pairs, and lists focus on sequences. The application used dictionaries and lists to create methods for filtering objects. Memory management in Python is automatic (Yegulalp, 2022). It highlights the ease of use and emphasizes simplistic design at the potential cost of performance overhead (Yegulalp, 2022). The “deleteExpense()” method removed a transaction from the transaction list within the Python application. The memory used by the deleted transaction’s dictionary was eventually freed automatically by Python’s garbage collector (Yegulalp, 2022). Developers utilize “try…except” code blocks to implement error handling (Lott & Phillips, 2021). It improves debugging and maintainability, contributing to the code’s readability (Lott & Phillips, 2021). [A few sentences about Java data structures, error handling, and memory management in terms of design, performance, and readability. ] The figures below visualize the key differences between each language regarding the main features of data structures, memory management, and error handling.

**Figure 1**

*Comparison of Data Structures*

| **Data Structures** | |
| --- | --- |
| Java | Python |
| Insert Code Snippet… | class Account:    def \_\_init\_\_(self) – > None:  self.transactions: List[Transaction] = []  self. transaction\_by\_type = defaultdict(list) # Caches transactions by type  self.lock = threading.Lock() # Ensures thread-safe operations |

*Note:* [Add a note for Java] The Python code uses lists, notated with brackets “[ ]”, to hold transaction dictionaries.

**Figure 2**

*Comparison of Error Handling*

| **Error Handling** | |
| --- | --- |
| Java | Python |
| Insert Code Snippet… | def \_parse\_date(self, date\_str: str) - > Optional[datetime.date]:  try:  return datetime.datetime.strptime(date, "%Y-%m-%d").date()  except ValueError:  print("Error: Incorrect date format. Use YYYY-MM-DD.")  return None |

*Note:*[Add a note for Java] The Python code highlights an example of error handling utilizing the “try…except” code block to a date format Value Error.

**Figure 3**

*Comparison of Memory management*

| **Memory Management** | |
| --- | --- |
| Java | Python |
| Insert Code Snippet… | There is no code example; memory management is automatic. |

*Note:*[Add a note for Java]

**Conclusion**

[Concluding paragraph]

**References**

Lott, S. F., & Phillips, D. (2021). Python Object-Oriented Programming : Build Robust and Maintainable Object-oriented Python Applications and Libraries. (pp. 125–162). Packt Publishing, Limited. https://research.ebsco.com/plink/91471866-7c84-3156-baf4-e96080663282

Phillips, D., Romano, F., Vo.T.H, P., Layton, R., Czygan, M., & Raschka, S. (2016). Python Data Structures. In Python: Real-World Data Science. (pp. 199–232). Packt Publishing, Limited. https://research.ebsco.com/plink/38bbd789-a4c7-32f3-921a-1544dac25d41

Yegulalp, S. (2022). Python garbage collection and the gc module. InfoWorld.Com, https://www.proquest.com/trade-journals/python-garbage-collection-gc-module/docview/2714107855/se-2