UDC 519.6
DOI https://doi.org/10.26661/2786-6254-2024-1-08

VISUALIZATION OF HASH MAP DATA STRUCTURE

Kolesnikov V. A.
PhD, Professor at the Department of Computer Science
Sumy State University
Kharkivska str., 116, Sumy, Ukraine
orcid.org/0000-0002-1991-3614
v.kolesnikov@cs.sumdu.edu.ua

Key words: *visualization, data structure, hash map, design, application.*

In today's data-centric world, understanding the intricate relationships within complex data structures is essential for making informed decisions and deriving meaningful insights. This article delves into the development of an application tailored for visualizing hash map data structures. Hash maps, a fundamental data structure in computer science, are widely used for efficient data storage and retrieval. However, comprehending the internal workings of hash maps can be challenging due to their dynamic nature and underlying complexities. The application discussed in this article offers a solution by providing users with an intuitive and interactive platform to explore hash maps comprehensively. Leveraging modern visualization techniques and user-centered design principles, the application transforms abstract data structures into visually engaging representations. Users can interact with the hash map visualization, dynamically adding or removing key-value pairs, adjusting parameters, and exploring the structure's nuances in real-time. Key features of the application include interactive animations to illustrate hash collisions and resizing. By combining these elements, the application facilitates a deeper understanding of hash map operations and behaviors. The application's user interface is designed to be intuitive and accessible, catering to users with varying levels of expertise in data structures and visualization. Error handling mechanisms are implemented to detect and address common issues, such as invalid input or hash collisions, ensuring the integrity of the visualization and enhancing the user experience. Overall, this article demonstrates the transformative potential of visualization techniques in elucidating complex data structures. By providing a user-friendly platform for visualizing hash maps, the application empowers users to explore, analyze, and gain insights from their data effectively. The discussion also highlights avenues for future research and development, including advanced visualization techniques, integration with other data structures, and applications in diverse domains such as education, software development, and data analysis.

ВІЗУАЛІЗАЦІЯ СТРУКТУРИ ДАНИХ HASH MAP

Колесніков В. А.
PhD, професор кафедри комп'ютерних наук
Сумський державний університет
вул. Харківська, 116, Суми, Україна
orcid.org/0000-0002-1991-3614
v.kolesnikov@cs.sumdu.edu.ua

Ключові слова: *visualization, data structure, hash map, design, application.*

У сучасному світі, орієнтованому на дані, розуміння складних взаємозв'язків у складних структурах даних має важливе значення для прийняття обґрунтованих рішень та отримання значущої інформації. У статті йдеться про розробку програми, призначеної для візуалізації структур даних хеш-мап. Хеш-мапа, фундаментальна структура даних в інформатиці, широко використовуються для ефективного зберігання та пошуку даних. Однак зрозуміти внутрішню роботу хеш-мап може бути складно через їх динамічну природу та складнощі, що лежать в основі цієї структури даних. Програма, про яку йдеться у статті, пропонує рішення, надаючи користувачам інтуїтивно зрозумілу інтерактивну платформу для всебічного вивчення хеш-мап. Використовуючи сучасні методи візуалізації й орієнтовані на користувача принципи проектування, програма перетворює абстрактні структури даних у візуально привабливі представлення. Користувачі можуть взаємодіяти з візуалізацією хеш-мапи, динамічно додаючи або видаляючи пари ключ – значення, коригуючи параметри та досліджуючи нюанси структури даних у режимі реального часу. Основні функції програми включають інтерактивну анімацію для ілюстрації хеш-колізій і зміни розміру самої структури даних. Поєднуючи ці елементи, програма сприяє глибшому розумінню операцій хеш-мапи та поведінки. Інтерфейс користувача програми розроблений інтуїтивно зрозумілим і доступним для користувачів із різним рівнем знань у структурах даних і візуалізації. Механізми обробки помилок реалізовано для виявлення та вирішення поширених проблем, таких як недійсний вхід або колізії хешів, забезпечуючи цілісність візуалізації та покращуючи взаємодію з користувачем. Загалом ця стаття демонструє трансформаційний потенціал методів візуалізації для з'ясування складних структур даних. Забезпечуючи зручну платформу для візуалізації хеш-мап, програма дає користувачам змогу ефективно досліджувати, аналізувати й отримувати розуміння своїх даних. Обговорення також висвітлює шляхи майбутніх досліджень і розробок, включаючи передові методи візуалізації, інтеграцію з іншими структурами даних і застосування в різних сферах, таких як освіта, розробка програмного забезпечення й аналіз даних.

1. Introduction. In today's data-driven world, the ability to comprehend and manipulate complex datasets efficiently is paramount. Data structures serve as the backbone of computer science, providing organized ways to store, access, and manipulate data. As datasets grow in size and complexity, the need for effective visualization techniques becomes increasingly crucial. Visualization of data structures bridges the gap between abstract data representations and human comprehension, offering intuitive ways to analyze and understand intricate data relationships.

This paper underscores the significance of visualizing data structures, delving into a simple visualization approach and tools employed to represent diverse types of data structures effectively. From fundamental structures like arrays and linked lists to more intricate ones such as trees, graphs, hash tables and hash maps, visualizations offer insights into the inner workings and behaviors of these structures. By leveraging visual representations, researchers, developers, and analysts can gain deeper insights, identify patterns, detect anomalies, and make informed decisions based on their data.

The effectiveness of a visualization heavily relies on its ability to convey complex information clearly and concisely. Therefore, this paper also strives to

put into practice principles of effective visualization design, including scalability, interactivity, and aesthetic appeal.

The research in this area has been extensive. For example, in [1] a sketch-based interface CSTutor is presented as an environment in which a user's sketched diagram and code are combined seamlessly. The tool analyzes the user's diagrams in real time and automatically generates code. The tool is intended to bridge the gap between the conceptual diagram of a data structure and the actual implementation. In [2] a web-based application for visualization of bubble, sequential and selection sorting algorithms are presented. [3] demonstrates preliminary work on building interactive teaching modules for data structures and algorithms courses that allow students to visualize course material and assess, log and analyze their learning progress. [4] presents an interactive visualization tool for understanding data structures. The work uses the inner execution trace to monitor the run-time data structure change and render the corresponding presentation in real-time for review and analysis. In [5] a dynamic heap analysis system is described that allows to examine and analyze how Java programs build and modify data structures. [6] presents Travioli system that can be used for visualizing data-structure traversals. [7] surveys the state of the art in visualizing dynamin graphs. The review of this research shows that visualization approaches are difficult for an average user and are highly specific for a given context.

In this paper we strive for simplicity in presenting the innerworkings of a HashMap by developing a java-based application for visualization of the given data structure. We also describe the design of the application that serves as the basis for the application implementation.

By elucidating the importance and potential of visualization techniques, this research endeavors to inspire further innovation in the field, fostering the development of simple but more powerful and intuitive tools for data analysis and interpretation.

2. Methods. Developing an application for visualizing a hash map data structure involved several key steps and methods. Here's an outline of how we approached this:

First, we defined the requirements and objectives of the visualization application. We considered the target audience, the specific features we wanted to include, and the platforms on which the application would run. With the help of this application, the user must be able to track changes in the hash map and its structure. The user must be able to add and remove values from the hash map, search for values by key, find the size of the structure, and clear it completely. This application can be used in the process of training specialists in computer science, as it helps to study the innerworkings of hash map data structure, as well as for individual research.

Next, we chose a Programming Language and Framework. The project code is written in the Java programming language version 17. We used the java.awt and javax.swing packages for the project interface. Next, we performed the User Interface design. We designed an intuitive user interface that allows users to interact with the hash map visualization effectively. We considered incorporating features such as search functionality, tooltips for displaying additional information, and controls for adjusting visualization parameters.

The implementation of the Hash Map Data Structure visualization followed. We developed the underlying hash map data structure in java programming language and implemented methods for adding, removing, and retrieving key-value pairs. Then, we chose the Visual Representation of the hash map. This involved using shapes and symbols to represent individual buckets, color-coding, and animations to illustrate insertions, deletions, and hash collisions. Next, we performed Rendering and Display using java.awt and javax.swing java libraries. We ensured that the visualization was responsive and could handle large datasets efficiently. We also considered Interactivity and implemented interactive features to enhance user engagement. This included allowing users to interactively add or remove key-value pairs, adjust visualization settings, and explore the hash map structure dynamically.

Then, Error Handling and Validation phase followed. We implemented error handling and validation mechanisms to ensure the integrity of the hash map data structure. This included handling edge cases such as invalid input, hash collisions, and resizing of the hash table appropriately.

And, finally, we performed Testing and Debugging. We thoroughly tested the visualization application to identify and address any bugs or issues, conducted both unit tests and user acceptance tests to validate the functionality and usability of the application.

3. Implementation of Application and Results

3.1. Screen Design. During the User Interface design phase, we developed the prototype of the main screen shown in Figure 1. The screen is divided into visualization and control panels. The control panel contains fields for key and value and buttons to perform operations with Hash Map data structure.

3.2. UML use-case diagram. After the requirement gathering phase, the requirements have been analyzed and the use-case diagram was created for use cases UC-01 through UC-06 (see Fig. 2). These are the use cases: UC-01 Viewing the graphic representation of Hash Map data structure, UC-02 Adding an object to the Hash Map data structure, UC-03

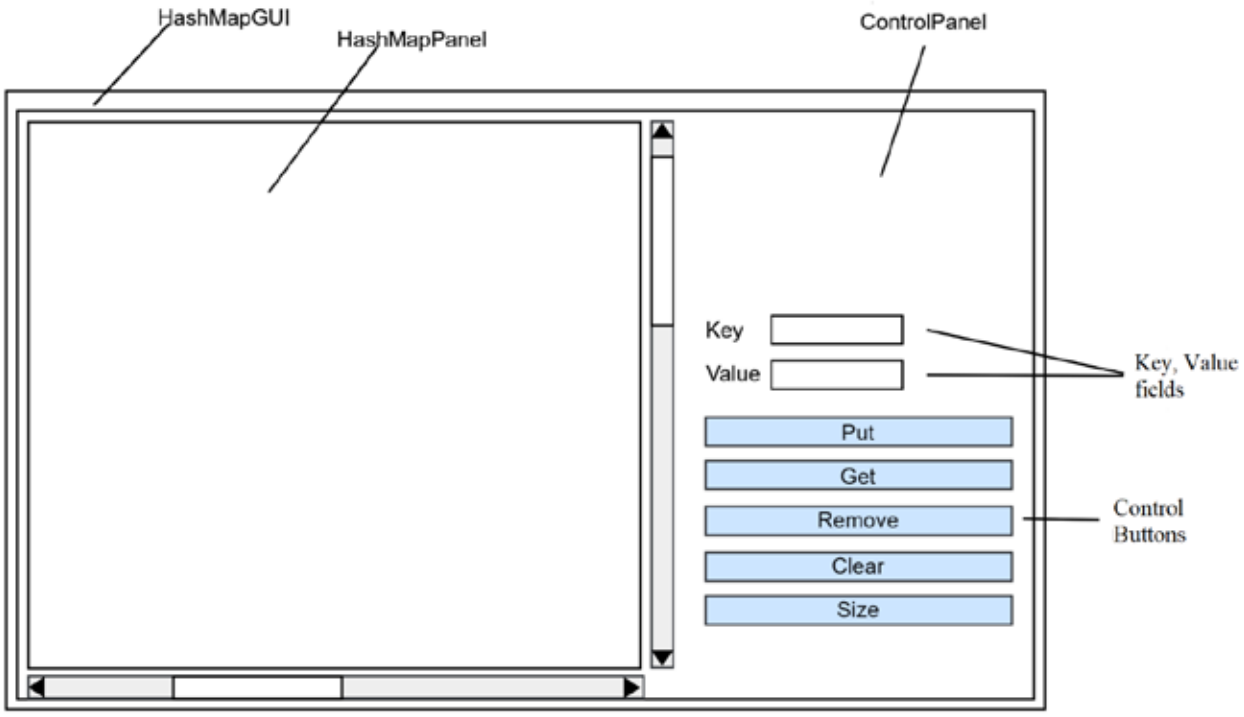

Fig. 1. Screen design

Getting the object by key from the Hash Map data structure, UC-04 Deleting the object by key from the Hash Map data structure, UC-05 Clearing the Hash Map data structure and UC-06 Getting the size of the Hash Map data structure.

3.3. UML class diagram. After the requirements have been analyzed for possible classes, the user interface screen model and the use case diagram created, we identified the classes and the relationships among them shown in Figure 3.

3.5. UML sequence diagrams. After the use-case and class diagrams were created, we created sequence diagrams for all use cases from the use-case diagram. We show one such sequence diagram for use case UC-02 Adding an object to the Hash Map data structure in Figure 4.

3.6. Screen shots of application at work. Next, we show some screen shots of the application during its work. Figure 5 shows the main window after the application has been started. The program has two panels: a panel for displaying the Hash Map data structure and a control panel. The initial size of the Hash Map data structure is 16 cells.

Figure 6 shows adding a new value in the Hash Map data structure.

Figure 7 shows a scenario when a value is added to the already occupied cell. As can be seen, a linked list is created in cell 15.

Figure 8 shows a scenario when the number of elements added to the Hash Map data structure is greater than 75% of its capacity. As can be seen, when adding

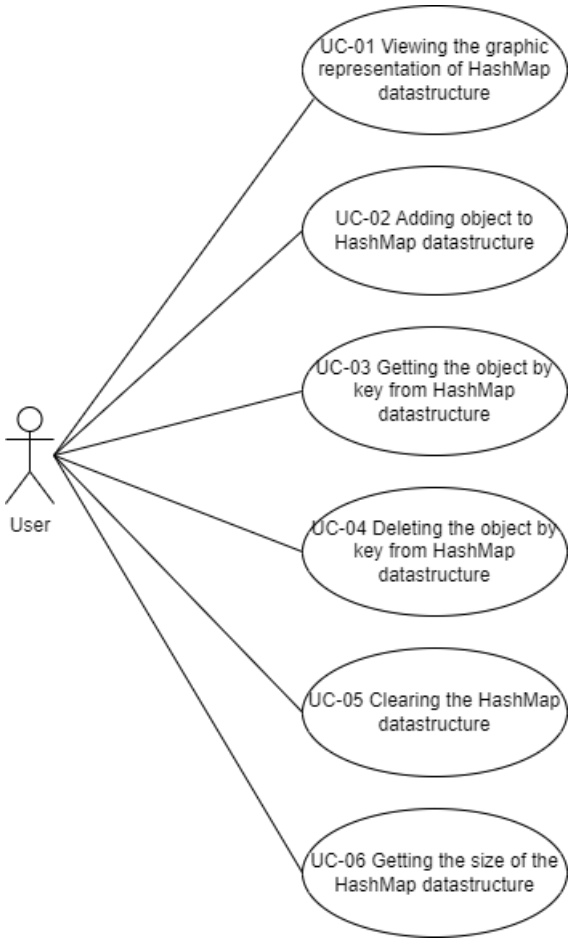

Fig. 2. Use-case diagram

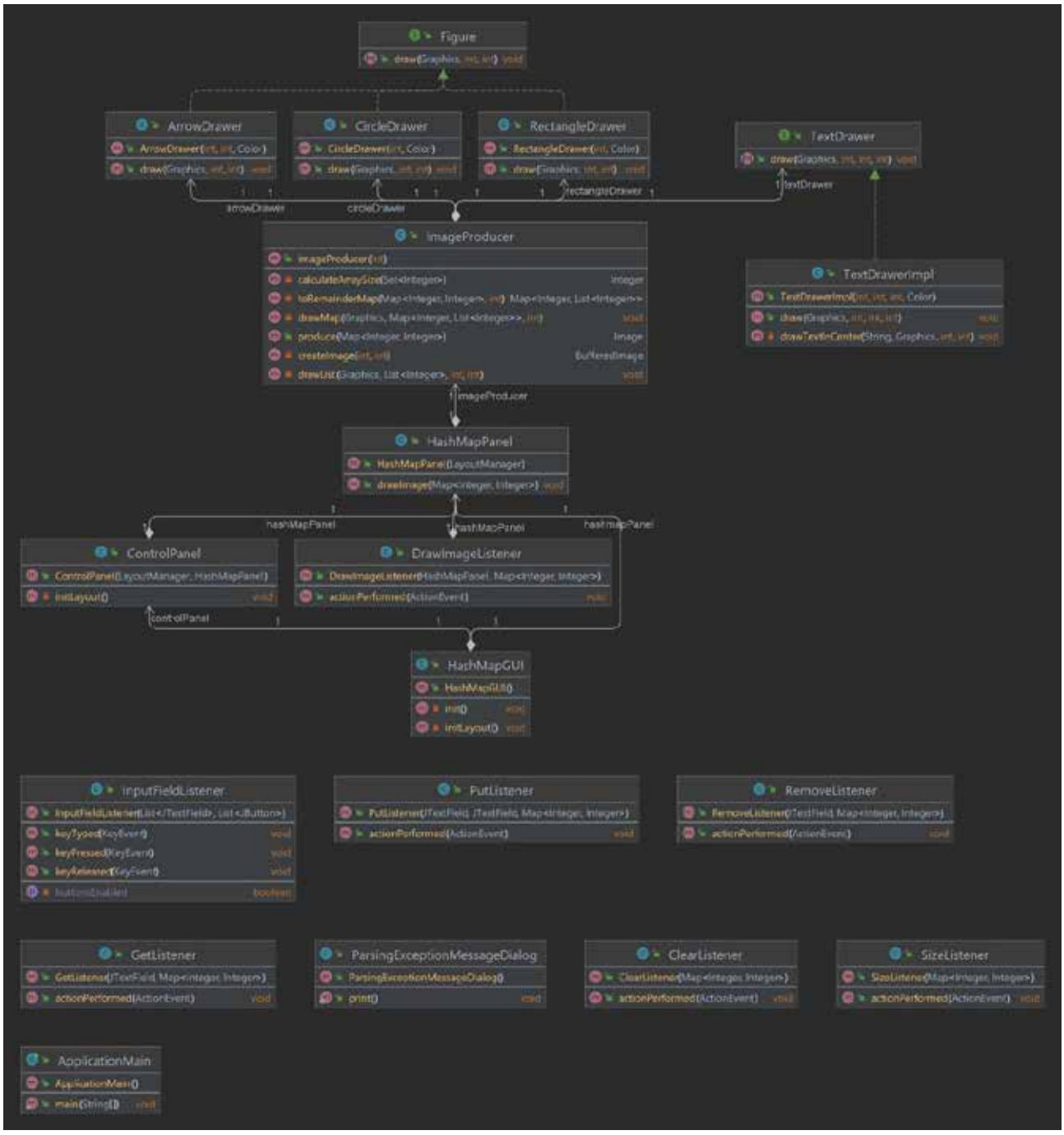

Fig. 3. UML class diagram

the 13th element, the size of the Hash Map data structure is doubled.

4. Discussion. The development of our hash map visualization application offers valuable insights into the challenges and opportunities inherent in representing complex data structures in a visual format. Through the process of designing and implementing the application, several key points emerge for consideration.

Firstly, the choice of visualization techniques significantly impacts the usability and effectiveness of the application. By employing intuitive visual representations, such as color-coded buckets and interactive animations, users can gain a deeper understanding of the hash map's structure and behavior. However, striking a balance between visual appeal and clarity is crucial to prevent information overload and maintain usability.

Secondly, the interactivity features incorporated into the application play a pivotal role in enhancing user engagement and exploration. Allowing users to dynamically interact with the hash map, add or remove

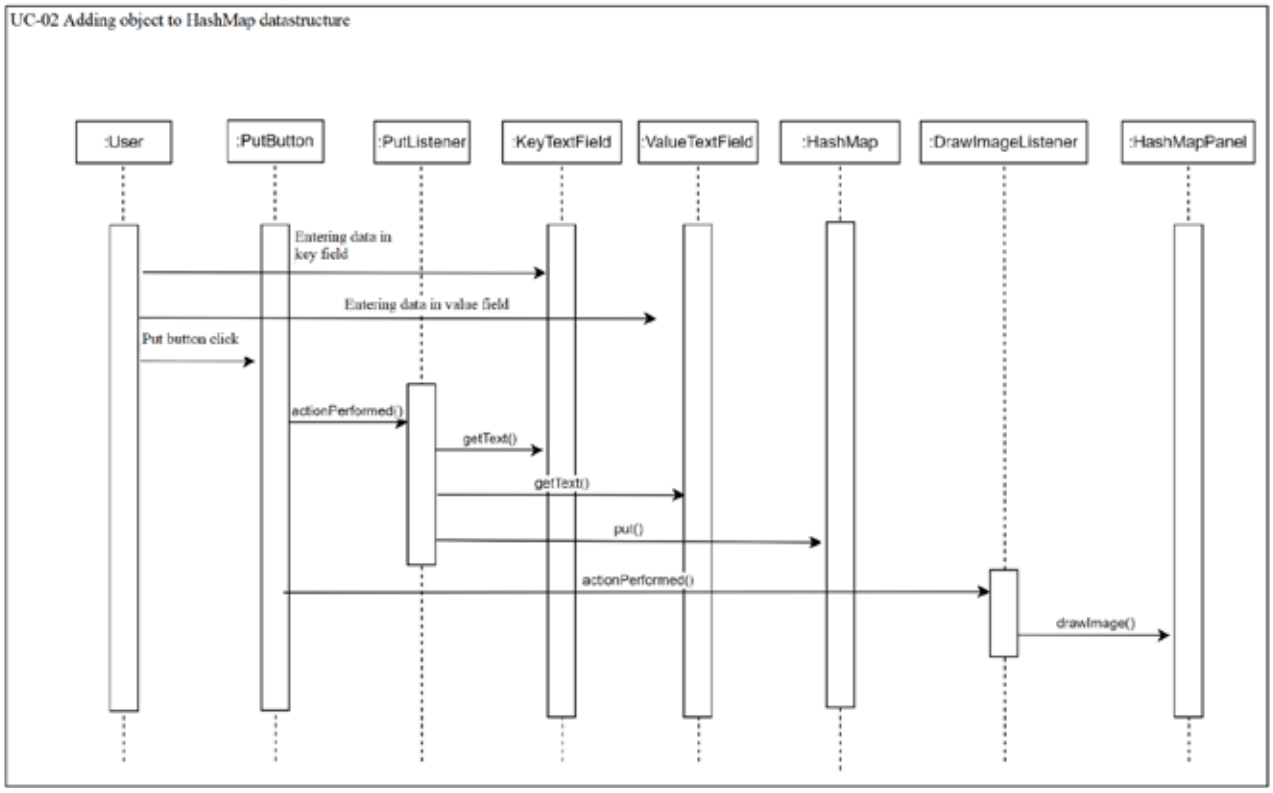

Fig. 4. UML sequence diagram for use case UC-02

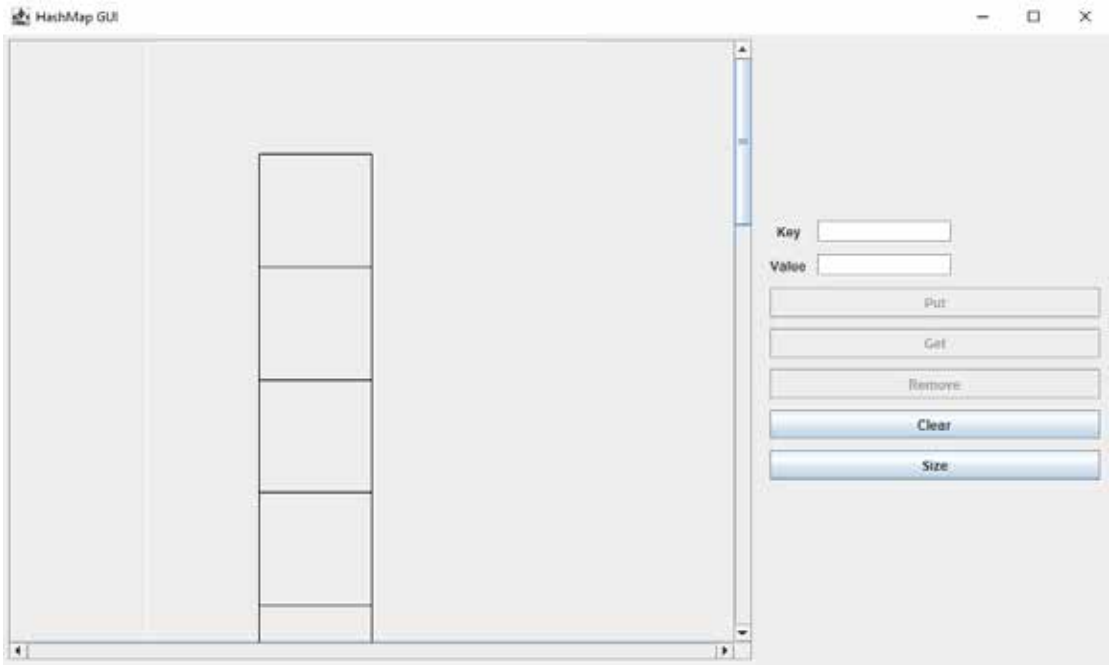

Fig. 5. Initial screen

key-value pairs, and adjust visualization parameters empowers them to customize their experience and delve into specific aspects of the data structure. This emphasizes the importance of user-centered design principles in developing effective visualization tools.

Additionally, the application highlights the significance of error handling and validation mechanisms in ensuring the integrity of the hash map data structure. Handling edge cases such as hash collisions, resizing, and invalid input gracefully contributes to the

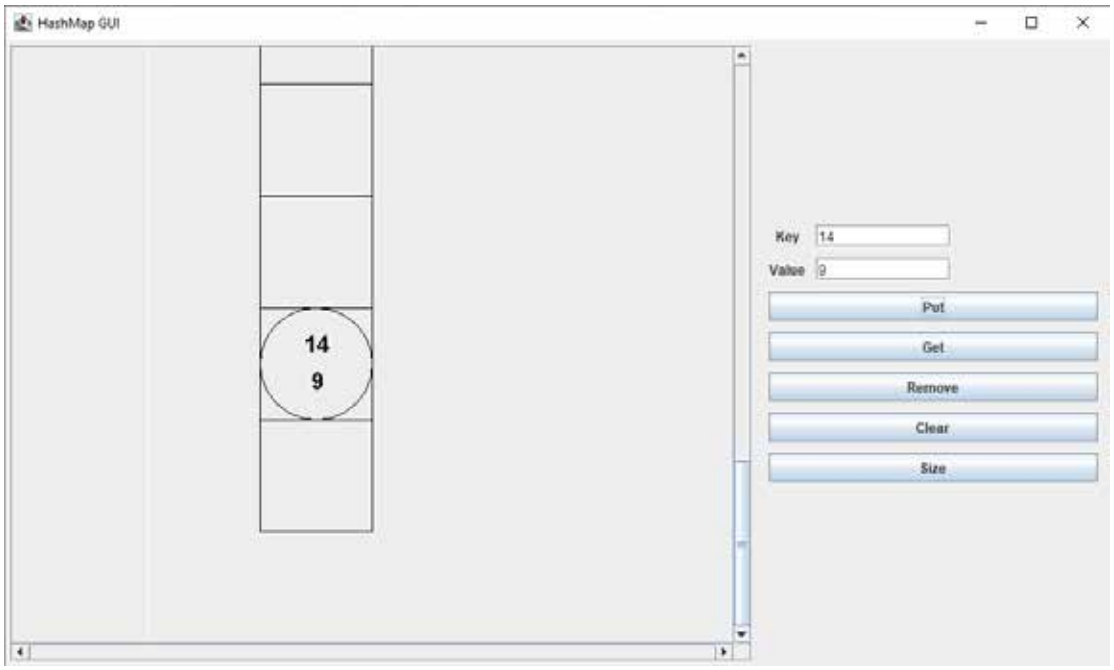

Fig. 6. Adding a new value

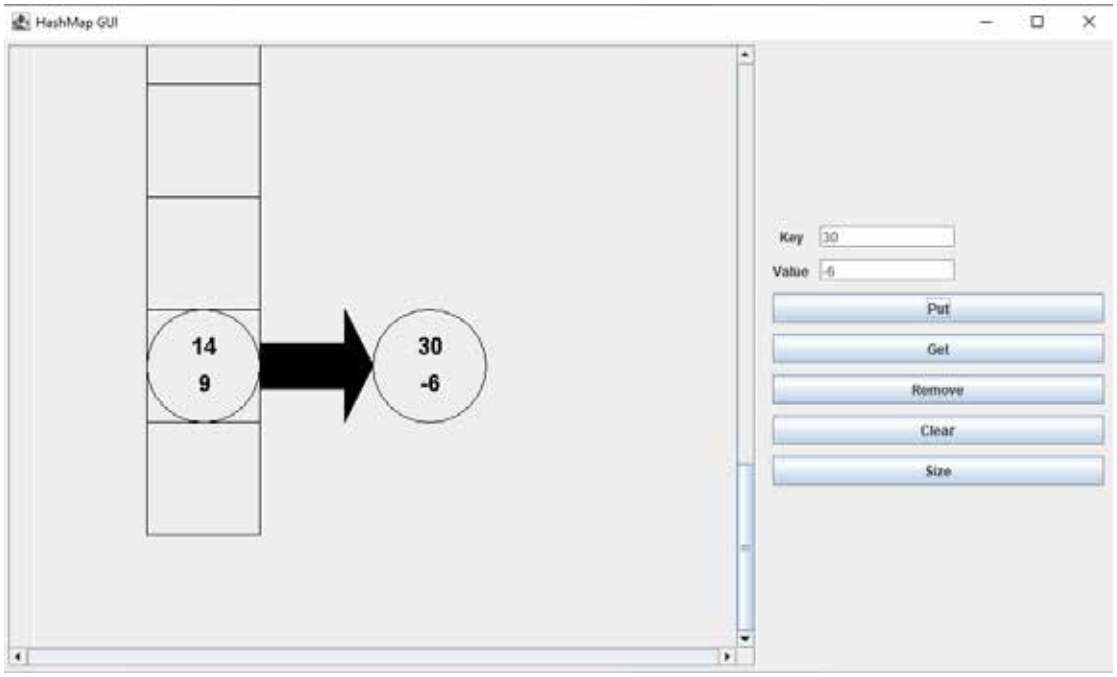

Fig. 7. Adding a value with the hashed key cell occupied

robustness and reliability of the application. Moreover, providing informative feedback to users when errors occur enhances the overall user experience and fosters trust in the application.

Despite the advancements made in developing the hash map visualization application, several challenges and opportunities for future improvement remain. Enhancing the scalability and performance of the application to handle larger datasets efficiently is a critical area for further development. Additionally, exploring alternative visualization techniques and incorporating advanced features, such as data clustering or trend analysis, could enrich the user experience and expand the application's capabilities.

5. Conclusion. In conclusion, the development of the hash map visualization application represents a

Fig. 8. Doubling the size when reaching 75% capacity

significant step towards facilitating the understanding and exploration of complex data structures. By leveraging intuitive visual representations, interactive features, and robust error handling mechanisms, the application empowers users to gain insights into the inner workings of hash maps effectively.

While the current version of the application demonstrates promising functionality and usability, there is ample room for further refinement and innovation. Continued efforts to enhance scalability, performance, and feature richness will contribute to the application's utility and relevance in various domains, including education, software development, and data analysis.

Overall, the hash map visualization application serves as a testament to the transformative potential of visualization techniques in elucidating intricate data structures. By embracing principles of user-centered design, innovation, and continuous improvement, we can unlock new possibilities for visualizing and understanding complex data in the digital age.

BIBLIOGRAPHY

1. Buchanan, S., Ochs, B., & LaViola Jr, J. J. (2012). CSTutor: a pen-based tutor for data structure visualization. Proceedings of the 43rd ACM technical symposium on Computer Science Education (pp. 565–570). doi: 10.1145/2157136.2157297
2. Najm, I. A., Hammash, N. M., Ismail, M., & Fatah, P. (2015). Web based application visualization for comprehensive data structures. Journal of Theoretical & Applied Information Technology, 77(2).
3. Mcquaigue, M., Burlinson, D., Subramanian, K., Saule, E., & Payton, J. (2018). Visualization, assessment and analytics in data structures learning modules. Proceedings of the 49th ACM Technical Symposium on Computer Science Education (pp. 864–869). doi: 10.1145/3159450.3159460
4. Lin, J., & Zhang, H. (2020). Data structure visualization on the Web. 2020 IEEE International Conference on Big Data (Big Data) (pp. 3272–3279). IEEE. doi: 10.1109/BigData50022.2020.9378249
5. Pheng, S., & Verbrugge, C. (2006). Dynamic Data Structure Analysis for Java Programs. 14th IEEE International Conference on Program Comprehension (ICPC'06) (pp. 191–201). doi: 10.1109/ICPC.2006.20
6. Padhye, R., & Sen, K. (2017) Travioli: A Dynamic Analysis for Detecting Data-Structure Traversals. 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE) (pp. 473–483). doi: 10.1109/ICSE.2017.50
7. Beck, F., Burch, M., Diehl, S., & Weiskopf, D. (2014). The State of the Art in Visualizing Dynamic Graphs. EuroVis (STARs).

REFERENCES

1. Buchanan S., Ochs B., LaViola J. CSTutor: a pen-based tutor for data structure visualization. Proceedings of the 43rd ACM technical symposium on Computer Science Education. 2012. P. 565–570. https://doi.org/10.1145/2157136.2157297

2. Najm I., Hammash N., Ismail M., Fatah P. Web based application visualization for comprehensive data structures. Journal of Theoretical & Applied Information Technology. 2015. Vol. 77, № 2.
3. Mcquaigue M., Burlinson D., Subramanian K. Visualization, assessment and analytics in data structures learning modules. Proceedings of the 49th ACM Technical Symposium on Computer Science Education. 2018. P. 864–869. https://doi.org/10.1145/3159450.3159460
4. Lin J., Zhang H. Data structure visualization on the Web. 2020 IEEE International Conference on Big Data. 2020. P. 3272–3279. https://doi.org/10.1109/BigData50022.2020.9378249
5. Pheng S., Verbrugge C. Dynamic Data Structure Analysis for Java Programs. 14th IEEE International Conference on Program Comprehension (ICPC'06). 2006. P. 191–201. https://doi.org/10.1109/ICPC.2006.20
6. Padhye R., Sen K. Travioli: A Dynamic Analysis for Detecting Data-Structure Traversals. 2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). 2017. P. 473–483. https://doi.org/10.1109/ICSE.2017.50
7. Beck F., Burch M., Diehl S. The State of the Art in Visualizing Dynamic Graphs. EuroVis (STARs). 2014.