ITEJ
Information Technology Engineering Journals
eISSN : 2548-2157

Url : https://syekhnurjati.ac.id/journal/index.php/itej
Email : itej@syekhnurjati.ac.id

Building a Product Category System for a Shopping App with Inheritance in Java

Saluky
Informatics Departement
UIN Siber Syekh Nurjati Cirebon
saluky@uinssc.ac.id

Revi Injani
Informatics Departement
UIN Siber Syekh Nurjati Cirebon
reviinjani24@gmail.com

Citta Amelia
Informatics Departement
UIN Siber Syekh Nurjati Cirebon
cittaamelia9@gmail.com

Abstract—This article discusses the development of a product category system for an online shopping application using the concept of inheritance in Java programming. In a shopping application, efficient product category management is essential to make it easier for users to search for items according to their type and preferences. This system is designed using an inheritance structure to define various product categories hierarchically. The parent category will provide basic attributes and methods, while the child category will inherit and develop these functionalities according to the specific needs of the product. The use of inheritance allows for more modular and manageable coding, and increases the ability to expand the application in the future. With this system, the shopping application not only simplifies the product search process but also allows for more structured and flexible product grouping. The implementation in Java is done using base classes and child classes, as well as a polymorphism mechanism to optimize interactions between product category objects. The results of this study indicate that an inheritance-based approach in Java can improve the efficiency and readability of code in developing shopping applications with dynamic and easily developed product categories.

Keywords—shopping application, java, inheritance, product category system, object oriented programming

I. INTRODUCTION

In developing online shopping applications, product category management is an important element to make it easier for users to navigate various types of products[1]. A good category system can improve the user experience by presenting products based on certain groups or types[2]. However, as the number of products grows, efficient category management becomes a challenge[3].

For this reason, the use of object-oriented programming (OOP) principles, especially the concept of inheritance, can be an effective solution[4]. With inheritance, we can create a hierarchy of product categories, where the base class defines common attributes, while the derived class inherits and adds specific functionality for more detailed categories. This approach allows the development of a more modular, easy-to-manage, and scalable system[5][6].

This article aims to explain how to build a product category system for a shopping application using inheritance in Java[7]. Through this approach, product category management becomes more structured and easy to develop, both for simple and complex product categories[8]. This article will review the basic concept of inheritance, as well as provide examples of practical implementation in the context of a Java-based online shopping application[9].

II. RELATED WORKS

In this study, the inheritance approach helps reduce code duplication and improve system maintenance efficiency[10].

Another study focuses on the application of inheritance in managing product and category data on online shopping platforms[11]. They identified that by using inheritance, applications can handle product category variations dynamically, while maintaining data consistency and reducing code complexity, which has an impact on more efficient application management[12].

On the other hand, discusses the application of OOP in developing e-commerce systems by using inheritance for product category structures[13][14]. They show how the application of inheritance principles allows for better category management, as well as providing flexibility in adding specific features for certain product categories without affecting the overall system[15][16].

In addition, proposes the use of inheritance in the design of microservices-based product categories for online shopping platforms[17]. This approach not only organizes product data more efficiently, but also enables system scalability by considering factors such as distributed processing and large-scale product management[18][19].

These works show that the use of inheritance in the design of product category systems continues to grow, with a focus on improving the efficiency, flexibility, and scalability of online shopping applications[20].

III. METHOD

These methods cover the basic operations of adding, deleting, and displaying product categories, as well as updating attributes for each category. The methodology is divided into several stages as follows:

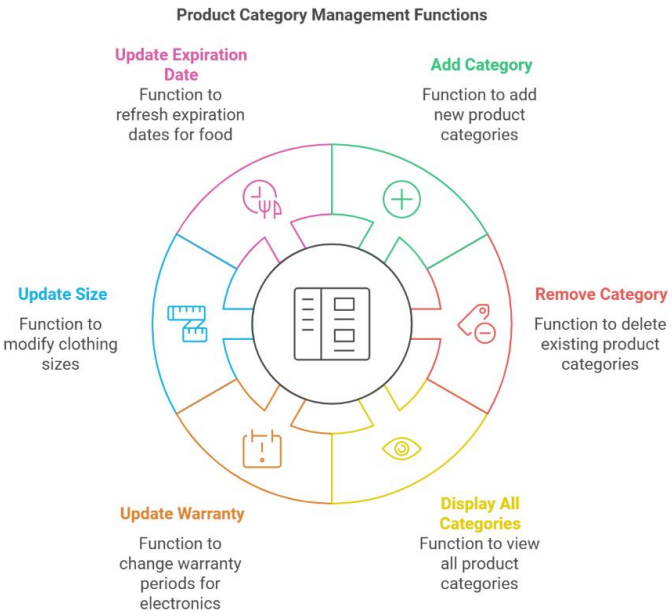

Figure 1. Product Category Management Function

A. Add Category

is used to add a new product category to the existing category list. In this application, product categories are stored in an ArrayList, which allows dynamic category management. When a user wants to add a new category (such as electronics, clothing, or food), this method will add it to the list.

B. Remove Category

is used to delete an existing product category based on categoryId. Each product category has a unique ID that is used to distinguish one category from another. Using this ID, this method will search for the category in the list based on that ID and remove it from the list.

C. Display All Categories

is used to display complete information of all product categories in the list. This helps the admin or application user to see the complete list of categories registered in the system.

D. Update Warranty

is used to display complete information of all product categories in the list. This helps the admin or application user to see the complete list of categories registered in the system.

E. Update Size

is used to update the clothing size in the clothing product category. Size is important information for clothing products because it allows customers to choose products that fit their body size.

F. Update Expiration Date

used to update clothing sizes in the clothing product category. Size is important information for clothing products because it allows customers to choose products that fit their body size.

III. RESULT AND DISCUSSION

The developed product category management system successfully meets the main objectives in managing product categories in shopping applications with various features:

A. Adding and Deleting Categories:

Admin can add new categories (e.g. Electronics, Clothing, Food) or delete old categories based on categoryId. The use of ArrayList allows dynamic category management without limitations on the number of categories.

The Adding and Deleting Categories feature enables administrators to manage product classifications dynamically within the system. Categories serve as a mechanism for grouping products based on similar characteristics, making it easier for users to browse, search, and manage inventory data. Examples of categories include Electronics, Clothing, Food, Books, and Home Appliances. In this system, category data are stored using the ArrayList data structure. Unlike conventional arrays that require a fixed size during initialization, ArrayList provides a flexible and dynamic storage mechanism. Administrators can continuously add new categories as business needs evolve without redefining the storage capacity. Similarly, categories that are no longer relevant can be removed efficiently from the collection.

When an administrator adds a category, the system creates a new category object containing a unique identifier (categoryId) and the category name (categoryName). The new object is then appended to the ArrayList. Before insertion, the system should validate

whether the categoryId already exists to prevent duplicate entries. For category deletion, the administrator specifies the corresponding categoryId. The system searches the ArrayList sequentially to locate the matching category. If the category is found, it is removed from the collection, and the remaining elements automatically shift their positions. If the specified categoryId does not exist, the system displays an appropriate notification message indicating that the deletion process failed.

The use of ArrayList offers several advantages in category management. First, it provides scalability, as the number of categories can grow according to organizational requirements. Second, it improves maintainability, allowing administrators to update category information without modifying the underlying data structure. Third, it enhances operational efficiency, as insertion operations can be performed directly at the end of the list with minimal implementation complexity.

The general workflow of the category management process is illustrated in Table 1.

Table 1. Workflow of Adding and Deleting Categories

Step	Process	Description	Output
1	Input Category Data	Administrator enters categoryId and categoryName.	New category information is received by the system.
2	Validation	The system checks whether the categoryId already exists.	Duplicate IDs are prevented.
3	Add Category	The new category object is inserted into the ArrayList.	Category list is updated with new data.
4	Display Categories	The system refreshes the category list shown to users.	Updated category information is displayed.
5	Select Category to Delete	Administrator specifies the categoryId to be removed.	Target category is identified.
6	Search Process	The system traverses the ArrayList to find the matching categoryId.	Category status is determined.
7	Delete Category	The matching category object is removed from the ArrayList.	Category list is updated.
8	Notification	The system informs whether the deletion process was successful or unsuccessful.	User feedback is provided.

An example of category data before and after the execution of add and delete operations is presented in Table 2.

Table 2. Example of Category Management Operations

Operation	Category ID	Category Name	Category List After Operation
Initial Data	C001	Electronics	Electronics, Clothing, Food
Initial Data	C002	Clothing	Electronics, Clothing, Food
Initial Data	C003	Food	Electronics, Clothing, Food
Add Category	C004	Books	Electronics, Clothing, Food, Books
Add Category	C005	Home Appliances	Electronics, Clothing, Food, Books, Home Appliances
Delete Category	C002	Clothing	Electronics, Food, Books, Home Appliances
Delete Category	C003	Food	Electronics, Books, Home Appliances

From an algorithmic perspective, adding a category to the end of an ArrayList generally requires **O(1)** time complexity under normal conditions. However, when the internal storage capacity is exceeded, the ArrayList automatically resizes its internal array, resulting in an occasional **O(n)** operation. On the other hand, deleting a category requires a search process with **O(n)** complexity because the system may need to examine multiple elements before locating the target categoryId. After deletion, the remaining elements are shifted to maintain the continuity of the list.

Table 3. Complexity Analysis of Category Operations

Operation	Data Structure	Average Time Complexity	Description
Add Category	ArrayList	O(1)	Inserts data at the end of the list.
Search Category	ArrayList	O(n)	Sequentially checks each element until a match is found.
Delete Category	ArrayList	O(n)	Includes searching and shifting remaining elements.
Display Categories	ArrayList	O(n)	Iterates through all categories for presentation.

B. Displaying Categories:

All listed categories can be easily displayed, including categoryId and categoryName, making it easier for admins to monitor and manage categories. The Displaying Categories feature is designed to present all available category information in a clear, organized, and easily understandable format. This functionality assists administrators in monitoring existing categories and ensures that category data remain accurate, up to date, and consistent with the products managed by the system.

In the proposed system, category data are stored in an ArrayList<Category> collection. Each category object contains essential attributes, namely categoryId and categoryName. The display process retrieves all category objects stored within the ArrayList and presents them sequentially to the administrator. Since ArrayList maintains the order of insertion, categories are displayed according to the sequence in which they were added, unless additional sorting mechanisms are implemented.

The primary objective of this feature is to provide administrators with a comprehensive overview of all registered categories. By viewing the complete list, administrators can

quickly identify whether a category already exists before adding a new one, reducing the risk of duplicate category creation. Furthermore, the display functionality helps administrators determine which categories may need to be updated or removed due to changes in organizational requirements or business operations.

The category display process typically begins when the administrator selects the "View Categories" menu from the system interface. The system then checks whether the ArrayList contains any category data. If categories are available, the system iterates through each element in the collection using a looping structure such as a for-each loop. During each iteration, the values of categoryId and categoryName are extracted and formatted for presentation on the screen. If no categories exist, the system provides an informative message indicating that the category list is empty.

From a usability perspective, displaying category information improves administrative efficiency by enabling rapid access to classification data without requiring database queries or manual record searches. This capability becomes increasingly important as the number of categories grows over time. A well-structured category display also supports other management functions, such as editing, deleting, and assigning products to appropriate categories.

The overall process of displaying category information is illustrated in Table 4.

Table 4. Process of Displaying Categories

Step	Process	System Activity	Output
1	Access Category Menu	Administrator selects the "Display Categories" option.	Display request is initiated.
2	Check Data Availability	The system verifies whether the ArrayList contains category objects.	Data existence status is determined.
3	Iterate Through Categories	The system traverses each element stored in the ArrayList.	Individual category records are retrieved.
4	Extract Category Attributes	The values of categoryId and categoryName are obtained from each category object.	Category details are prepared for presentation.
5	Present Information	The category information is displayed in tabular or list format.	Administrators can review all categories.
6	Handle Empty List Condition	If no categories are found, the system displays a notification message.	Users receive appropriate feedback.

An example of the category display output is shown below.

Tabl5. Category Display

Category ID	Category Name
C001	Electronics
C002	Clothing
C003	Food
C004	Books
C005	Home Appliances

The implementation of this feature requires traversing all elements stored within the ArrayList. Consequently, the time complexity of the display operation is **O(n)**, where **n** represents the total number of categories stored in the system. The complexity is considered efficient and acceptable because each category record must be accessed at least once to be presented to the administrator.

C. Updating Category Data:

Admin can update specific data based on category type, such as electronic warranty, clothing size, and food expiration date, which makes the system flexible and adaptable to product changes.

D. Dynamic Management with ArrayList:

ArrayList is used to store categories, allowing easy and flexible data changes. This provides advantages in the scale and flexibility of the system.

E. Inheritance Benefits:

By using inheritance, the code is more structured and easy to extend. Each product category has its own derived class, making it easy to add new categories.

F. Challenges:

The main challenge is data validation to ensure product information updates are done correctly. Integration with database and user interface creation are also required for system scalability and reliability in real applications.

Case studies along with code examples from this research are as follows

```
// Kelas induk untuk kategori produk
class Category {
    private String categoryName;
    private String description;

    // Constructor
    public Category(String categoryName, String description) {
        this.categoryName = categoryName;
        this.description = description;
    }

    // Getter dan Setter
    public String getCategoryName() {
        return categoryName;
    }

    public void setCategoryName(String categoryName) {
        this.categoryName = categoryName;
    }

    public String getDescription() {
        return description;
    }

    public void setDescription(String description) {
        this.description = description;
    }

    // Method umum yang dapat digunakan oleh semua kategori
    public void displayCategoryInfo() {
        System.out.println("Category: " + categoryName);
        System.out.println("Description: " + description);
    }
}
```

```
// Subclass untuk kategori produk Elektronik
class ElectronicsCategory extends Category {
    private int warrantyPeriod; // dalam bulan

    // Constructor
    public ElectronicsCategory(String categoryName, String
description, int warrantyPeriod) {
        super(categoryName, description);  // Memanggil
constructor induk
        this.warrantyPeriod = warrantyPeriod;
    }

    // Getter dan Setter
    public int getWarrantyPeriod() {
        return warrantyPeriod;
    }

    public void setWarrantyPeriod(int warrantyPeriod) {
        this.warrantyPeriod = warrantyPeriod;
    }
```

V. CONCLUSION

This article discusses how to build a product category system for a shopping application using the concept of inheritance in the Java programming language. By utilizing inheritance, developers can design a more organized and flexible category structure, where each product category can inherit basic properties from the parent class, such as category name, description, and other common attributes. This system also allows each category to have additional attributes or methods that are specific to the type of product in question. This approach not only makes the application more manageable, but also supports object-oriented programming (OOP) principles such as encapsulation, polymorphism, and abstraction. Thus, the development of shopping applications becomes more efficient, can be easily extended, and is easier to maintain or modify in the future, making it a scalable and durable solution.

REFERENCES

[1] A. S. Reiffer, J. Kübler, M. Kagerbauer, and P. Vortisch, “Agent-based model of last-mile parcel deliveries and travel demand incorporating online shopping behavior,” *Res. Transp. Econ.*, vol. 102, p. 101368, Dec. 2023, doi: 10.1016/J.RETREC.2023.101368.

[2] M. Wu, O. C. Demirag, W. Xue, and M. Xu, “Retail category management under shelf-space dependent demand: The effectiveness of category captainship,” *Int. J. Prod. Econ.*, vol. 276, p. 109365, Oct. 2024, doi: 10.1016/J.IJPE.2024.109365.

[3] I. Elyashevich, V. Sergeev, V. Dybskaya, and A. Ivanova, “Category management for the operational resource procurement,” *J. Innov. Knowl.*, vol. 9, no. 3, p. 100507, Jul. 2024, doi: 10.1016/J.JIK.2024.100507.

[4] K. M. Hosny, A. M. Khalid, W. Said, M. Elmezain, and S. Mirjalili, “A novel metaheuristic based on object-oriented programming concepts for engineering optimization,” *Alexandria Eng. J.*, vol. 98, pp. 221–248, Jul. 2024, doi: 10.1016/J.AEJ.2024.04.060.

[5] M. Gatterer, H. Leonhardt, K. Salhofer, and U. Morawetz, “The legacy of partible

inheritance on farmland fragmentation: Evidence from Austria," *Land use policy*, vol. 140, p. 107110, May 2024, doi: 10.1016/J.LANDUSEPOL.2024.107110.

[6] L. Aumann, H. Gasteiger, and R. M. Puca, "Early childhood teachers' feedback in natural mathematical learning situations: Development and validation of a detailed category system," *Acta Psychol. (Amst).*, vol. 244, p. 104175, Apr. 2024, doi: 10.1016/J.ACTPSY.2024.104175.

[7] Q. Luo, Q. Deng, G. Gong, X. Guo, and X. Liu, "A distributed flexible job shop scheduling problem considering worker arrangement using an improved memetic algorithm [Expert Systems with Applications 207 (2022) 117984]," *Expert Syst. Appl.*, vol. 239, p. 120161, Apr. 2024, doi: 10.1016/J.ESWA.2023.120161.

[8] D. A. Akinpelu, O. A. Adekoya, P. O. Oladoye, C. C. Ogbaga, and J. A. Okolie, "Machine learning applications in biomass pyrolysis: From biorefinery to end-of-life product management," *Digit. Chem. Eng.*, vol. 8, p. 100103, Sep. 2023, doi: 10.1016/J.DCHE.2023.100103.

[9] K. N. Rahman, M. W. Hridoy, M. Mizanur Rahman, M. R. Islam, and S. Banik, "Highly secured and effective management of app-based online voting system using RSA encryption and decryption," *Heliyon*, vol. 10, no. 3, p. e25373, Feb. 2024, doi: 10.1016/J.HELIYON.2024.E25373.

[10] N. Hajimirza Amin, A. Etemad, and A. Abdalisousan, "Data-driven performance analysis of an active chilled beam air conditioning system: A machine learning approach for energy efficiency and predictive maintenance," *Results Eng.*, vol. 23, p. 102747, Sep. 2024, doi: 10.1016/J.RINENG.2024.102747.

[11] A. C. B. Trude *et al.*, "'I Don't Want an App to Do the Work for Me': A Qualitative Study on the Perception of Online Grocery Shopping From Small Food Retailers," *J. Acad. Nutr. Diet.*, vol. 124, no. 7, pp. 804–822, Jul. 2024, doi: 10.1016/J.JAND.2023.12.005.

[12] C. Zhou, D. Xu, and Z. Wang, "Conversion and fusion method of multi-source and different populations maintainability prior data," *Heliyon*, vol. 9, no. 11, p. e21208, Nov. 2023, doi: 10.1016/J.HELIYON.2023.E21208.

[13] L. A. Huwaida *et al.*, "Generation Z and Indonesian Social Commerce: Unraveling key drivers of their shopping decisions," *J. Open Innov. Technol. Mark. Complex.*, vol. 10, no. 2, p. 100256, Jun. 2024, doi: 10.1016/J.JOITMC.2024.100256.

[14] S. Vafainia, R. P. Rooderkerk, E. Breugelmans, and T. H. A. Bijmolt, "Decision support system development for store flyer space allocation: Leveraging own- and cross-category sales effects," *Int. J. Res. Mark.*, Jul. 2024, doi: 10.1016/J.IJRESMAR.2024.07.002.

[15] Z. Li, C. Guo, L. Wang, and W. Zeng, "A multi-objective co-optimization method of controller parameters for the overall system of small pressurized water reactor," *Energy*, vol. 308, p. 132888, Nov. 2024, doi: 10.1016/J.ENERGY.2024.132888.

[16] F. Pournaropoulos, A. Patras, C. D. Antonopoulos, N. Bellas, and S. Lalis, "Fluidity: Providing flexible deployment and adaptation policy experimentation for serverless and distributed applications spanning cloud–edge–mobile environments," *Futur. Gener. Comput. Syst.*, vol. 157, pp. 210–225, Aug. 2024, doi: 10.1016/J.FUTURE.2024.03.031.

[17] V. N. Huynh Anh, "An Architectural View Model for Designing and Implementing Microservices-based Systems: Use Case in FinTech," *Procedia Comput. Sci.*, vol. 237, pp. 667–674, Jan. 2024, doi: 10.1016/J.PROCS.2024.05.152.

[18] H. Wu, F. Wu, Z. Li, X. Gao, X. Wu, and G. Bao, "Considering scale effects in water quality analysis to enhance the precision of influencing factor response analysis," *Ecol. Indic.*, vol. 163, p. 112091, Jun. 2024, doi: 10.1016/J.ECOLIND.2024.112091.

[19] I. Graessler, J. Hentze, and A. Poehler, "Self-organizing production systems: Implications for product design," *Procedia CIRP*, vol. 79, pp. 546–550, Jan. 2019,

doi: 10.1016/J.PROCIR.2019.02.092.

[20] J. M. Pearson, "A Review: Breeding behavior and management strategies for improving reproductive efficiency in bulls," *Anim. Reprod. Sci.*, p. 107669, Dec. 2024, doi: 10.1016/J.ANIREPROSCI.2024.107669.