Generics and Collections in Action for

Master of Science in Information Technology

Software Design and Programming

Joseph DiBiasi, Estell Moore, James McKenna, Anthony Martuscelli

University of Denver College of Professional Studies

10/10/2025

Faculty: Nirav Shah, M.S.

Director: Cathie Wilson, M.S.

Dean: Michael J. McGuire, MLS

Abstract

In this portion of our assignment, the purpose of our MonthApp class was to write an implementation that not only used generics but also showed safe programming and reusability. In order to demonstrate this, specific use cases were needed to store and manage all the instances for our MonthApp class we had to make. This implementation would require 12 instances of the month class that would be stored using various objects that could be stored within an array, array list, hashmap and so on. The code reusability not only showed us the use cases and strengths of type safing and generics but also shows the challenges that come with trying to write code that can be used for the sake of modularity.

CONTENTS

Introduction

Generics are an incredibly useful feature that not only allow users to write code, but also write safe code. Allowing all sorts of components that are written within Java to be used in a safe way but also can help with debugging and running code. Generics can be demonstrated for reducing runtime errors and mismatches at compile time by explicitly using type specified parameters for our interfaces, classes, methods, hash maps, arrays, and more. For our assignment our team had to code a Calendar Month app that would showcase the powerful use cases for generics and how to demonstrate reusable code in a safe and efficient manner. Our problem statement was needed to showcase our class for our months in multiple formats by representing months through integers and strings. This will be discussed and broken down in our implementation and decisions section on how we approached this.

Implementation Decisions

The month-app is a maven project designed to show off generics and collections in action. Code was split between two classes. The Month class contained only one field, month, which used a generic T to allow the class flexibility in how the month was stored; the month of January could be stored as is in a String or it could be the Integer value of 1. This class was then used by the MonthApp class to demonstrate collections.

MonthApp contained the main method. When run, the main method instantiates the collections. This involves creating two arrays of size 12, two ArrayLists, and a single HashMap. These were then passed into the populateMonthData method. This method starts out by creating a String array called monthNames filled with the written months of the year in

sequential order. The order is important because all collections are populated by the same initialization variable. The String value of month is found by using the initialization variable as an index value to access monthNames; the Integer value is found by taking that same variable and adding one to accommodate for zero-based arrays. Arrays add these values directly using the initialization variable to track index. ArrayLists are added directly to the end of the lists. HashMaps add the Integer as the key and the String as the value. This data is then printed out to the user in the readOutMonthMethod. Arrays and ArrayLists use a For Loop to print out the month Integer and String values. HashMaps are unordered by default, they only care about the uniqueness of their keys. The solution was to stream the collection and sort the data manually before terminating the stream with a ForEach Loop that reads out the month Integer and String value in the proper order.

Challenges

Some of the primary challenges encountered when implementing collections and generics involved determining the most effective way to store and utilize the data. Arrays and lists are well suited for maintaining elements of a single data type, while maps are better suited for associating elements of different data types. In this implementation, the objective was to establish a mapping between a String and an Integer. However, initiating the mapping process required consideration of how string data would be handled and linked to its numeric value. As a result, much of the decision-making for structuring the collections was centered on the logical design within the for loops.

While the concept of generics within the Month class was relatively straightforward to understand, applying that logic in the MonthApp class, particularly when determining which data types to use for mapping, proved to be more challenging.

Conclusion

Overall, this week provided us with yet another opportunity to understand code structures. Specifically, it was hands-on work to better understand how generics and collections work to improve flexibility and efficiency in Java. By implementing testing with different types of collections (arrays, ArrayLists, and HashMaps), our team was able to see how each of these structures handles data differently, and moreover why it is important to choose the correct one for the task. Using generics within the Month class also showed how the use of well structured code can be adaptable and even reusable. This leads to reduced redundancy and allows for developers to make cleaner choices during coding.

Beyond the actual technical implementation, this week offered yet another chance for group collaboration. The collaboration this week offered insight into how even small implementation decisions can change and impact the overall group and flow of a program. This week, we were reminded to consider concepts such as structure, naming conventions, and data management. All areas which can be overlooked in the early stages of development. As a team, we learned to create a balance between simplicity and scalability, allowing us to focus on code that not only works, but can be maintained. Additionally, these decisions create code that allows for handoff and understanding to those who may not have been present during earlier stages of development.

This week's exercise also highlighted the connection between concepts found in generics and collections, and larger programming principles such as reusability. Seeing these concepts in action helped reinforce some of the concepts we learned as we were able to actually implement them. Going forward, the lessons learned this week will more than likely shape how we make decisions regarding complex data structures.

```
[INFO] Running month.MonthAppTest
Print out using Arrays:
1 = January
2 = February
3 = March
4 = April
5 = May
6 = June
7 = July
8 = August
9 = September
10 = October
11 = November
12 = December

Print out using ArrayLists:
1 = January
2 = February
3 = March
4 = April
5 = May
6 = June
7 = July
8 = August
9 = September
10 = October
11 = November
12 = December

Print out using HashMap:
1 = January
2 = February
3 = March
4 = April
5 = May
6 = June
7 = July
8 = August
9 = September
10 = October
11 = November
12 = December
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.067 s -- in month.MonthAppTest
[INFO] Running month.MonthTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.010 s -- in month.MonthTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 10, Failures: 0, Errors: 0, Skipped: 0
[INFO]
```

Figure 1. Month Print Out and Test Results