SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)
DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITL306 **DATE:2/11/2023**
COURSE NAME: Programing Laboratory 1 (Advanced Java) **CLASS: I1-Batch1**

NAME: Ayush Vinod Upadhyay
ROLL NO: I025
SAP ID: 60003220131
BRANCH: Information Technology

EXPERIMENT NO. 02

CO/LO: CO1- Modify the behaviour of methods, classes, and interfaces at runtime.

AIM / OBJECTIVE: To implement different collection types

Problem Statements:1. Write a Java program that creates 2 lists, 1 for integer and other for strings. Define a generic method to display the elements of both lists using arrays with the use of for-each loop.

CODE:

```
import java.util.*;

class Gen1 {
    static <T> void display(ArrayList<T> list) {
        for (T i : list) {
            System.out.print(i + " ");
        }
    }
    public static void main(String args[]) {
        ArrayList<Integer> l1 = new ArrayList<Integer>();
        l1.add(1);
        l1.add(2);
        l1.add(3);
        ArrayList<String> l2 = new ArrayList<String>();
        l2.add("a");
        l2.add("b");
        l2.add("c");
        display(l1);
        System.out.println();
        display(l2);
    }
}
```

COURSE CODE: DJS22ITL306 **DATE:2/11/2023**
COURSE NAME: Programing Laboratory 1 (Advanced Java) **CLASS: I1-Batch1**

OUTPUT:

```
PS C:\Users\ayush\Desktop\SEM 3\Adv Java>
} ; if ($?) { java Gen1 }
1 2 3
a b c
```

2. Write a simple generic version of method isEqualTo that compares its two arguments with the equals method and returns true if they're equal and false otherwise. Use this generic method in a program that calls isEqualTo with a variety of built-in types, such as Object or Integer. What result do you get when you attempt to run this program?

CODE:

```
import java.util.*;

class Gen2 {
    static <T> void isEqualTo(T t1, T t2) {
        if (t1.equals(t2)) {
            System.out.println("Equal");
        } else {
            System.out.println("Not Equal");
        }
    }

    public static void main(String args[]) {
        Object o1 = new Object();
        Object o2 = new Object();
        isEqualTo(o1, o2);
        Integer i1 = 20;
        Integer i2 = 200;
        isEqualTo(i1, i2);
        String s1 = "string";
        String s2 = "string";
        isEqualTo(s1, s2);
    }
}
```

OUTPUT:

```
PS C:\Users\ayush\Desktop\SEM 3\Adv Java>
} ; if ($?) { java Gen2 }
Not Equal
Not Equal
Equal
```

COURSE CODE: DJS22ITL306 **DATE:2/11/2023**
COURSE NAME: Programing Laboratory 1 (Advanced Java) **CLASS: I1-Batch1**

3. Write a generic method Sort based on the sort program. Write a test program that inputs, sorts and outputs an Integer array and a Float array. [Hint: Use > in the type-parameter section for method Sort, so that you can use method compareTo to compare the objects of the type that T represents.]

CODE:

```
import java.util.*;


class Gen3 {
    static <T extends Comparable<T>> void sort(T[] arr) {
        int n = arr.length;
        T temp;
        for (int i = 0; i < n; ++i) {
            for (int j = i + 1; j < n; ++j) {
                if (arr[i].compareTo(arr[j]) == 1) {
                    temp = arr[j];
                    arr[j] = arr[i];
                    arr[i] = temp;
                }
            }
        }
        for (T i : arr) {
            System.out.print(i + " ");
        }
        System.out.println();
    }

    public static void main(String args[]) {
        Integer[] int_arr = { 5, 1, 3, 2, 4 };
        Float[] float_arr = { 5.1f, 2.2f, 1.2f, 1f, 5.9f };
        sort(int_arr);
        sort(float_arr);
    }
}
```

OUTPUT:

```
PS C:\Users\ayush\Desktop\SEM 3\Adv Java>
} ; if ($?) { java Gen3 }
1 2 3 4 5
1.0 1.2 2.2 5.1 5.9
```

SHRI VILEPARLE KELAVANI MANDAL'S
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)
NAAC ACCREDITED with "A" GRADE (CGPA : 3.18)
DEPARTMENT OF INFORMATION TECHNOLOGY

COURSE CODE: DJS22ITL306 **DATE:2/11/2023**
COURSE NAME: Programing Laboratory 1 (Advanced Java) **CLASS: I1-Batch1**

OBSERVATION:

Q. What is the need for generic types. List various types of generics.

Generics in programming provide a way to create flexible and reusable code by allowing types to be specified later. This enables you to write code that can work with different data types without sacrificing type safety. The need for generic types arises in situations where you want to write code that is not tied to a specific data type, but rather can be used with a variety of data types. Some common reasons for using generics include:

1. **Code Reusability:** Generics allow you to write functions and classes that can work with different data types, promoting code reuse across different parts of your program.
2. **Type Safety:** Generics provide compile-time type checking, ensuring that the code is type-safe. This helps catch type-related errors at compile time rather than at runtime.
3. **Abstraction:** Generics enable the creation of abstract data structures and algorithms that can operate on a variety of data types without being tied to any specific one.
4. **Performance:** Generics can improve performance by allowing the use of specialized algorithms and data structures for different types, while maintaining a single, generic implementation.
5. **Flexibility:** With generics, you can write more flexible and adaptable code, making it easier to extend and maintain.

Various types of generics exist, depending on the programming language. Some common types include:

1. **Generic Classes:** Classes that can work with different data types. Examples include Java's `ArrayList<T>` or C#'s `List<T>`.
2. **Generic Functions/Methods:** Functions or methods that can operate on parameters of different types. For example, a generic sorting function in C++.
3. **Generic Interfaces:** Interfaces that can be implemented by different classes, where the interface is parameterized by a type. Java's `Comparable<T>` is an example.
4. **Generic Constraints:** Some languages allow you to impose constraints on generic types, specifying that the type must have certain methods or properties. This adds additional type safety. C# uses constraints, for instance.
5. **Wildcard Generics:** Some languages provide constructs like wildcards that allow for more flexible type specifications. Java's wildcard (`?`) is an example.
6. **Generic Enums:** In some languages, you can create generic enums where the values can be of different types. This is less common but can be found in languages like C#.
7. **Generic Delegates:** Some languages allow you to define generic delegates, enabling the creation of functions with different parameter and return types. C# supports generic delegates.

COURSE CODE: DJS22ITL306 **DATE:2/11/2023**
COURSE NAME: Programing Laboratory 1 (Advanced Java) **CLASS: I1-Batch1**

The specific types and features available depend on the programming language being used. The use of generics enhances the expressiveness, flexibility, and safety of code, making it a valuable feature in many modern programming languages.

CONCLUSION:

In this experiment we learnt how to use generics in java. Generics make code shorter, reusable, robust, and increase performance in modern programming. Generics is a fundamental part of programming and can be used to make our code more generalized.