PRACTICAL 1 – JAVA GENERICS (DETAILED VERSION)

AIM

To study and implement Java Generics by demonstrating: 1. Generic Class 2. Generic Method 3. Wildcards in Java Generics

SOFTWARE REQUIREMENTS

• Operating System: Windows / Linux
• JDK: Version 7 or 8
• NetBeans IDE: Version 7.x

THEORY

Java Generics were introduced to provide type safety and reduce runtime errors. Generics allow classes, methods, and interfaces to operate on different data types without explicit type casting. Type checking is done at compile time.

PART A – GENERIC CLASS

A Generic Class is a class that can work with different data types using a type parameter. Syntax: class ClassName

```
package genericsdemo;

class Box<T> {
    private T value;

    public void set(T value) {
        this.value = value;
    }

    public T get() {
        return value;
    }
}

public class GenericClassDemo {
    public static void main(String[] args) {
        Box<Integer> intBox = new Box<>();
        intBox.set(100);

        Box<String> strBox = new Box<>();
        strBox.set("Advanced Java");

        System.out.println("Integer Value: " + intBox.get());
        System.out.println("String Value: " + strBox.get());
    }
}
```

OUTPUT: Integer Value: 100 String Value: Advanced Java

PART B – GENERIC METHOD

A Generic Method defines its own type parameter and can accept arguments of any type.

```
package genericsdemo;

class Demo {
    public <T> void display(T data) {
        System.out.println(data);
    }
}

public class GenericMethodDemo {
    public static void main(String[] args) {
        Demo d = new Demo();
        d.display("Java");
        d.display(10);
        d.display(45.5);
    }
}
```

PART C – WILDCARDS

Wildcards provide flexibility in generics. • Unbounded wildcard • Upper bounded wildcard • Lower bounded wildcard

```
package genericsdemo;

import java.util.*;

public class WildcardDemo {

    static void printList(List<?> list) {
        for(Object o : list)
            System.out.println(o);
    }

    public static void main(String[] args) {
        List<Integer> list = Arrays.asList(1,2,3);
        printList(list);
    }
}
```

NETBEANS 7 – STEP BY STEP EXECUTION

1. Open NetBeans IDE 7
2. File → New Project → Java → Java Application
3. Project Name: GenericsDemo
4. Uncheck 'Create Main Class'
5. Click Finish
6. Right-click Source Packages → New → Java Package → genericsdemo
7. Create Java Classes as shown above
8. Paste code in respective classes
9. Press Shift + F6 to run each program

RESULT

Thus, Java Generics were successfully implemented using Generic Class, Generic Method, and Wildcards.