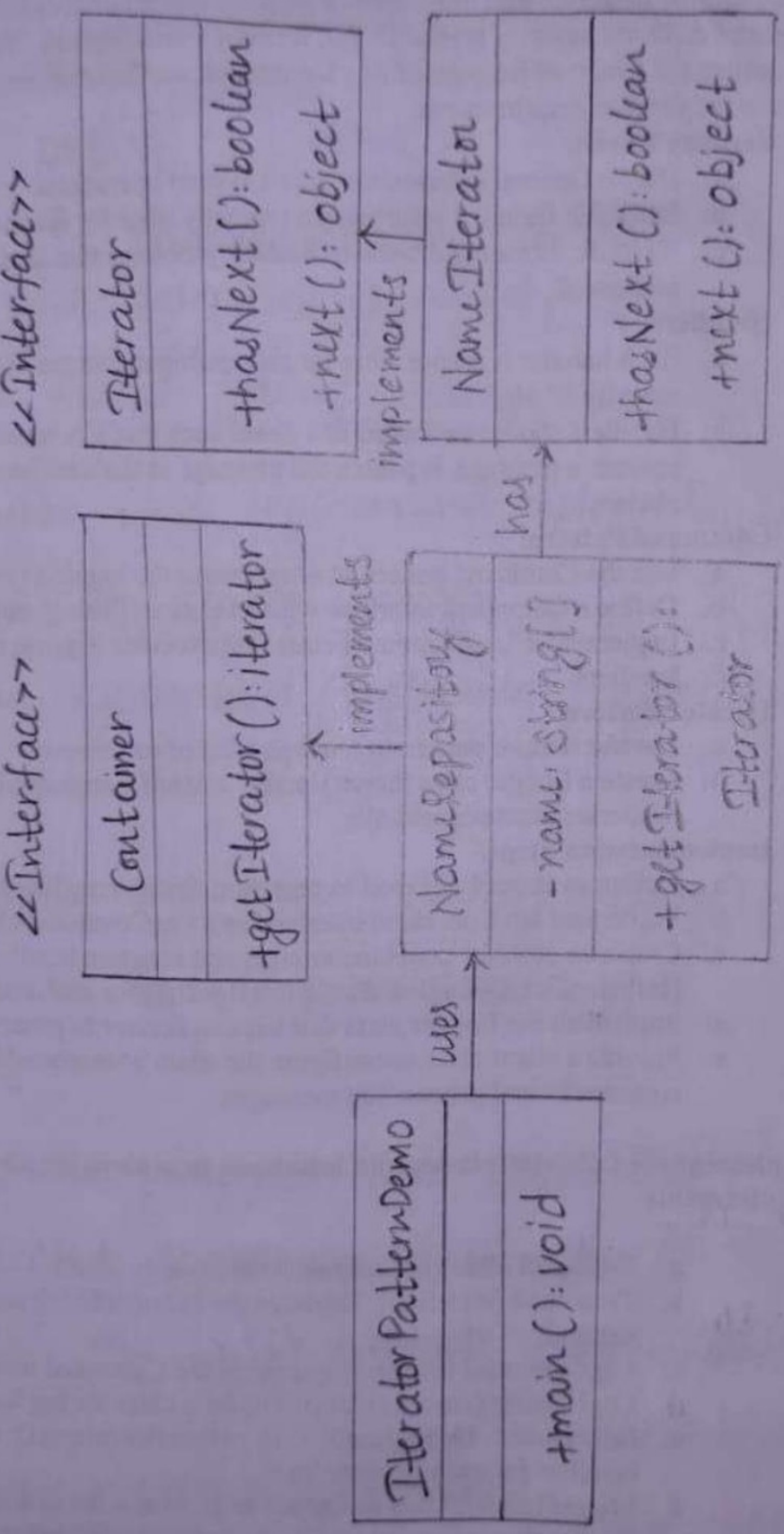
2) Draw the UML Relationship Diagram for Iterator Design Pattern for customized Scenarios.



Experiment#		Student ID	
Date		Student Name	

Procedure/Program:

```
public enum LogLevel {
    INFO,
    DEBUG,
    ERROR
} public interface Command {
void execute (String message);
} public class LogCommand implements Command {
private LogHandler handler;
public LogCommand (LogHandler handler) {
  this.handler = handler;
} public void execute (String message) {
  handler.handleRequest (message);
    }
} public abstract class LogHandler {
protected LogHandler nextHandler;
public void setNextHandler (LogHandler nextHandler)
{
  this.nextHandler = nextHandler;
```

```
}
public abstract void handleRequest(String
                                   message);
}
public class InfoHandler extends LogHandler{
public void handleRequest(String message){
if(message.contains(LogLevel.INFO.name())){
System.out.println("INFO Handler: " + message);
} else if (nextHandler != null) {
nextHandler.handleRequest(message);
    }
  }
} public class DebugHandler extends LogHandler
{
public void handleRequest(String message){
if(message.contains(LogLevel.DEBUG.name())){
System.out.println("DEBUG Handler: " + message);
} else if (next Handler != null) {
  nextHandler.(message);
```

```
        }
    }
} public class ErrorHandler extends LogHandler {
public void handleRequest(String message) {
if (message.contains(LogLevel.ERROR.name())) {
System.out.println("ERROR Handler: " + message);
} else if (nextHandler != null) {
nextHandler.handleRequest(message);
        }
    }
}
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
public class Logger {
 private List<Command> commands=new
                    ArrayList<>();
public void addCommand(Command command) {
  commands.add(command);
} public void processCommands() {
```

```
Iterator<Command> iterator = commands.iterator();
 while (iterator.hasNext()) {
 Command command = iterator.next();
 command.execute("Log message of severity level,
        INFO, DEBUG or ERROR");
    }
  }
}
```

✓ **Data and Results:**

INFO Handler: Log message of severity level INFO, DEBUG or ERROR

DEBUG Handler: Log message of severity level INFO, DEBUG or ERROR

ERROR Handler: Log message of severity level INFO, DEBUG or ERROR

✓ **Analysis and Inferences:**

The integration of the Chain of Responsibility, command and Iterator patterns in the logging system ensures that it is well-organized, flexible and easy to maintain, allowing for effective handling of various log levels and logging operations.

VIVA-VOCE Questions (In-Lab):

1) State at which situation that we need Chain of Responsibility Design Patter

The Chain of Responsibility design pattern is particularly useful in situations where you need to process a request through a series of handlers, each of which may or maynot handle the request.

2) Discuss the Pros and Cons of Iterator Design Pattern.

Iterator Design pattern provides valuable abstraction and flexibility for traversing collections, but it may introduce additional complexity and overhead in certain scenarios.

3) Discuss the Pros and Cons of Command Design Pattern

Command Design Pattern provides significant advantages in terms of flexibility, decoupling and support for complex operations such as undo. It may introduce additional complexity and overhead, particularly for simple requests.

4) Which pattern is used to hide the implementation details while traversing over an ArrayList?

A: The **Iterator Pattern** is used to hide implementation details while traversing an `ArrayList`. It allows sequential access to elements without exposing the collection's internal structure.

		Student ID	
Experiment#		Student Name	
Date			

✓ **Data and Results:**

INFO Handler: Log message of severity level INFO, DEBUG or ERROR

DEBUG Handler: Log message of severity level INFO, DEBUG or ERROR

ERROR Handler: Log message of severity level INFO, DEBUG or ERROR

✓ **Analysis and Inferences:**

The integration of the chain of responsibility, Command and Iterator patterns in the logging system ensures that it is well-organized, flexible and easy to maintain.

Evaluator Remark (if Any):	Marks Secured: ____out of 50
Evaluator MUST ask Viva-voce prior to signing and posting marks for each experiment.	Signature of the Evaluator with Date