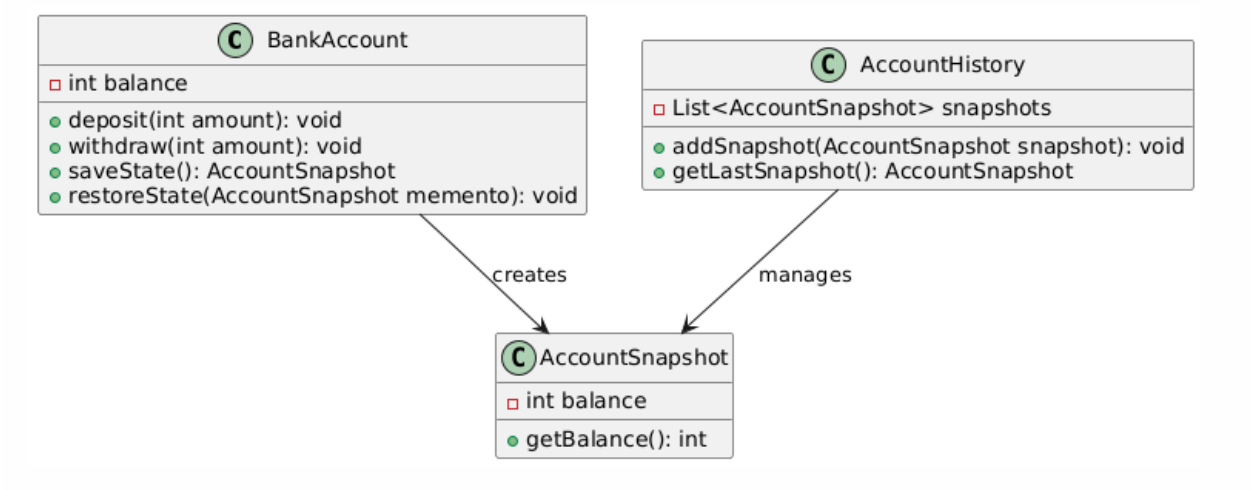
Laiba binte tahir
FA21-BSE-019

Task 4: Bank Account Snapshot

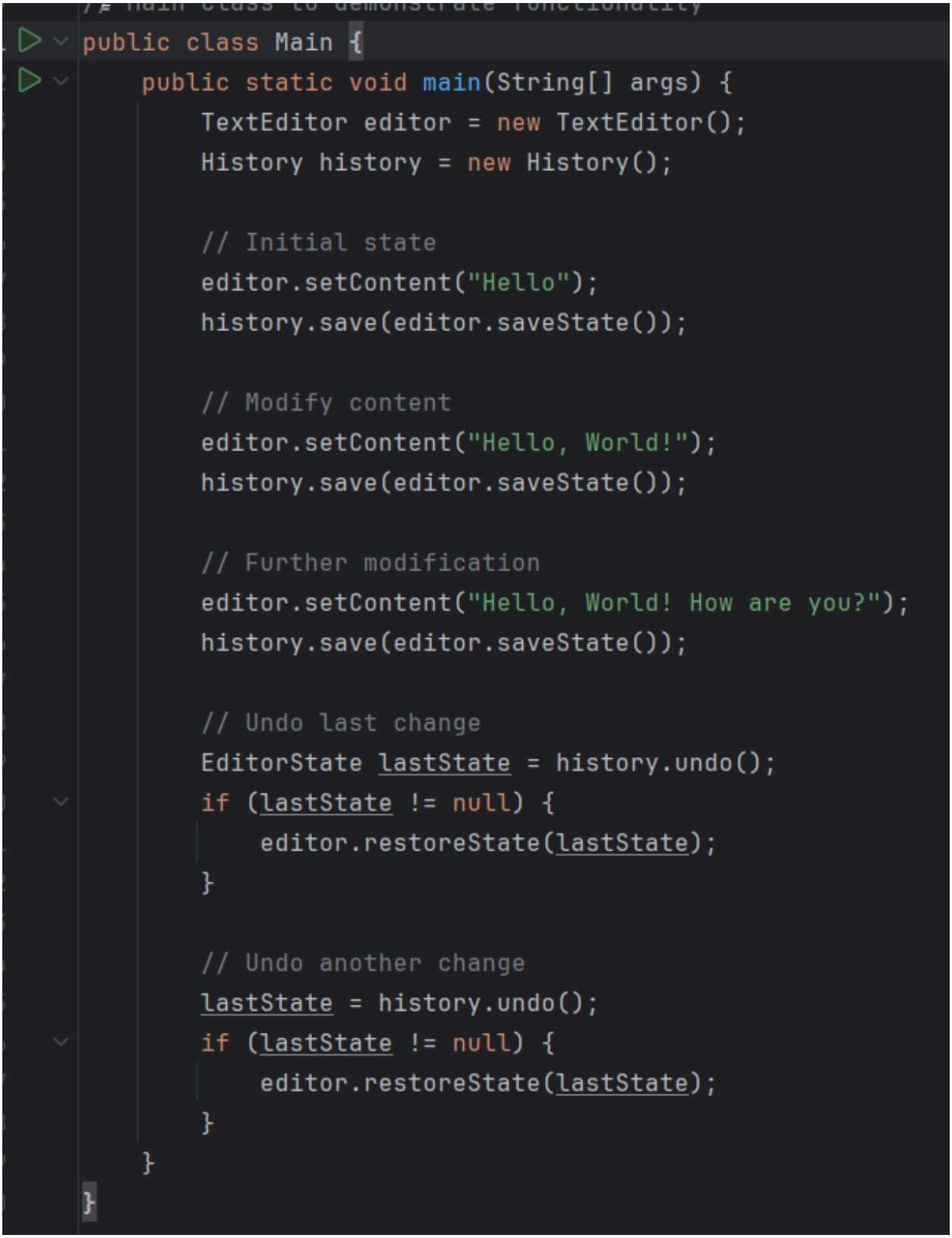
```java
// Originator
class BankAccount {
    private int balance;

    public BankAccount(int initialBalance) {
        this.balance = initialBalance;
    }

    public void deposit(int amount) {
        balance += amount;
        System.out.println("Deposited: " + amount + ", Current Balance: " + balance);
    }

    public void withdraw(int amount) {
        if (amount > balance) {
            System.out.println("Insufficient balance for withdrawal!");
        } else {
            balance -= amount;
            System.out.println("Withdrawn: " + amount + ", Current Balance: " + balance);
        }
    }

    public AccountSnapshot saveState() {
        return new AccountSnapshot(balance);
    }

    public void restoreState(AccountSnapshot memento) {
        this.balance = memento.getBalance();
        System.out.println("Restored Balance: " + balance);
    }
}
```

```java
// Memento
class AccountSnapshot {
    private final int balance;

    public AccountSnapshot(int balance) {
        this.balance = balance;
    }

    public int getBalance() {
        return balance;
    }
}

// Caretaker
class AccountHistory {
    private final List<AccountSnapshot> snapshots = new ArrayList<>();

    public void addSnapshot(AccountSnapshot snapshot) {
        snapshots.add(snapshot);
    }

    public AccountSnapshot getLastSnapshot() {
        if (!snapshots.isEmpty()) {
            return snapshots.remove( index: snapshots.size() - 1);
        }
        System.out.println("No snapshots available!");
        return null;
    }
}
```

```java
// Main Class to demonstrate functionality
public class Main {
    public static void main(String[] args) {
        BankAccount account = new BankAccount( initialBalance: 1000);
        AccountHistory history = new AccountHistory();

        account.deposit( amount: 500);
        history.addSnapshot(account.saveState());

        account.withdraw( amount: 200);
        history.addSnapshot(account.saveState());

        account.withdraw( amount: 1500); // Should fail due to insufficient balance

        // Undo last transaction
        AccountSnapshot lastSnapshot = history.getLastSnapshot();
        if (lastSnapshot != null) {
            account.restoreState(lastSnapshot);
        }

        // Undo another transaction
        lastSnapshot = history.getLastSnapshot();
        if (lastSnapshot != null) {
            account.restoreState(lastSnapshot);
        }
    }
}
```



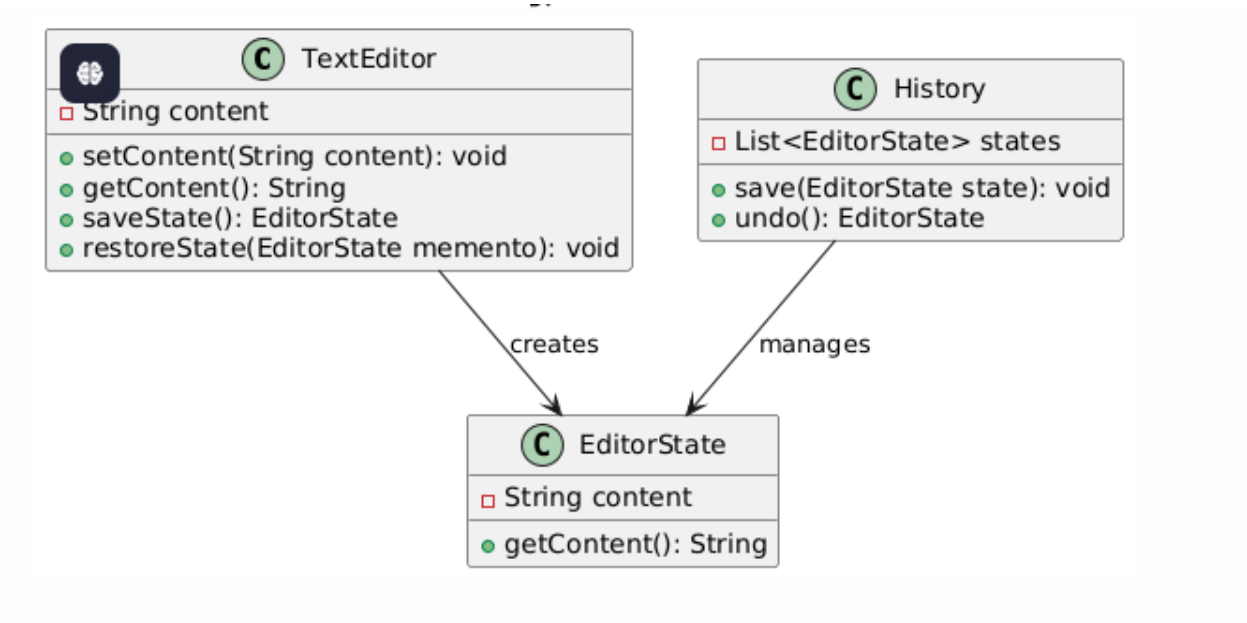
Task 3: Text Editor Undo



```
// Main class to demonstrate functionality
public class Main {
    public static void main(String[] args) {
        TextEditor editor = new TextEditor();
        History history = new History();

        // Initial state
        editor.setContent("Hello");
        history.save(editor.saveState());

        // Modify content
        editor.setContent("Hello, World!");
        history.save(editor.saveState());

        // Further modification
        editor.setContent("Hello, World! How are you?");
        history.save(editor.saveState());

        // Undo last change
        EditorState lastState = history.undo();
        if (lastState != null) {
            editor.restoreState(lastState);
        }

        // Undo another change
        lastState = history.undo();
        if (lastState != null) {
            editor.restoreState(lastState);
        }
    }
}
```

```java
package org.example;
import java.util.ArrayList;
import java.util.List;
import java.util.ArrayList;
import java.util.List;

// Originator
class TextEditor {
    private String content;

    public void setContent(String content) {
        this.content = content;
        System.out.println("Set Content: " + content);
    }

    public String getContent() {
        return content;
    }

    public EditorState saveState() {
        return new EditorState(content);
    }

    public void restoreState(EditorState memento) {
        this.content = memento.getContent();
        System.out.println("Restored Content: " + content);
    }
}
```

```java
// Memento
class EditorState {
    private final String content;

    public EditorState(String content) {
        this.content = content;
    }

    public String getContent() {
        return content;
    }
}

// Caretaker
class History {
    private final List<EditorState> states = new ArrayList<>();

    public void save(EditorState state) {
        states.add(state);
    }

    public EditorState undo() {
        if (!states.isEmpty()) {
            return states.remove( index: states.size() - 1);
        }
        System.out.println("No states to undo!");
        return null;
    }
}
```

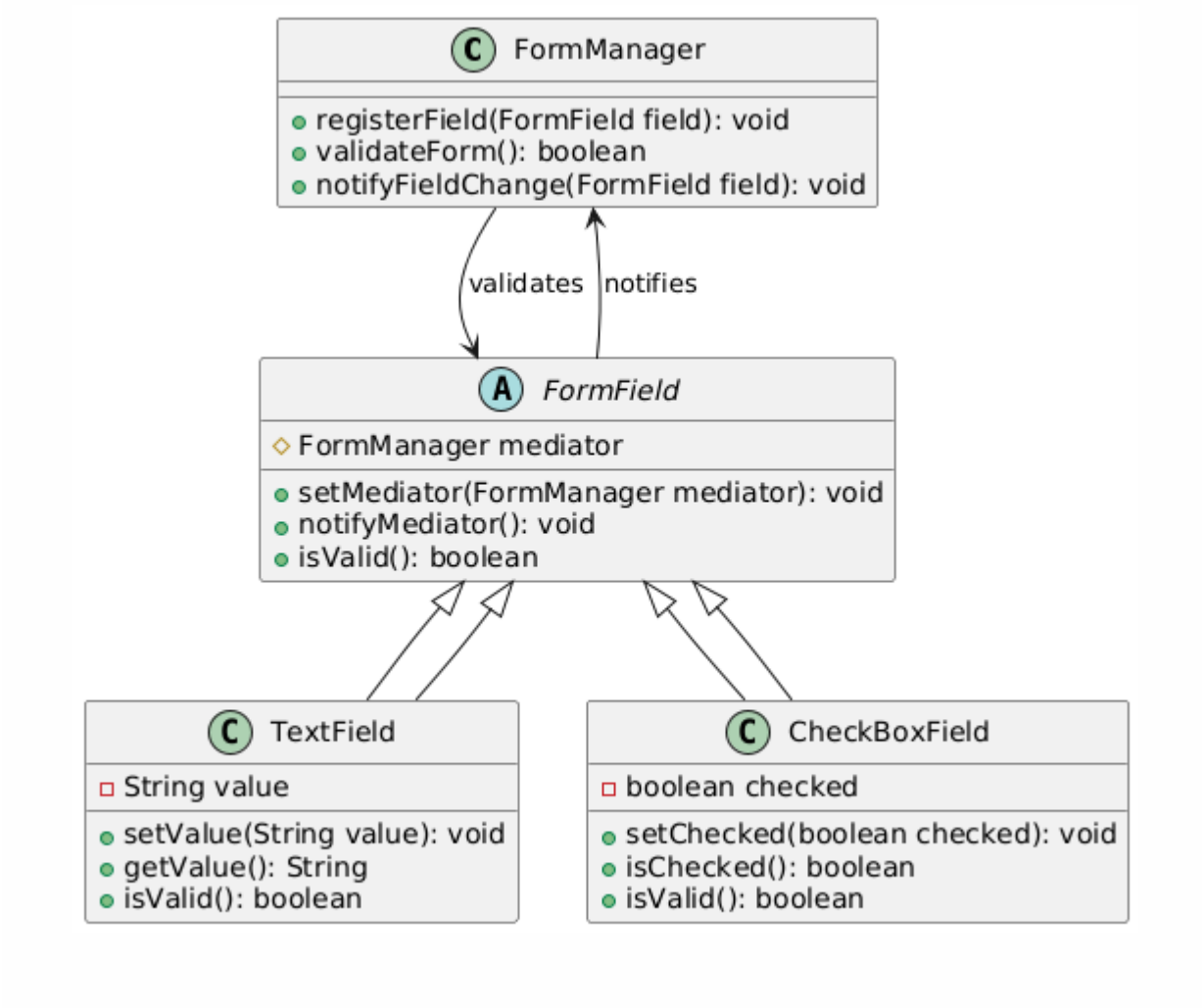

TextEditor
String content
setContent(String content): void
getContent(): String
saveState(): EditorState
restoreState(EditorState memento): void
History
List<EditorState> states
save(EditorState state): void
undo(): EditorState
creates
manages
EditorState
String content
getContent(): String

Task 4: UI Form with Validation

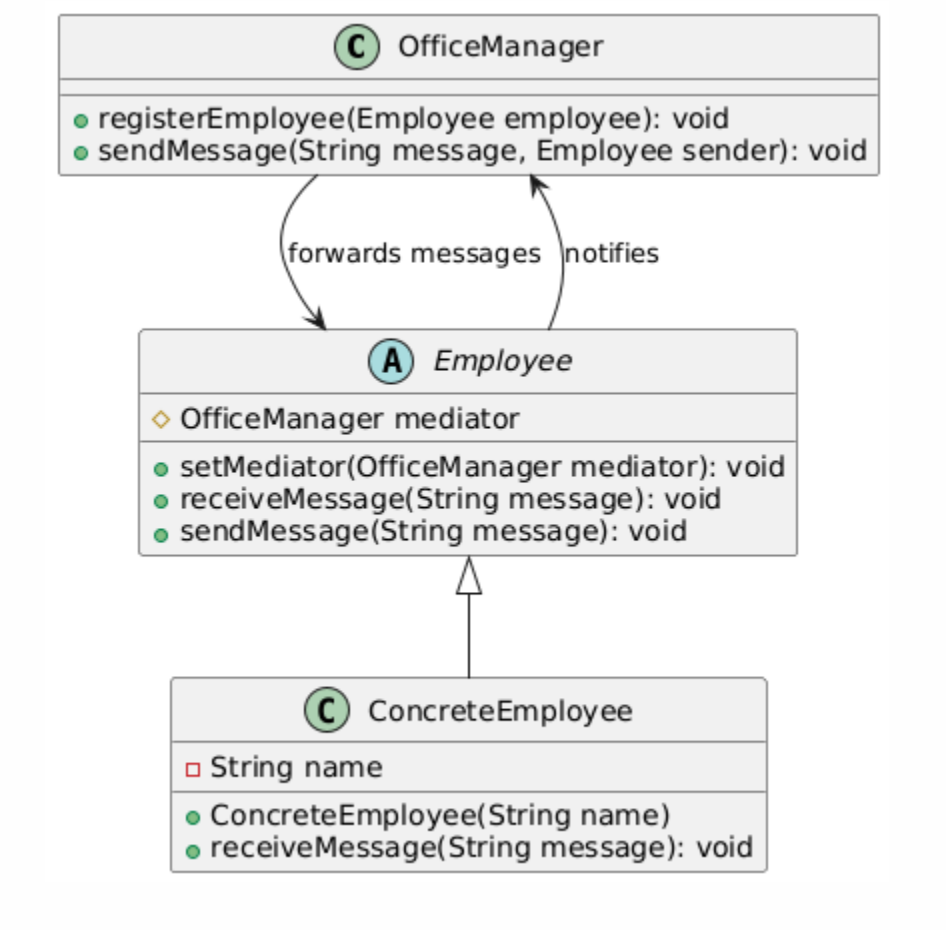
```
public class Main {
    public static void main(String[] args) {
        FormManager formManager = new FormManager();

        TextField usernameField = new TextField();
        TextField passwordField = new TextField();
        CheckBoxField termsCheckbox = new CheckBoxField();

        formManager.registerField(usernameField);
        formManager.registerField(passwordField);
        formManager.registerField(termsCheckbox);

        // Test scenario
        System.out.println("Initial validation:");
        formManager.validateForm();

        System.out.println("\nFilling form:");
        usernameField.setValue("User123");
        passwordField.setValue("Password!");
        termsCheckbox.setChecked(true);

        System.out.println("\nFinal validation:");
        boolean isFormValid = formManager.validateForm();
        System.out.println("Form is " + (isFormValid ? "valid" : "invalid"));
    }
}
```

```java
package org.example;
import java.util.ArrayList;
import java.util.List;

import java.util.ArrayList;
import java.util.List;

// Mediator
class FormManager {
    private final List<FormField> fields = new ArrayList<>();

    public void registerField(FormField field) {
        fields.add(field);
        field.setMediator(this);
    }

    public boolean validateForm() {
        boolean isValid = true;
        for (FormField field : fields) {
            if (!field.isValid()) {
                isValid = false;
                System.out.println(field.getClass().getSimpleName() + " is invalid!");
            }
        }
        return isValid;
    }

    public void notifyFieldChange(FormField field) {
        System.out.println(field.getClass().getSimpleName() + " has been updated.");
        validateForm();
    }
}
```

```
abstract class FormField {
    protected FormManager mediator;

    public void setMediator(FormManager mediator) {
        this.mediator = mediator;
    }

    public void notifyMediator() {
        if (mediator != null) {
            mediator.notifyFieldChange(this);
        }
    }

    public abstract boolean isValid();
}

// Concrete Colleague - TextField
class TextField extends FormField {
    private String value = "";

    public void setValue(String value) {
        this.value = value;
        notifyMediator();
    }

    public String getValue() {
        return value;
    }

    @Override
    public boolean isValid() {
        return value != null && !value.trim().isEmpty();
    }
```


FormManager
registerField(FormField field): void
validateForm(): boolean
notifyFieldChange(FormField field): void
validates
notifies
FormField
FormManager mediator
setMediator(FormManager mediator): void
notifyMediator(): void
isValid(): boolean
TextField
String value
setValue(String value): void
getValue(): String
isValid(): boolean
CheckBoxField
boolean checked
setChecked(boolean checked): void
isChecked(): boolean
isValid(): boolean

Task 3: Colleague Communication in an Office



```java
class ConcreteEmployee extends Employee {
    private final String name;

    public ConcreteEmployee(String name) {
        this.name = name;
    }

    @Override
    public void receiveMessage(String message) {
        System.out.println(name + " received: " + message);
    }
}

// Main Class to demonstrate functionality
public class Main {
    public static void main(String[] args) {
        OfficeManager manager = new OfficeManager();

        Employee alice = new ConcreteEmployee( name: "Alice");
        Employee bob = new ConcreteEmployee( name: "Bob");
        Employee charlie = new ConcreteEmployee( name: "Charlie");

        manager.registerEmployee(alice);
        manager.registerEmployee(bob);
        manager.registerEmployee(charlie);

        // Communication
        alice.sendMessage("Hello, everyone!");
        bob.sendMessage("Hi Alice!");
        charlie.sendMessage("Good morning!");
    }
}
```

```java
class OfficeManager {
    private final List<Employee> employees = new ArrayList<>();

    public void registerEmployee(Employee employee) {
        employees.add(employee);
        employee.setMediator(this);
    }

    public void sendMessage(String message, Employee sender) {
        for (Employee employee : employees) {
            // Don't send the message back to the sender
            if (employee != sender) {
                employee.receiveMessage(message);
            }
        }
    }
}

// Abstract Colleague
abstract class Employee {
    protected OfficeManager mediator;

    public void setMediator(OfficeManager mediator) {
        this.mediator = mediator;
    }

    public abstract void receiveMessage(String message);

    public void sendMessage(String message) {
        if (mediator != null) {
            System.out.println(this.getClass().getSimpleName() + " sent: " + message);
            mediator.sendMessage(message, sender: this);
        }
```