```
import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

/**
 * Representa um compromisso com data, horário e descrição.
 */
class Compromisso {
    private String data;
    private String horario;
    private String descricao;

    /**
     * Constrói um novo compromisso.
     *
     * @param data      A data do compromisso no formato DD/MM/AAAA.
     * @param horario   O horário do compromisso no formato HH:MM.
     * @param descricao A descrição do compromisso.
     */
    public Compromisso(String data, String horario, String descricao) {
        this.data = data;
        this.horario = horario;
        this.descricao = descricao;
    }

    /**
     * Obtém a data do compromisso.
     *
     * @return A data do compromisso.
     */
    public String getData() {
        return data;
    }

    /**
     * Obtém o horário do compromisso.
     *
     * @return O horário do compromisso.
     */
    public String getHorario() {
        return horario;
    }

    /**
     * Obtém a descrição do compromisso.
     *
     * @return A descrição do compromisso.
     */
    public String getDescricao() {
        return descricao;
    }

    /**
     * Define a data do compromisso.
```

```
     *
     * @param data A nova data do compromisso.
     */
    public void setData(String data) {
        this.data = data;
    }

    /**
     * Define o horário do compromisso.
     *
     * @param horario O novo horário do compromisso.
     */
    public void setHorario(String horario) {
        this.horario = horario;
    }

    /**
     * Define a descrição do compromisso.
     *
     * @param descricao A nova descrição do compromisso.
     */
    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    /**
     * Retorna uma representação textual do compromisso.
     *
     * @return A representação textual do compromisso.
     */
    @Override
    public String toString() {
        return "Data: " + data + ", Horário: " + horario + ", Descrição:
" + descricao;
    }
}

/**
 * Gerencia uma lista de compromissos.
 */
class Agenda {
    private List<Compromisso> compromissos;

    /**
     * Constrói uma nova agenda.
     */
    public Agenda() {
        this.compromissos = new ArrayList<>();
    }

    /**
     * Adiciona um compromisso à agenda.
     *
     * @param compromisso O compromisso a ser adicionado.
```

```
     */
    public void adicionarCompromisso(Compromisso compromisso) {
        compromissos.add(compromisso);
    }

    /**
     * Altera um compromisso existente na agenda.
     *
     * @param index       O índice do compromisso a ser alterado.
     * @param compromisso O novo compromisso.
     */
    public void alterarCompromisso(int index, Compromisso compromisso) {
        if (index >= 0 && index < compromissos.size()) {
            compromissos.set(index, compromisso);
        } else {
            System.out.println("Índice inválido.");
        }
    }

    /**
     * Remove um compromisso da agenda.
     *
     * @param index O índice do compromisso a ser removido.
     */
    public void removerCompromisso(int index) {
        if (index >= 0 && index < compromissos.size()) {
            compromissos.remove(index);
        } else {
            System.out.println("Índice inválido.");
        }
    }

    /**
     * Obtém a lista de compromissos da agenda.
     *
     * @return A lista de compromissos.
     */
    public List<Compromisso> getCompromissos() {
        return compromissos;
    }
}

/**
 * Representa uma ação de menu abstrata.
 */
abstract class AcaoMenu {
    protected Agenda agenda;
    protected Scanner scanner;

    /**
     * Constrói uma nova ação de menu.
     *
     * @param agenda  A agenda associada.
     * @param scanner O scanner para entrada do usuário.
```

```
     */
    public AcaoMenu(Agenda agenda, Scanner scanner) {
        this.agenda = agenda;
        this.scanner = scanner;
    }

    /**
     * Executa a ação do menu.
     */
    public abstract void executar();
}

/**
 * Insere um novo compromisso na agenda.
 */
class InserirCompromisso extends AcaoMenu {
    /**
     * Constrói uma nova ação de inserção de compromisso.
     *
     * @param agenda  A agenda associada.
     * @param scanner O scanner para entrada do usuário.
     */
    public InserirCompromisso(Agenda agenda, Scanner scanner) {
        super(agenda, scanner);
    }

    /**
     * Executa a inserção do compromisso.
     */
    @Override
    public void executar() {
        System.out.println("Digite a data do compromisso (DD/MM/AAAA):");
        String data = scanner.nextLine();
        System.out.println("Digite o horário do compromisso (HH:MM):");
        String horario = scanner.nextLine();
        System.out.println("Digite a descrição do compromisso:");
        String descricao = scanner.nextLine();

        Compromisso compromisso = new Compromisso(data, horario,
descricao);
        agenda.adicionarCompromisso(compromisso);
        System.out.println("Compromisso adicionado com sucesso!");
    }
}

/**
 * Altera um compromisso existente na agenda.
 */
class AlterarCompromisso extends AcaoMenu {
    /**
     * Constrói uma nova ação de alteração de compromisso.
     *
     * @param agenda  A agenda associada.
     * @param scanner O scanner para entrada do usuário.
```

```
     */
    public AlterarCompromisso(Agenda agenda, Scanner scanner) {
        super(agenda, scanner);
    }

    /**
     * Executa a alteração do compromisso.
     */
    @Override
    public void executar() {
        System.out.println("Digite o índice do compromisso a ser
alterado:");
        int index = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha

        System.out.println("Digite a nova data do compromisso
(DD/MM/AAAA):");
        String data = scanner.nextLine();
        System.out.println("Digite o novo horário do compromisso
(HH:MM):");
        String horario = scanner.nextLine();
        System.out.println("Digite a nova descrição do compromisso:");
        String descricao = scanner.nextLine();

        Compromisso compromisso = new Compromisso(data, horario,
descricao);
        agenda.alterarCompromisso(index, compromisso);
        System.out.println("Compromisso alterado com sucesso!");
    }
}

/**
 * Remove um compromisso da agenda.
 */
class RemoverCompromisso extends AcaoMenu {
    /**
     * Constrói uma nova ação de remoção de compromisso.
     *
     * @param agenda  A agenda associada.
     * @param scanner O scanner para entrada do usuário.
     */
    public RemoverCompromisso(Agenda agenda, Scanner scanner) {
        super(agenda, scanner);
    }

    /**
     * Executa a remoção do compromisso.
     */
    @Override
    public void executar() {
        System.out.println("Digite o índice do compromisso a ser
removido:");
        int index = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha
```

```
        agenda.removerCompromisso(index);
        System.out.println("Compromisso removido com sucesso!");
    }
}

/**
 * Lista todos os compromissos na agenda.
 */
class ListarCompromissos extends AcaoMenu {
    /**
     * Constrói uma nova ação de listagem de compromissos.
     *
     * @param agenda  A agenda associada.
     * @param scanner O scanner para entrada do usuário.
     */
    public ListarCompromissos(Agenda agenda, Scanner scanner) {
        super(agenda, scanner);
    }

    /**
     * Executa a listagem dos compromissos.
     */
    @Override
    public void executar() {
        System.out.println("Compromissos agendados:");
        List<Compromisso> compromissos = agenda.getCompromissos();
        if (compromissos.isEmpty()) {
            System.out.println("Não há compromissos agendados.");
        } else {
            for (int i = 0; i < compromissos.size(); i++) {
                System.out.println(i + ": " + compromissos.get(i));
            }
        }
    }
}

/**
 * Encerra o programa.
 */
class Sair extends AcaoMenu {
    /**
     * Constrói uma nova ação de saída.
     *
     * @param agenda  A agenda associada.
     * @param scanner O scanner para entrada do usuário.
     */
    public Sair(Agenda agenda, Scanner scanner) {
        super(agenda, scanner);
    }

    /**
     * Executa a ação de saída do programa.
     */
```

```
    @Override
    public void executar() {
        System.out.println("Saindo...");
        System.exit(0);
    }
}

/**
 * Gerencia o menu de opções para o usuário.
 */
class Menu {
    private Scanner scanner;
    private Agenda agenda;

    /**
     * Constrói um novo menu.
     *
     * @param agenda A agenda associada.
     */
    public Menu(Agenda agenda) {
        this.scanner = new Scanner(System.in);
        this.agenda = agenda;
    }

    /**
 * Exibe o menu e gerencia a interação com o usuário.
 * Permite ao usuário escolher entre adicionar, alterar, remover, listar
compromissos ou sair.
 */
public void exibirMenu() {
    while (true) {
        System.out.println("\nEscolha uma opção:");
        System.out.println("1. Adicionar compromisso");
        System.out.println("2. Alterar compromisso");
        System.out.println("3. Remover compromisso");
        System.out.println("4. Mostrar compromissos");
        System.out.println("5. Sair");

        int escolha = scanner.nextInt();
        scanner.nextLine(); // Consumir a quebra de linha

        AcaoMenu acao;
        switch (escolha) {
            case 1:
                acao = new InserirCompromisso(agenda, scanner);
                break;
            case 2:
                acao = new AlterarCompromisso(agenda, scanner);
                break;
            case 3:
                acao = new RemoverCompromisso(agenda, scanner);
                break;
            case 4:
                acao = new ListarCompromissos(agenda, scanner);
```

```
                break;
            case 5:
                acao = new Sair(agenda, scanner);
                break;
            default:
                System.out.println("Opção inválida. Por favor, escolha
novamente.");
                continue;
        }
        acao.executar();
    }
}

/**
 * Ponto de entrada principal do programa.
 * Inicializa a agenda e o menu, e exibe o menu para interação do
usuário.
 *
 * @param args Argumentos da linha de comando (não utilizados).
 */
public class Main {
    public static void main(String[] args) {
        Agenda agenda = new Agenda();
        Menu menu = new Menu(agenda);
        menu.exibirMenu();
    }
}
```