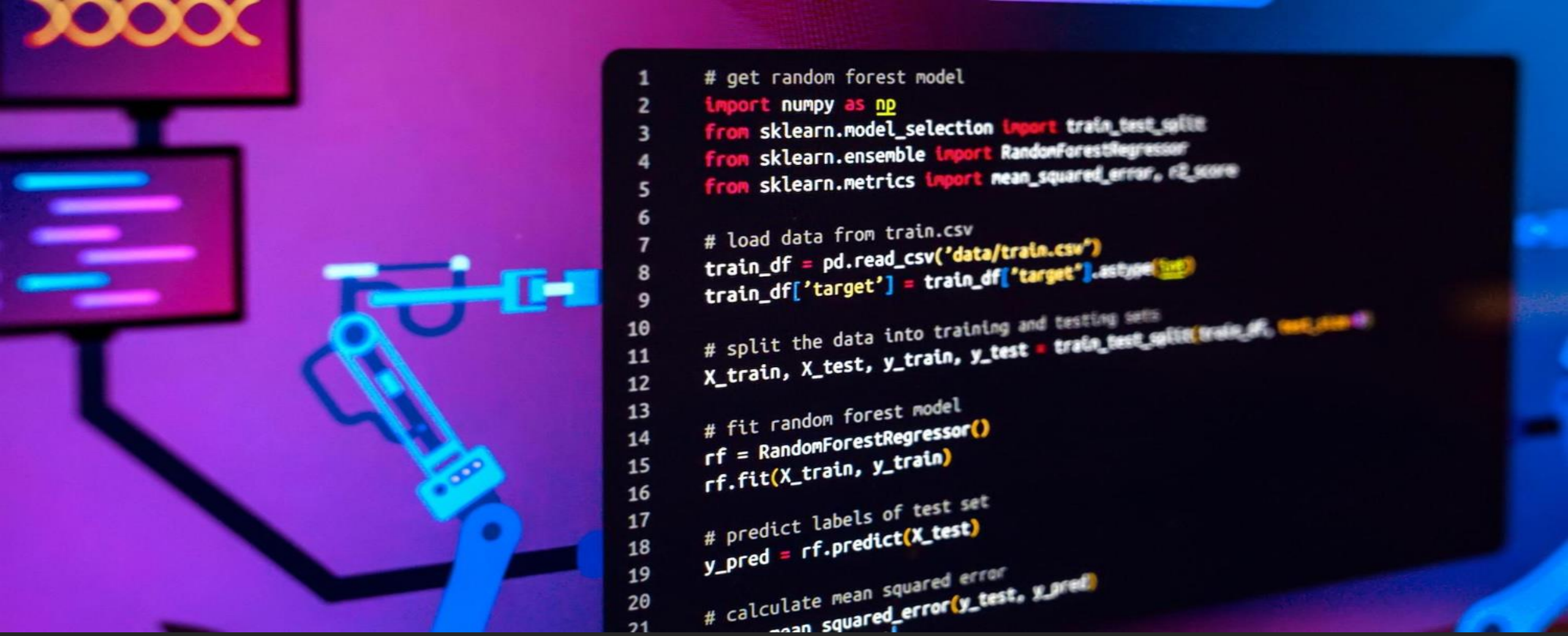
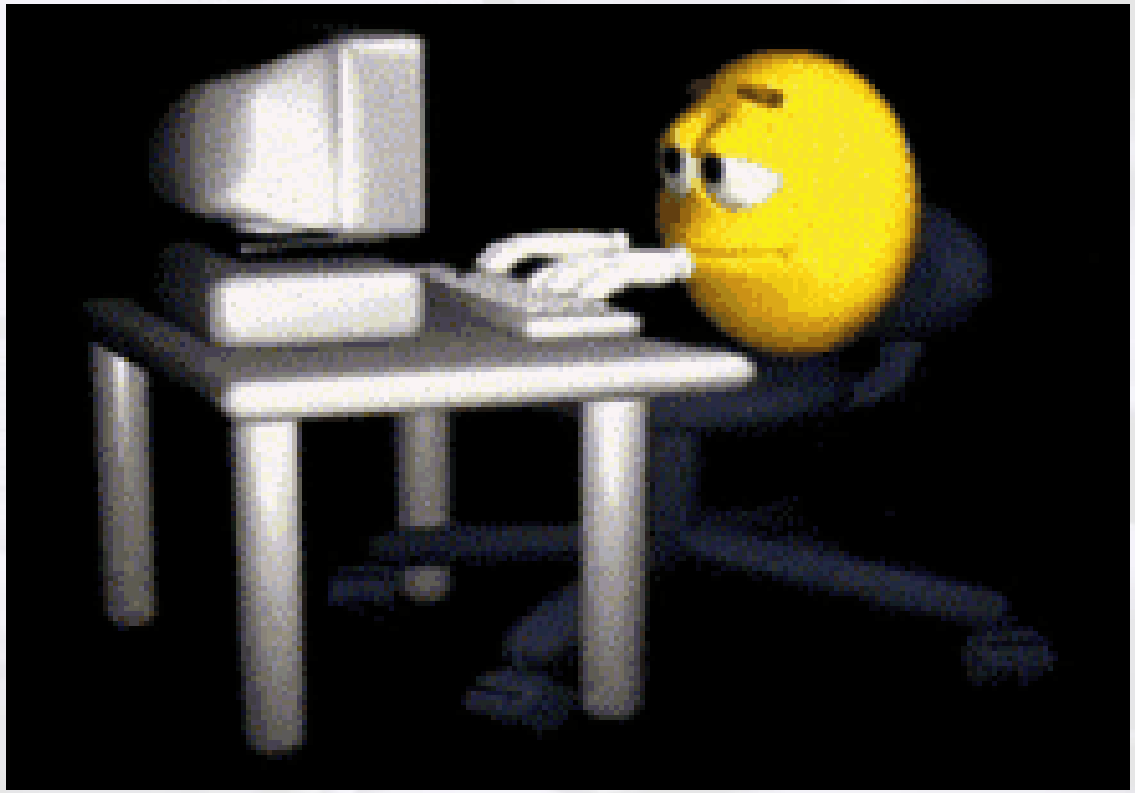


Sistema em Java

INTEGRANTES: CAIO OLIVEIRA, FELYPPE DE ASSIS, GABRIEL GONÇALVES, THIAGO ALVES, VICTOR HUGO.

SISTEMA DE CONTROLE DE KITS

1. Sistema Geral
2. Cadastro de professores
3. Cadastro de Kits
4. Registro de Retirada e Devolução de Kit
5. Histórico



Sistema Geral

O que acontece:

- Cadastrar, editar e remover professores e kits.
- Registrar quando um professor retira ou devolve um kit.
- Mostrar o histórico dessas ações.

```
private void run() {
    do {
        System.out.println("\nMenu:");
        System.out.println("1. Cadastrar Professor");
        System.out.println("2. Listar Professor");
        System.out.println("3. Editar Professor");
        System.out.println("4. Remover Professor");
        System.out.println("5. Cadastrar Kit");
        System.out.println("6. Listar Kit");
        System.out.println("7. Editar Kit");
        System.out.println("8. Remover Kit");
        System.out.println("9. Registrar Retirada de Kit");
        System.out.println("10. Registrar Devolução de Kit");
        System.out.println("11. Ver Histórico");
        System.out.println("12. Sair");
        System.out.print("Escolha uma opção: ");
        opcao = scanner.nextInt();
        scanner.nextLine(); // consumir a quebra de linha

        switch (opcao) {
            case 1:
                cadastrarProfessor();
                break;
            case 2:
                listarProfessor();
                break;
            case 3:
                editarProfessor();
                break;
            case 4:
                removerProfessor();
                break;
            case 5:
                cadastrarKit();
                break;
            case 6:
                listarKit();
                break;
            case 7:
                editarKit();
                break;
            case 8:
                removerKit();
                break;
            case 9:
                registrarRetirada();
                break;
            case 10:
                registrarDevolucao();
                break;
            case 11:
                verHistorico();
                break;
            case 0:
                System.out.println("Saindo do sist
                break;
            default:
                System.out.println("Opção inválida
        }
    } while (opcao != 12);
}
```

Classe Professor

O que acontece:

- É criado uma classe Professor a qual recebe e retorna duas variáveis de tipo String (texto).
- O sistema pede o **nome** e o **CPF** do professor.
- Cria um novo objeto `Professor` e adiciona na lista `professores`.

```
static class Professor {
    private String cpf;
    private String nome;

    public Professor(String cpf, String nome) {
        this.cpf = cpf;
        this.nome = nome;
    }

    public String getCpf() {
        return cpf;
    }

    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    @Override
    public String toString() {
        return "Nome: " + nome + " | CPF: " + cpf;
    }
}

public static boolean validarCpf(String cpf, List<Professor> professores) {
if (cpf == null || cpf.isEmpty()) {
    return false;
}

// Remover caracteres não numéricos
String cpfNumerico = cpf.replaceAll(regex:"\\D", replacement:"");

// Verifica se tem 11 digitos numéricos
if (cpfNumerico.length() != 11) {
    return false;
}

// Verifica se o CPF já está cadastrado
 for (Professor professor : professores) {
    if (professor.getCpf().replaceAll(regex:"\\D", replacement:"").equals(cpfNumerico)) {
        return false; // CPF já existe
    }
}

return true; // CPF válido e ainda não cadastrado
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 1
=== Cadastro de Professores ===
Nome: 1
Nome inválido!
Nome: Carlos
CPF: 1234567
CPF inválido !
CPF: 123.456.789-12
Professor cadastrado com sucesso!
Deseja cadastrar mais professores? (s/n): sim
=== Cadastro de Professores ===
Nome: Marcos
CPF: 987.654.321-98
Professor cadastrado com sucesso!
Deseja cadastrar mais professores? (s/n): não
Cadastro finalizado!
```

Cadastro de Professores

O que acontece:

- É criado um método cadastrarProfessor o qual é o responsável de registrar as duas variáveis criadas anteriormente na classe Professor, sendo cpf único, por meio de um laço condicional.
- Registra essa ação no `histórico`.
- É realizado um laço repetitivo, que pergunta ao usuário se deseja realizar a ação novamente, assim como é feito em todos os outros métodos.

```
private void cadastrarProfessor() {


    do {
        System.out.println(x:"=== Cadastro de Professores ===");

    Scanner scanner = new Scanner(System.in, charsetName:"UTF-8");
    String nome;
    while(true) {
        System.out.print(s:"Nome: ");
       nome = scanner.nextLine();

       if (nome.matches(regex:"^[A-Za-zÀ-ÿ'\\s]+$")) {
        break;
       } else {
        System.out.println(x:"Nome inválido!");
       }
    }

        String cpf;
        while (true) {
        System.out.print(s:"Digite o CPF: ");
        cpf = scanner.nextLine().trim();

        if (validarCpf(cpf, professores)) {
            break;
        } else {
            System.out.println(x:"CPF inválido ou já cadastrado!");
        }

    professores.add(new Professor(cpf.replaceAll(regex:"\\D", replacement:""), nome));
    historico.add(new AcaoRegistro("Professor cadastrado: " + nome));
    System.out.println(x:"Professor cadastrado com sucesso!");

    } while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja cadastrar mais professores?"

    System.out.println(x:"Cadastro finalizado!");
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 1
=== Cadastro de Professores ===
Nome: 1
Nome inválido!
Nome: Carlos
CPF: 1234567
CPF inválido !
CPF: 123.456.789-12
Professor cadastrado com sucesso!
Deseja cadastrar mais professores? (s/n): sim
=== Cadastro de Professores ===
Nome: Marcos
CPF: 987.654.321-98
Professor cadastrado com sucesso!
Deseja cadastrar mais professores? (s/n): não
Cadastro finalizado!
```

Listagem de Professores

O que acontece:

- É criado um método listarProfessor o qual realiza um laço condicional, verificando se há professores adicionados no método anterior, e se houver, são listados.

```java
private void listarProfessor() {


    if (professores.isEmpty()) {
        System.out.println(x:"Nenhum professor cadastrado.");
        return;
    }
    System.out.println(x:"\n=== Lista de Professores ===");
    for (int i = 0; i < professores.size(); i++) {
        System.out.println((i + 1) + " - " + professores.get(i));
    }
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 2

=== Lista de Professores ===
1 - Nome: Carlos | CPF: 123.456.789-12
2 - Nome: Marcos | CPF: 987.654.321-98
```

Edição de Professores

O que acontece:

- É criado um método editarProfessor o qual realiza um laço condicional, verificando se há professores adicionados no método anterior, e se houver, são listados.
- Após isso, é pedido que se receba o valor recebido de cpf, para que se prossiga, e assim permitindo o usuário, se quiser, editar o nome e o CPF.

```
private void editarProfessor() {
    do {
        System.out.println(x:"=== Editar Professor ===");
    listarProfessor();

    if (professores.isEmpty()) {
        System.out.println(x:"Nenhum professor cadastrado.");
        break;
    }

    Professor professor = null;

    while (true) {

    System.out.print(s:"Digite o CPF do professor para editar: ");
    String cpf = scanner.nextLine();


    for (Professor p: professores) {
        if(p.getCpf().equals(cpf)) {
            professor = p;
            break;
        }
    }

    if (professor != null) {
        break;
    } else {
        System.out.println(x:"CPF inválido.");
    }
}


    System.out.print(s:"Novo nome: ");
    String novoNome = scanner.nextLine();
    professor.setNome(novoNome);

    System.out.print(s:"Novo CPF: ");
    String novoCpf = scanner.nextLine();
    professor.setCpf(novoCpf);

    System.out.println(x:"Professor atualizado!");
    historico.add(new AcaoRegistro("Professor editado: " + novoNome));
    } while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja editar mais algum professor?"));

    System.out.println(x:"Edição realizada!");
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 3
=== Editar Professor ===

=== Lista de Professores ===
1 - Nome: Carlos | CPF: 123.456.789-12
2 - Nome: Marcos | CPF: 987.654.321-98
Digite o CPF do professor para editar: 123.456.789-12
Novo nome: Carlos da Silva
Novo CPF: 123.798.345-68
Professor atualizado!
Deseja editar mais algum professor? (s/n): sim
=== Editar Professor ===

=== Lista de Professores ===
1 - Nome: Carlos da Silva | CPF: 123.798.345-68
2 - Nome: Marcos | CPF: 987.654.321-98
Digite o CPF do professor para editar: 123.123.124-12
CPF inválido.
```

Remoção de Professores

O que acontece:

- É criado um método removerProfessor o qual realiza um laço condicional, verificando se há professores adicionados no método anterior, e se houver, são listados.
- Após isso, é pedido que se receba o valor recebido de cpf, para que se prossiga, e assim permitindo o usuário, se quiser, editar o nome e o CPF.

```java
private void removerProfessor() {

    do {
        System.out.println(x:"=== Remover Professor ===");
    listarProfessor();
    if (professores.isEmpty()) return;

    System.out.print(s:"Digite o CPF do professor para remover: ");
    String cpf = scanner.nextLine();

    Professor professorRemover = null;
    for (Professor p: professores) {
        if(p.getCpf().equals(cpf)) {
            professorRemover = p;
            break;
        }
    }

    if (professorRemover == null) {
        System.out.println(x:"CPF inválido.");
        return;
    }

    professores.remove(professorRemover);
    System.out.println(x:"Professor removido!");
    historico.add(new AcaoRegistro("Professor removido: " + professorRemover));
    } while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja remover mais algum professor?"));

    System.out.println(x:"Remoção realizada!");
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 4
=== Remover Professor ===

=== Lista de Professores ===
1 - Nome: Carlos da Silva | CPF: 123.798.345-68
2 - Nome: Marcos | CPF: 987.654.321-98
Digite o CPF do professor para remover: 987.654.321-98
Professor removido!
Deseja remover mais algum professor? (s/n): não
Remoção realizada!
```

Classe Kit

O que acontece:

- É criado uma classe Kit a qual recebe três variáveis, sendo duas de tipo String (texto), e uma booleano (verdadeiro ou falso).
- O sistema pede o **código do kit** e a **descrição**.
- Cria um novo objeto `Kit` e adiciona na lista `kits`.

```
static class Kit {
    private String codigo;
    private String descricao;
    private boolean disponivel = true;

    public Kit(String codigo, String descricao) {
        this.codigo = codigo;
        this.descricao = descricao;
    }

    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public String getDescricao() {
        return descricao;
    }

    public void setDescricao(String descricao) {
        this.descricao = descricao;
    }

    public boolean isDisponivel() {
        return disponivel;
    }

    public void setDisponivel(boolean disponivel) {
        this.disponivel = disponivel;
    }

    @Override
    public String toString() {
        return "Kit: " + descricao + " | Código: " + codigo;
    }
}

public static boolean codigoKitUnico(String codigo, List<Kit> kits) {
    for (Kit kit : kits) {
        if (kit.getCodigo().equalsIgnoreCase(codigo)) {
            return true;
        }
    }
    return false;
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 5
=== Cadastro de Kits ===
Digite o código do kit: 1289
Digite a descrição do kit: Matematica
Kit cadastrado com sucesso!
Deseja cadastrar mais algum kit? (s/n): s
=== Cadastro de Kits ===
Digite o código do kit: 1289
Este código já está em uso. Digite um diferente.
Digite o código do kit: 1288
Digite a descrição do kit: Estatistica
Kit cadastrado com sucesso!
Deseja cadastrar mais algum kit? (s/n): não
Cadastro finalizado !
```

Cadastro de Kits

O que acontece:

- É criado um método cadastrarKit o qual registra as três variáveis criadas na classe Kit, e agora informadas pelo usuário
- O sistema realiza um laço condicional dentro de um laço repetitivo, para o usuário fornecer o código do kit, sendo esse, único.
- Registra essa ação no `histórico`.

```
private void cadastrarKit() {

    do {

    System.out.println(x:"=== Cadastro de Kits ===");
    String codigo;
    while (true) {
    System.out.print(s:"Digite o código do kit: ");
    codigo = scanner.nextLine().trim();

    if (!codigoKitUnico(codigo, kits)) {
        break;
    } else {
        System.out.println(x:"Este código já está em uso. Digite um diferente.");
    }
}


    System.out.print(s:"Digite a descrição do kit: ");
    String descricao = scanner.nextLine();

    kits.add(new Kit(codigo, descricao));
    historico.add(new AcaoRegistro("Kit " + descricao + " cadastrado: " + codigo));
    System.out.println(x:"Kit cadastrado com sucesso!");

    } while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja cadastrar mais algum kit?"));

    System.out.println(x:"Cadastro finalizado !");
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 5
=== Cadastro de Kits ===
Digite o código do kit: 1289
Digite a descrição do kit: Matematica
Kit cadastrado com sucesso!
Deseja cadastrar mais algum kit? (s/n): s
=== Cadastro de Kits ===
Digite o código do kit: 1289
Este código já está em uso. Digite um diferente.
Digite o código do kit: 1288
Digite a descrição do kit: Estatistica
Kit cadastrado com sucesso!
Deseja cadastrar mais algum kit? (s/n): não
Cadastro finalizado !
```

Listagem de Kits

O que acontece:

- É criado um método listarKit o qual utiliza um laço condicional para verificar se há algum kit cadastrado e se houver, lista os kits por meio de um laço repetitivo.

```
private void listarKit() {
    if (kits.isEmpty()) {
        System.out.println(x:"Nenhum kit cadastrado.");
        return;
    }
    System.out.println(x:"\n=== Lista de Kits ===");
    for (int i = 0; i < kits.size(); i++) {
        System.out.println((i + 1) + " - " + kits.get(i));
    }
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 6

=== Lista de Kits ===
1 - Kit: Matematica | Código: 1289
2 - Kit: Estatistica | Código: 1288
```

Edição de Kits

O que acontece:

- É criado um método editarKit o qual utiliza um laço condicional para verificar se há algum kit cadastrado e se houver, edita o kit por meio do código registrado anteriormente.

```java
private void editarKit() {
    do {
        System.out.println(x:"=== Editar Kit ===");
    listarKit();

     if (kits.isEmpty()) {
        System.out.println(x:"Nenhum Kit cadastrado.");
        break;
    }
    Kit kit = null;

    while (true) {
     System.out.print(s:"Digite o código do kit para editar: ");
     String codigo = scanner.nextLine().trim();

    for(Kit k: kits) {
        if(k.getCodigo().equals(codigo)) {
            kit = k;
            break;
        }
    }

    if (kit != null) {
        break;
    } else {
        System.out.println(x:"Código Inválido.");
    }
}
    System.out.print(s:"Novo kit: ");
    String novoKit = scanner.nextLine();
    kit.setDescricao(novoKit);

    System.out.print(s:"Novo código: ");
    String novoCodigo = scanner.nextLine();
    kit.setCodigo(novoCodigo);

    System.out.println(x:"Kit atualizado!");
    historico.add(new AcaoRegistro("Kit editado: " + novoKit));
    } while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja editar mais algum kit?"));

    System.out.println(x:"Edição realizada!");
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 7
=== Editar Kit ===

=== Lista de Kits ===
1 - Kit: Matematica | Código: 12B9
2 - Kit: Estatistica | Código: 12B8
Digite o código do kit para editar: 12B9
Novo kit: Artes
Novo código: 12B10
Kit atualizado!
Deseja editar mais algum kit? (s/n): n
Edição realizada!
```

Remoção de Kits

O que acontece:

- É criado um método removerKit o qual utiliza um laço condicional para verificar se há algum kit cadastrado e se houver, remove o kit por meio do código registrado anteriormente.

```java
private void removerKit() {

    do {
        System.out.println(x:"=== Remover Kit ===");
    listarKit();
    if (kits.isEmpty()) return;

    System.out.print(s:"Digite o código do kit para remover: ");
    String codigo = scanner.nextLine();

    Kit kitRemover = null;
    for (Kit k: kits) {
        if(k.getCodigo().equals(codigo)) {
            kitRemover = k;
            break;
        }
    }

    if (kitRemover == null) {
        System.out.println(x:"Kit inválido.");
        return;
    }

    kits.remove(kitRemover);
    System.out.println(x:"Kit removido!");
    historico.add(new AcaoRegistro("Kit removido: " + kitRemover));
    } while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja remover mais algum kit?"));

    System.out.println(x:"Remoção realizada!");

}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 8
=== Remover Kit ===

=== Lista de Kits ===
1 - Kit: Matematica | Código: 12B9
Digite o código do kit para remover: 12B9
Kit removido!
Deseja remover mais algum kit? (s/n): n
Remoção realizada!
```

Registro de Retirada e Devolução

O que acontece:

- É criado uma classe Registro a qual recebe três variáveis, sendo uma de tipo String (texto), e as outras criadas a partir das classes Professor e Kit.
- É criado outra classe AcaoRegistro a qual receberá as variáveis das classes Professor e Kit, além de duas outras, String e LocalDateTime.
- Então é separado em dois construtores de mesmo nome AcaoRegistro, o primeiro usado quando não houver professor nem kit, sendo só uma descrição de ação, e o outro envolvendo ambos inseridos como retirada ou devolução.

```
static class Registro {
    private Professor professor;
    private Kit kit;
    private String descricao;

    public Registro(Professor professor, Kit kit) {
        this.professor = professor;
        this.kit = kit;
        this.descricao = "Retirada: " + professor.getNome() + " retirou " + kit.getDescricao();
    }

    public Professor getProfessor() {
        return professor;
    }

    public Kit getKit() {
        return kit;
    }

    public Registro(String descricao) {
        this.descricao = descricao;
    }

    @Override
    public String toString() {
        return descricao;
    }
}
```

```
static class AcaoRegistro {
private Professor professor;
private Kit kit;
private String tipo;
private LocalDateTime dataHora;


public AcaoRegistro(String descricao) {
    this.professor = null;
    this.kit = null;
    this.tipo = descricao;
    this.dataHora = LocalDateTime.now();
}

public AcaoRegistro(Professor professor, Kit kit, String tipo) {
    this.professor = professor;
    this.kit = kit;
    this.tipo = tipo;
    this.dataHora = LocalDateTime.now();
}
```

Registro de Retirada e Devolução

O que acontece na Retirada:

- Solicita o **CPF do professor** e o **código do kit**.
- Verifica se o professor existe e se o kit está **disponível**.
- Marca o kit como **indisponível** (`disponivel = false`).
- Registra a ação no histórico com data/hora.

```
private void registrarRetirada() {

    do {
        System.out.println(x:"=== Registro de Retirada de Kit ===");
    System.out.print(s:"CPF do Professor: ");
    String cpf = scanner.nextLine();
    Optional<Professor> professor = professores.stream()
            .filter(p -> p.getCpf().equals(cpf))
            .findFirst();

    if (professor.isEmpty()) {
        System.out.println(x:"Professor não encontrado!");
        return;
    }

    System.out.print(s:"Código do Kit: ");
    String codigo = scanner.nextLine();

    Optional<Kit> kit = kits.stream()
            .filter(k -> k.getCodigo().equalsIgnoreCase(codigo) && k.isDisponivel())
            .findFirst();

    if (kit.isEmpty()) {
        System.out.println(x:"Kit não disponivel ou não encontrado!");
        return;
    }

    kit.get().setDisponivel(disponivel:false);
    historico.add(new AcaoRegistro(professor.get(), kit.get(), tipo:"Retirada"));
    System.out.println(x:"Retirada registrada com sucesso!");

    } while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja registrar mais alguma retirada de kit?"));

    System.out.println(x:"Registro realizado!");


}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 9
=== Registro de Retirada de Kit ===
CPF do Professor: 123.456.789-12
Código do Kit: 1289
Retirada registrada com sucesso!
Deseja registrar mais alguma retirada de kit? (s/n): s
=== Registro de Retirada de Kit ===
CPF do Professor: 123.456.789-12
Código do Kit: 1281
Kit não disponivel ou não encontrado!
```

Registro de Retirada e Devolução

O que acontece na Devolução:

- Solicita o **CPF do professor** e o **código do kit**.
- Verifica se o professor existe e se o kit existe (não precisa estar indisponível).
- Marca o kit como **disponível** (`disponivel = true`).
- Registra a ação no histórico com data/hora.

```
private void registrarDevolucao() {

do {
    System.out.println(x:"=== Registro de Devolução de Kit ===");
System.out.print(s:"CPF do Professor: ");
String cpf = scanner.nextLine();
Optional<Professor> professor = professores.stream()
        .filter(p -> p.getCpf().equals(cpf))
        .findFirst();

if (professor.isEmpty()) {
    System.out.println(x:"Professor não encontrado!");
    return;
}

System.out.print(s:"Código do Kit: ");
String codigo = scanner.nextLine();

Optional<Kit> kit = kits.stream()
        .filter(k -> k.getCodigo().equalsIgnoreCase(codigo))
        .findFirst();

if (kit.isEmpty()) {
    System.out.println(x:"Kit não encontrado!");
    return;
}

kit.get().setDisponivel(disponivel:true);
historico.add(new AcaoRegistro(professor.get(), kit.get(), tipo:"Devolução"));
System.out.println(x:"Devolução registrada com sucesso!");
} while (AcaoRegistro.Continuacao.desejaContinuar(scanner, mensagem:"Deseja registrar mais alguma devolução de kit?"));

System.out.println(x:"Registro realizado!");
}
```

```
Menu:
1. Cadastrar Professor
2. Listar Professor
3. Editar Professor
4. Remover Professor
5. Cadastrar Kit
6. Listar Kit
7. Editar Kit
8. Remover Kit
9. Registrar Retirada de Kit
10. Registrar Devolução de Kit
11. Ver Histórico
12. Sair
Escolha uma opção: 10
=== Registro de Devolução de Kit ===
CPF do Professor: 123.456.789-12
Código do Kit: 12B9
Devolução registrada com sucesso!
Deseja registrar mais alguma devolução de kit? (s/n): n
Registro realizado!
```

Histórico

O que acontece :

- É criado uma variável sobreposta de data de formatação, informando a data da retirada e devolução do kit, que aparecerá no histórico.
- Mostra todas as ações registradas até agora com essas datas.
- A classe `AcaoRegistro` formata a saída com **data e hora** da ação.

```
    @Override
public String toString() {
    DateTimeFormatter formatter = DateTimeFormatter.ofPattern(pattern:"dd/MM/yyyy HH:mm");
    if (professor == null && kit == null) {
        return tipo + " - " + dataHora.format(formatter);
    }
    return tipo + ": " + professor.getNome() + " - Kit " + kit.getDescricao() +
            " - " + dataHora.format(formatter);
}

private void verHistorico() {
    System.out.println(x:"\nHistórico de Ações:");
    for (AcaoRegistro acao : historico) {
        System.out.println(acao);
    }
}
```

```
Menu:
1. Cadastrar Professor
2. Cadastrar Kit
3. Registrar Retirada de Kit
4. Registrar Devolução de Kit
5. Ver Histórico
6. Sair
Escolha uma opção: 5

Histórico de Ações:
Professor cadastrado: Tales - 26/05/2025 12:37
Kit cadastrado: 12B34 - 26/05/2025 12:49
Retirada: Tales - Kit Kit de Direito - 26/05/2025 13:15
Devolução: Tales - Kit Kit de Direito - 26/05/2025 13:15
```

✅ Resumo

Função	O que faz
Sistema Geral	Menu de interação com opções (1 a 6).
Cadastro de Professor	Adiciona professor à lista e registra no histórico.
Cadastro de Kit	Adiciona kit à lista e registra no histórico.
Registrar Retirada	Verifica professor e kit, marca como indisponível e registra.
Registrar Devolução	Marca o kit como disponível e registra devolução no histórico.
Ver Histórico	Exibe todas as ações feitas com data e hora.

Python :

```
a = int(input())
```

Java :

```
import java.util.Scanner ;
Scanner sc = new Scanner(System.in);
int a = sc.nextInt();
```

🏁 Fim