**Projeto:** Pokédex.

**Equipe:** Alcides Ribeiro Sampaio Neto – 402138.
Carlos Eduardo da Silva Ferreira – 397611.

**Funcionalidades:**

- Mostrar a pokédex da primeira geração (151 pokémons), com número de entrada da pokédex, nome e status dos pokémons.

- Buscar um pokémon por nome e mostrar seu status.

- Simular uma batalha entre dois pokémons e retornar o vencedor.

- Gerar um time aleatório de 1 – 6 pokémons e salvá-lo em arquivo.

**Compilando:**

1. No diretório src:
   ```
   $ javac -d ../bin/ *.java
   ```

**Executando:**

1. No diretório bin:
   ```
   $ rmiregistry
   ```

2. Execute o servidor: abra outro terminal:
   ```
   $ java -Djava.server.rmi.codebaseile -
   Djava.security.policy=policy Server
   ```

3. Execute o cliente: abra outro terminal:
   ```
   $ java -Djava.server.rmi.codebaseile -
   Djava.security.policy=policy Client
   ```

**Classes:**

- **Pokemon**: Classe que representa o objeto pokémon.
  - Atributos:
    - Número da pokédex
    - Nome
    - Tipo 1
    - Tipo 2
    - Ataque
    - Defesa
    - Velocidade

  - Métodos:
    - Apenas um construtor que inicializa o objeto e setters.

- **Pokedéx:** Classe que armazena os Pokémons em um ArrayList.
  - Atributos:
    - ArrayList<Pokemon> pokedex

  - Métodos:
    - Contrutor que abre o arquivo “pokedex.txt”, lê os dados dos pokémons e armazena no ArrayList.
    - getAllPokemon(): retorna a pokedex inteira.
    - getByName(String name): busca no ArrayList um pokémon recebendo seu nome como parâmetro, retorna null caso não encontre.

- **File:** Classe simples que para criar objetos para escrever em arquivo, é usada para salvar o time gerado aleatoriamente em arquivo.
  - Métodos:
    - write(String texto): recebe uma String e a salva em arquivo.

- **InterfaceRemota:** Interface com as referências para os métodos que implementas as funcionalidades do projeto.
  - Métodos:
    - getPokedex(): Método referente a funcionalidade 1 (mostrar pokédex).
    - getPokemonStats(): Método referente a funcionalidade 2.
    - Battle(): Método referente a funcionalidade 3.
    - GenerateRandomTeam() + generateFileTeam(): Métodos referentes a funcionalidade 4.

- **Servant:** Classe que implementa os métodos da interface remota.
  - Métodos:
    - Construtor que instancia uma pokédex.
    - getPokedex(): O método retorna o ArrayList de pokémons acessando o método getAllPokemon() da classe Pokedex.
    - Battle(String name_1, String name_2): Recebe duas Strings referentes aos nomes dos pokémons que irão batalhar, o método busca esses pokémons na pokédex e retorna null caso algum deles não for encontrado. Caso contrário, acessamos os atributos de HP, ataque, defesa e velocidade de ambos os pokemons. Primeiramente vemos qual é o mais rápido, o mais rápido tem direito de atacar primeiro. O cálculo do dano é: ataque_do_pokemon_atacante – defesa_do_pokemon_atacado. Caso a defesa do atacado seja maior que o ataque do atacante, o dano será simplesmente o ataque do atacante / 2. Então o HP do pokémon atacado é subtraído pelo dano. Fazemos isso várias vezes com ambos pokémons e quem tiver HP zerado primeiro perde. O vencedor é retornado.
    - GenerateRandomTeam(int generateNumber): O método recebe um inteiro referente a quantidade de pokémons que serão gerados, um time pode ter de 1 a 6 pokémons, um valor diferente desse range retorna null. Caso contrário, o método embaralha aleatoriamente a pokédex com o método shuffle(ArrayList<T>) da classe java.util.Collections, então retornamos o ArrayList que receberá os pokémons.

    - generateFileTeam(ArrayList<Pokemon> team, String caminho): O método recebe o ArrayList gerado no método GenerateRandomTeam() e o camindo onde o arquivo será gerado e então salva o time.

- **Server:** Classe que inicia o servidor. Aqui instanciamos a interface e o objeto remoto.

- **Client:** Classe que implementa a interação com o usuário, com um menu para alternar as funcionalidades. Primeiramente instanciamos um novo SecurityManager e damos um lookup para encontrar o objeto remoto.