```ts
import { clsx, type ClassValue } from "clsx"
import { twMerge } from "tailwind-merge"
import { Pokemon, Ivs, Evs, Moves, Ability, Stats } from "./types"
import axios from "axios"
// This utility function combines class names and merges them intelligently.
// It uses clsx to handle conditional class names and tailwind-merge to resolve
conflicts.
// It allows you to pass any number of class names as arguments and returns a
single string with the merged class names.
/**
 * Combines class names and merges them intelligently.
 * @param inputs - The class names to combine.
 * @returns The merged class names.
 */
export function cn(...inputs: ClassValue[]) {
  return twMerge(clsx(inputs))
}

/**
 * Capitalizes the first letter of a string.
 * If the string contains a hyphen, it capitalizes the first part and formats the
second part in parentheses.
 * @param str - The string to capitalize.
 * @returns The formatted string with the first letter capitalized.
 */
function firstCapitalize(str: string): string {
  if (str.includes("-")) {
    const pindex = str.indexOf("-")
    const name = str.slice(0, pindex)
    const form = str.slice(pindex + 1)
    // If the string contains hyphens, capitalize the first part and join with the
rest
    return name.charAt(0).toUpperCase() + name.slice(1) + " (" +
form.charAt(0).toUpperCase() + form.slice(1) + ")"
  }
  // Capitalizes the first letter of a string
  return str.charAt(0).toUpperCase() + str.slice(1)
}

/**
 * Formats the move name by capitalizing the first letter and replacing hyphens
with spaces.
 * @param name - The move name to format.
 * @returns The formatted move name.
 */
function formatMoveName(name: string): string {
  // Formats the move name by capitalizing the first letter and replacing hyphens
with spaces and capitalizing the first letter
  if (name.includes("-")) {
    const pindex = name.indexOf("-")
    const firstPart = name.slice(0, pindex)
    const secondPart = name.slice(pindex + 1)
```

```
    // Capitalizes the first letter of the first part and formats the second part
    return firstPart.charAt(0).toUpperCase() + firstPart.slice(1) + " " +
secondPart.charAt(0).toUpperCase() + secondPart.slice(1)
  }
  // If no hyphen is found, just capitalize the first letter
  return name.charAt(0).toUpperCase() + name.slice(1)
}

// This Interface defines the structure of the raw Pokémon data fetched from the
API.
interface RawPokemonData {
  id: number
  name: string
  types: string[]
  spriteUrl: string
}

/**
 * Converts raw Pokémon data from the API into instances of the Pokemon class.
 * This function maps over the raw data and creates a new Pokemon instance for
each entry.
 * Values such as abilities, IVs, EVs, stats, and moves are initialized with
default values.
 * The function also sets a placeholder ability and move in case of errors.
 * @param data - The raw Pokémon data to convert.
 * @returns An array of Pokemon instances.
 */
export function convertToPokemonClass(data: RawPokemonData[]): Pokemon[] {
  return data.map((entry) => {
    return new Pokemon(
      entry.id,
      entry.name,
      null,
      [new Ability("Error", "If this appears, something went wrong")],
      100,
      "Male",
      "Hardy",
      false,
      entry.types,
      [
        new Ivs("hp", 31),
        new Ivs("attack", 31),
        new Ivs("defense", 31),
        new Ivs("special-attack", 31),
        new Ivs("special-defense", 31),
        new Ivs("speed", 31),
      ],
      [
        new Evs("hp", 0),
        new Evs("attack", 0),
        new Evs("defense", 0),
        new Evs("special-attack", 0),
        new Evs("special-defense", 0),
        new Evs("speed", 0),
```

```
      ],
      [
        new Stats("hp", 1),
        new Stats("attack", 1),
        new Stats("defense", 1),
        new Stats("special-attack", 1),
        new Stats("special-defense", 1),
        new Stats("speed", 1),
      ],
      [new Moves("Error", "Normal", 1, 1, 1, "Physical")],
      [null, null, null, null],
      entry.spriteUrl,
      "Empty",
    )
  })
}

/**
 * Fetches Pokémon data from the PokeAPI and converts it into Pokemon instances.
 * @returns An array of Pokemon instances fetched from the PokeAPI.
 * @throws Will log an error if the fetch fails.
 */
export async function fetchPokemonDataAndConvert() {
  try {
    // Fetch Pokémon data from the PokeAPI
    const response = await axios.get('https://pokeapi.co/api/v2/pokemon?
limit=493&page=0')
    const results = response.data.results as { name: string; url: string }[]

    // Get detailed data for each Pokémon
    // This part fetches additional details for each Pokémon, such as types and
sprites.
    const detailedData = await Promise.all(
      results.map(async (entry) => {

        const res = await axios.get(entry.url)
        const pokemon = res.data

        // Extract types and sprite URL from the Pokémon data
        const types = pokemon.types.map((typeEntry: any) =>
firstCapitalize(typeEntry.type.name))
        const spriteUrl = pokemon.sprites.front_default

        // Create a raw data object with the necessary fields
        // This object contains the Pokémon's ID, name, types, and sprite URL.
        const rawData: RawPokemonData = {
          id: pokemon.id,
          name: firstCapitalize(pokemon.name),
          types: types,
          spriteUrl,
        }

        return rawData
      })
```

```
    )
    // Convert the detailed data into instances of the Pokemon class and return
the list
    const pokemonList = convertToPokemonClass(detailedData)

    return pokemonList
  } catch (error) {
    // Log any errors that occur during the fetch or conversion process
    console.error('Fehler beim Abrufen der Pokémon-Daten:', error)
    return []
  }
}

/**
 * Fetches specific Pokémon data from the PokeAPI and updates the provided Pokémon
instance.
 * This function retrieves detailed information about a specific Pokémon,
including its abilities, stats, and moves.
 * It updates the provided Pokémon instance with this data.
 * @param pokemon - The Pokémon instance to update with fetched data.
 * @returns A promise that resolves to null if the fetch is successful, or logs an
error if it fails.
 * @throws Will log an error if the fetch fails.
 */
export async function fetchSpecificPokemonData(pokemon: Pokemon): Promise<null> {
  try {
    const response = await
axios.get(`https://pokeapi.co/api/v2/pokemon/${pokemon.getName().toLowerCase()}`)
    const data = response.data

    //Gets the possible Abilities of the Pokémon out of the API response
    const abilities: Ability[] = await Promise.all(
      data.abilities.map(async (entry: any) => {
        const abilityRes = await axios.get(entry.ability.url)
        const effect = abilityRes.data.effect_entries.find((e: any) =>
e.language.name === 'en')
        return new Ability(firstCapitalize(entry.ability.name), effect?.effect ||
'No description')
      })
    )

    // Gets the base stats of the Pokémon out of the API response
    // Maps the stats to the Stats class, which includes the stat name and base
stat value
    const stats: Stats[] = data.stats.map(
      (entry: any) => new Stats(entry.stat.name, entry.base_stat)
    )

    // Gets the moves of the Pokémon out of the API response
    // Limits the number of moves to 50 for performance reasons
    // Maps the moves to the Moves class, which includes the move name, type,
power, accuracy, pp, and damage class
    const moveData = data.moves.slice(0, 50)
    const moves: Moves[] = await Promise.all(
```

```
        moveData.map(async (entry: any) => {
          const moveRes = await axios.get(entry.move.url)
          const move = moveRes.data
          return new Moves(
            formatMoveName(move.name),
            firstCapitalize(move.type.name),
            move.power ?? 0,
            move.accuracy ?? 100,
            move.pp ?? 10,
            move.damage_class.name
          )
        })
      )

      // Update the Pokémon instance with the fetched data
      pokemon.setAbilitys(abilities)
      pokemon.setStats(stats)
      pokemon.setMoves(moves)
      pokemon.setSprite(data.sprites.front_default)
      pokemon.setSprite_back(data.sprites.back_default)


      return null
    } catch (error) {
      // Log any errors that occur during the fetch or update process
      console.error('Fehler beim Abrufen der Pokémon-Daten:', error)
      return null
    }
  }
```