Exercise: "Pakuri: Let's Go!"

Overview

This project will provide students with practice working with object-oriented and functional constructs in Rust, including structs and iterators, by building a module to represent creatures and a cataloguing system.

Scenario

NOTE: This project concept is a work of satire. To state the obvious: we do not advise one to go around imprisoning creatures in small receptacles held in one's pockets and/or having them fight for sport.

Pouch Creatures – abbreviated "Pakuri" – are the latest craze sweeping elementary schools around the world. Tiny magical creatures small enough to fit into one's trouser pouches (with enough force applied, 'natch) have begun appearing all around the world in forests. They come in all shapes and colors. When stolen from their parents at a young enough age, they can be kept in small spherical cages (for their own good) easily carried by elementary school children (though they are also popular with adults). This has led to an unofficial catch phrase for the phenomenon – "Gotta steal 'em all!" – a play on the abbreviation "Pakuri" (which doubles as Japanese slang meaning "to steal"). Young children can then pit their Pakuri against one another in battle for bragging rights or to steal them from one another. (Don't worry – they heal their wounds quickly!)

Lately, those without the resources to go with real Pakuri have latched onto "Pakuri ni Ikou", with the English title "Pakuri: Let's Go". (The literal translation of the title is "Let's Go Pakuri", but localizers felt it was weird. They also changed all of the dialog, as they always do.) This game is a world-wide AR game using geolocation to collect data on users… purely for in-game purposes, of course.

Of course, keeping track of all these critters can be a real task, especially when you are trying to steal so many of them at such a young age! You've decided to cash in – hey, if you don't someone else will – on the morally ambiguous phenomenon by developing an indexing system – a ***Pakudex*** – for kids and adult participants.

Requirements

Students will write and submit two files: a driver program with the **`main()`** function as the entry point (`pakudex.rs`) and a file containing the `Pakuri` struct (`pakuri.rs`).

Driver Program

When run, the program should…

1) Display a welcome message & display the menu
2) Prompt for input & conduct input error checking
3) Follow the output and formatting in this document
4) Not have print statements in the struct method calls
5) Have ***no global variables***

```
Welcome to Pakudex: Let's Go!

Pakudex Main Menu
-----------------
1. List Pakuri
2. Show Pakuri
3. Add Pakuri
4. Remove Pakuri
5. Change Pakuri Level
6. Exit

What would you like to do?
```

Listing Pakuri

This should number and list creatures in lexicographical order. For example, if "Chuchu" and "Allie" were added to the Pakudex (in that order), the list should be:

Success

```
Pakuri In Pakudex:
1. Allie (Gator, level 10)
2. Chuchu (ShockRat, level 4)
```

Failure

```
No Pakuri in Pakudex yet!

Pakudex Main Menu
```

Show Pakuri

The program should prompt for a name, read the name from the user, then display the Pakuri:

Success

```
Enter the name of the Pakuri to display: Quackers

Name: Quackers
Species: PsyGoose
Level: 15
CP: 286
HP: 45
```

Failure

```
Enter the name of the Pakuri to display: PsyDuck
Error: No such Pakuri!

Pakudex Main Menu
-----------------
1. List Pakuri
2. Show Pakuri
```

Adding Pakuri

When adding a Pakuri, a prompt should be displayed to read in the individual's name, species, and level. If the individual is already in the Pakudex, the program should return to the main menu.

Success

```
Pakuri Information
------------------
Name: Quackers
Species: PsyGoose
Level: 15

Pakuri Quackers (PsyGoose, level 15) added!

Pakudex Main Menu
-----------------
```

Failure – Duplicate

```
Name: Quackers
Error: Pakudex already contains this Pakuri!
```

Failure – Level

```
Level: -15
Level cannot be negative.
Level: 500
Maximum level for Pakuri is 50.
Level: NINE-THOUSAND
Invalid level!
```

Removing Pakuri

The program should prompt for a name and then remove the Pakuri if it is in the Pakudex:

Success

```
Enter the name of the Pakuri to remove: Quackers
Pakuri Quackers removed.
```

Failure

```
Enter the name of the Pakuri to remove: PsyDuck
Error: No such Pakuri!
```

Change Pakuri Level

Changing the Pakuri's level should follow the same format used in other options:

Success

```
Enter the name of the Pakuri to change: Quackers
Enter the new level for the Pakuri: 42
```

Failure – No Element

```
Enter the name of the Pakuri to change: PsyGoose
Error: No such Pakuri!
```

Failure – Level

```
Enter the name of the Pakuri to change: Quackers
Enter the new level for the Pakuri: -42
Level cannot be negative.
Enter the new level for the Pakuri: 500
Maximum level for Pakuri is 50.
Enter the new level for the Pakuri: GAJILLION
Invalid level!
```

Note: for numeric input, only the integer form should be stored.

Exit

On exit, the program should print "**`Thanks for using Pakudex: Let's Go! Bye!`**" to the screen.

Pakuri Struct

This struct will be the blueprint for the different creature objects that you will create. You will need to store information about the name, species, and level, along with three attributes: attack, defense, and stamina. The attributes should be private by following standard Rust conventions for protecting struct data.

Attributes are based on name and species via this formula:

$$\text{Attribute} = \text{Species Base} + \text{Individual Value}$$

Attributes are calculated by hashing name and species via the included **`fxhash::hash_bytes(&[u8])`** function:

Attribute	Species Base	Individual Value
Attack	`<species_hash> % 16`	`<name_hash> % 16`
Defense	`(<species_hash> + 5) % 16`	`(<name_hash> + 5) % 16`
Stamina	`(<species_hash> + 11) % 16`	`(<name_hash> + 11) % 16`

While **`Pakuri`** attributes are never revealed to the user, they are used (along with level) to determine the combat power (CP) and healthy points (HP), as follows:

$$\boldsymbol{HP} = Floor(\boldsymbol{Stamina} \times \boldsymbol{Level} / \mathbf{6})$$

$$\boldsymbol{CP} = Floor(\boldsymbol{Attack} \times \sqrt{\boldsymbol{Defense}} \times \sqrt{\boldsymbol{Stamina}} \times \boldsymbol{Level} \times \mathbf{0.08})$$

Required Instance Methods

pub **new**(name: *String*, species: *String*) -> *Self*
This is a constructor for the **`Pakuri`** type. The object should have a level of zero (0).

pub **with**(name: *String*, species: *String*, level: *i32*) -> *Self*
This is a constructor for the **`Pakuri`** type. The object should be at the specified **`level`**.

pub **name**(&self) -> &*String*
Returns a borrowed reference to the **`Pakuri`** object's name field.

pub **species**(&self) -> &*String*
Returns a borrowed reference to the **`Pakuri`** object's species field.

pub **hp**(&self) -> *i32*
Calculates and returns the **`Pakuri`** object's health points (HP).

pub **cp**(&self) -> *i32*
Calculates and returns the **`Pakuri`** object's combat power (CP).

pub **level**(&self) -> *i32*
Gets the **`Pakuri`** object's level field.

pub **set_level**(&mut self, level: *i32*)
Sets the **`Pakuri`** object's level field.

Submissions

NOTE: Your output must match the example output *exactly*, otherwise ***you will not receive full credit for your submission***! Please submit only and exactly these files:

Files: pakuri.rs, pakudex.rs
Method: Submit on Canvas

We will compile the course using this command:

```
finn@BMO:~$ rustc ./pakudex.rs
```