01 ETS

ETS types

Setup

In this section you will observe how different types of ETS tables behave when inserting new data. You are provided with a set of commands to set up different ETS tables. To begin exercises start the `iex` shell:

```
$ iex
```

For convenience, the tables are named according to the type they represent:

- `set_table`
- `ordered_set_table`
- `bag_table`
- `duplicate_bag_table`

Useful functions

- List the contents of a table: `:ets.tab2list(table_name)`
- Add a new element to a table: `:ets.insert(table_name, object)`

Goals of the exercise

- Insert new elements into each table.
- Attempt to insert elements that violate the constraints.

Commands

```
:ets.new(:set_table, [:set, :public, :named_table])
:ets.insert(:set_table, {1, "alice"})
:ets.insert(:set_table, {2, "bob"})
:ets.new(:ordered_set_table, [:ordered_set, :public, :named_table])
:ets.insert(:ordered_set_table, {1, "alice"})
:ets.insert(:ordered_set_table, {2, "bob"})
:ets.new(:bag_table, [:bag, :public, :named_table])
:ets.insert(:bag_table, {1, "alice"})
:ets.insert(:bag_table, {1, "bob"})
:ets.new(:duplicate_bag_table, [:duplicate_bag, :public, :named_table])
:ets.insert(:duplicate_bag_table, {1, "alice"})
:ets.insert(:duplicate_bag_table, {1, "bob"})
```

ETS access

Setup

Similarly to the previous section you are provided with three different tables, but now with different access rights. For convenience, the tables are named after the access rights set on them: `:public_table`, `:protected_table`, `:private_table`.

Moreover, to better showcase access patterns for different ETS tables, you are also provided with three preconfigured modules with client APIs that allow for communication with or between the tables.

NOTE: The modules used in this exercise define a GenServer (a type of process) that will keep ETS table in memory. At this stage, only concern yourself with ETS function - GenServer is a separate topic that will be covered later.

Client API

- `insert/2` - example: `Module.insert(table_name, object)`
- `search/2` - example: `Module.search(table_name, object)`
- `list/1` - example: `Module.list(table_name)`

Exercises

To access exercises, in the `beam2024` folder go to `ets_access` located in `01 ETS` folder.

In the shell start the project with the command:

```
$ iex -S mix
```

For each module call each available function with different table names and observe results.

ETS matching

In the `iex` shell start by running the below commands.

- Let's recreate the table from the slides

  ```
  :ets.new(:countries, [:bag, :named_table])
  ```

- and insert some data to play with

  ```
  :ets.insert(:countries, {:yves, :france, :cook})
  :ets.insert(:countries, {:sean, :ireland, :bartender})
  :ets.insert(:countries, {:marco, :italy, :cook})
  :ets.insert(:countries, {:chris, :ireland, :tester})
  ```

Showcase

- find all cooks, return part of the object

```
:ets.match(:countries, {:"$1", :_, :cook})
```

- find all cooks, reverse order of fields:

```
:ets.match(:countres, {:"$1", :"$0", :cook})
```

- find matching objects

```
:ets.match_object(:countries, {:_, :ireland, :_})
```