3-2: Encapsulation and Generics (Practice)

Artem Pavlov, TII, Abu Dhabi, 16.04.2025

Create new crate

- Create new branch in the repository p32
- Create new library crate p32
- Check that p32 is listed as a member of the workspace in the root Cargo.toml

Bank model 1

- Create module bank
- Create User struct with String field name, u64 field credit_line, and i64 field balance (positive number means debit, negative credit)
- Create Bank struct which contains list of Users, name of the bank, u64 fields credit_interest and debit_interest in basis points, i.e. 0.01%
- Derive appropriate traits for the types

Bank model 2

- Implement the following methods for Bank:
  - calc_balance: calculates bank's balance sheet in the form of two numbers: total bank liabilities and assets (liabilities represent all debit accounts, assets represent all credit accounts).
  - transfer_funds: accepts two user names and transfer amount as positive integer. Transfers the specified amount from one user to another. Returns an error, if its can not be done (e.g. if the origin user hit his credit limit).
  - accrue_interest: update user balances according to bank's interest rates on credit and debit.

Bank model 3

- Implement merge_bank method for Bank.
- The method accepts another bank and moves all user balances from the merged bank to the merging bank.
- If a user has balances in both banks, its balance should be updated in the merging bank.
- After successful execution the merged bank should be destroyed.

Generic shapes

- Create shapes module
- Create a Shape trait with associated constant NAME and the following methods:
  - perimeter: compute shape's perimeter
  - area: compute shape's area
  - scale: accepts factor as `f32` and scales the shape
  - area_to_perimiter: calculates area-to-perimiter ratio (should be equal to 0 for Point)
  - biggest_area: accepts two shapes (with potentially different types) and returns reference to a shape with the biggest area
  - print_properties: prints name, area, and perimeter of the shape

Generic shapes 2

- Define Point, Triangle, Circle, and Rectangle, and DynamicShape (enum with 4 shape variants) types and implement the Shape trait for them.
- Write function which accepts 2 slices of (potentially different) shapes, finds shape with the biggest perimeter to area ratio, print it using Debug formatting, and return reference to it using enum with two variants (the first variant will be used for the shape type in the first slice, and the second variant for the second slice).
- Write unit tests which use the Shape trait.