```
use clap::{arg, command, Command};
mod classes;
mod functions;
mod hello;
mod test;
// https://docs.rs/clap/4.4.18/clap/struct.Command.html#method.help_template
// Valid tags are:
//
// {name} - Display name for the (sub-)command.
// {bin} - Binary name.(deprecated)
// {version} - Version number.
// {author} - Author information.
// {author-with-newline} - Author followed by \n.
// {author-section} - Author preceded and followed by \n.
// {about} - General description (from Command::about or Command::long_about).
// {about-with-newline} - About followed by \n.
// {about-section} - About preceded and followed by â  \nâ  .
// {usage-heading} - Automatically generated usage heading.
// {usage} - Automatically generated or given usage string.
// {all-args} - Help for all arguments (options, flags, positional arguments, and subcommands) including titles
.
// {options} - Help for options.
// {positionals} - Help for positional arguments.
// {subcommands} - Help for subcommands.
// {tab} - Standard tab sized used within clap
// {after-help} - Help from Command::after_help or Command::after_long_help.
// {before-help} - Help from Command::before_help or Command::before_long_help.
// {usage-heading} [Options] [Commands] [Options]

fn main() {

    let matches = command!() // requires clap `cargo` feature in Cargo.toml
        .help_template("{before-help}{name}-{version} {about-with-newline}{author-with-newline}
{usage-heading} [Options] [Commands] [Options]

{all-args}{after-help} ")    // requires clap `help` feature in Cargo.toml
        .version("1.1")
        .propagate_version(true)
        .subcommand_required(true)
        .arg_required_else_help(true)
        .subcommand(
            Command::new("create")
                .about("C-RUD: to create a file.\n Ex: hello create test.txt")
                .arg(arg!([NAME])),
        )
        .subcommand(
            Command::new("retrieve")
                .about("C-R-UD: to retrieve a file")
                .arg(arg!([NAME])),
        )
        .subcommand(
            Command::new("update")
                .about("CR-U-D ...")
                .arg(arg!([NAME])),
        )
        .subcommand(
            Command::new("delete")
                .about("CRU-D ...")
                .arg(arg!([NAME])),
        )
        .get_matches();

    match matches.subcommand() {
        Some(("create", sub_matches)) => println!(
            "'myapp create' was used, name is: {:?}",
            sub_matches.get_one::<String>("NAME")
        ),
        Some(("retrieve", sub_matches)) => println!(
            "'myapp retrieve' was used, name is: {:?}",
            sub_matches.get_one::<String>("NAME")
        ),
        Some(("update", sub_matches)) => println!(
            "'myapp update' was used, name is: {:?}",
            sub_matches.get_one::<String>("NAME")
        ),
        Some(("delete", sub_matches)) => println!(
            "'myapp delete' was used, name is: {:?}",
            sub_matches.get_one::<String>("NAME")
        ),

        _ => unreachable!("Exhausted list of subcommands and subcommand_required prevents `None`"),
    }


    hello::hello();
    let result = functions::add(5, 3);
```

```
    println!("Sum: {}", result);

    let result = functions::multiply(5, 3);
    println!("Product: {}", result);

    let rectangle = classes::Rectangle { width: 10.0, height: 5.0 };
    let circle = classes::Circle { radius: 3.0 };
    let triangle = classes::Triangle { base: 8.0, height: 4.0 };
    println!("Rectangle: width = {}, height = {}, area = {}", rectangle.width, rectangle.height, rectangle.area
());
    println!("Circle: radius = {}, circumference = {}", circle.radius, circle.circumference());
    println!("Triangle: base = {}, height = {}, area = {}", triangle.base, triangle.height, triangle.area());
}
```