# Lab 05 Tasks

## Task 01

Create a class called `Square` with the following attributes:

- `sideLength` – `float` variable
- `area` – `float` variable
- `allareas` – `static` float variable

Create the following methods for your class:

- Constructors (default and parameterized (only takes `sideLength` as input))
- Function to calculate area, the calculated area should always be added to the `allareas` variable each time an area is calculated. (assume it's called only once for each object)
- Getters and Setters for all variables

In your `main` function create 3 Squares with different `sideLength`. Call the `area` method for each of those `Square` objects. After each time the `area` method is called, call that square's getters for `area` and `allareas` to display the updated values.

## Task 02

Create a class called `LoanHelper`, which helps the user calculate their loan payments. The class should have a variable that stores interest `rate` for the loan as a user defined `constant` value. Aside from the that, it also stores the `amount` for the loan taken and `amount of months` that the user will repay the loan in. The loan repayment should be calculated on a monthly basis, and the interest rate should be applied over the monthly return amount. The output should be something like:

"You have to pay 999 every month for 12 months to repay your loan"

Note: first divide loan amount by number of months, then apply the interest rate on it. Interest rate should be a value between `0-0.5%`

## Task 03

Create a class called `ValidateString`. The purpose of this class will be check if the given characters in a string are alphabet only. Numbers and symbols will mean that the string is invalid. By using a parameterized constructor, create multiple objects for your class that accept different strings. Create a constant function that checks whether the string variable is valid or not.

What happens if you do not make your function constant? Add a comment right above your function explaining why we make a function constant.

## Task 04

Create a `BankAccount` class. Which contains following details and functionalties:

Attributes: The `BankAccount` class has three private member variables: `accountNumber`, `accountHolderName`, and `balance`.

Create Constructor: The class has a constructor that takes parameters to initialize the account details (`accountNumber`, `accountHolderName`, and `balance`).

Create following Member Functions:

- `deposit(double amount)`: Adds the specified amount to the account balance.
- `withdraw(double amount)`: Subtracts the specified amount from the account balance, if sufficient funds are available.
- `display()`: Displays the account details including the account number, account holder name, and balance.

In the `main()` function, create an array accounts of `BankAccount` objects. The array contains three elements, each representing a different bank account.

Initialize Each `BankAccount` object with specific account details such as account number, holder name, and initial balance.

Perform following operations:

- Iterate through each account in the accounts array.
- For each account, display the account details using the `display()` function.
- Perform Two Transactions:
- Deposits `500.0` rupees into the account.
- Withdraws `200.0` rupees from the account.
- After each transaction, display the updated account details, including the new balance.

## Task 05

Keeping in mind our previous car example, write a class that represents a car, and use aggregation and composition to combine different components like `engine`, `wheels`, `headlights` and `steering` to create the car object.

> Hint: create the individual classes first before performing the composition. Remember that for aggregation, you will need pointers! You'll be needing constructors and setters to set these values in case of aggregation. Same hint applies to other questions.

## Task 06

Write a program that uses composition to implement a game engine. A game engine is made up of several components. For example:

a) Graphics rendering engine

b) Input handler

c) Physics engine

You don't have to write the logic for how these individual components work.

## Task 07

Implement a restaurant ordering system that holds information about the restaurant's menus, `menu items`, `orders`, and `payments`. Identify the relationship between the five entities mentioned. Keep in mind the following information as well:

a) Menu Items hold two things: `food name` and `food price`.

b) `Menu` is a class that holds an array of menu items. You can have different functions to add and remove items, or display the entire menu.

c) Restaurant ordering system has one menu.

d) Any staff member can place an order to the system on behalf of a customer. The order class consists of one or more menu items and a payment.

e) `Payment` is a class that holds the amount of money that a customer needs to pay. This is generated when the order is placed.