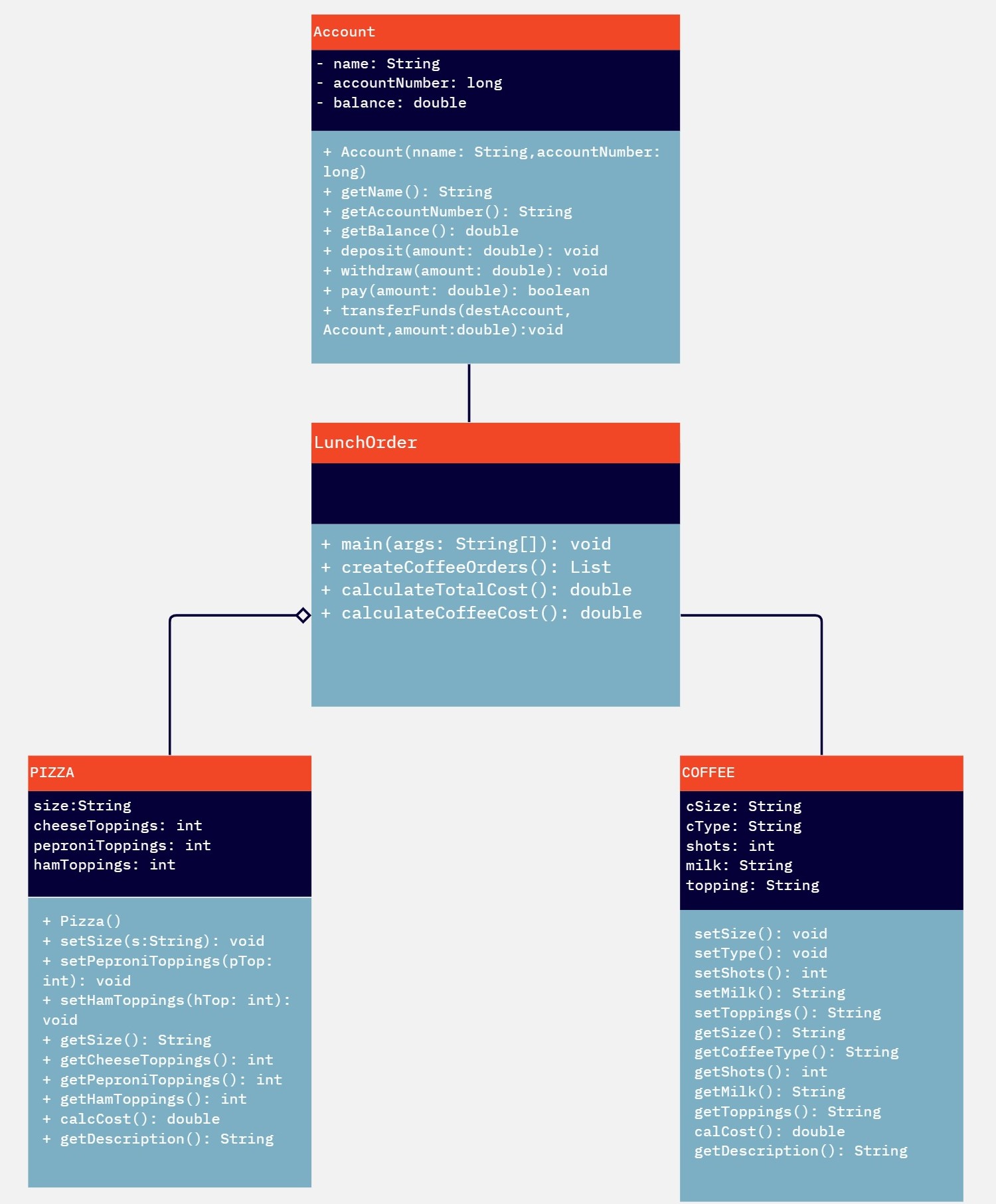
# **Design Document: LunchOrder**

**Problem Description**

The **LunchOrder** program aims to provide a solution for customers to place orders for coffee and pizza using their accounts. The program should allow customers to order items, make payments using their wallet balance, and transfer funds between accounts.

**Design Overview**

The solution consists of four classes: Account, Pizza, Coffee, and **LunchOrder**. Each class has specific responsibilities and relationships with other classes. The UML diagram below illustrates the inheritance hierarchy and class relationships:



**Class Descriptions**

## **Account Class**

The Account class represents a customer account and handles account-related operations such as managing the account balance and processing transactions. The class has the following attributes:

* name: Name of the customer
* accountNumber: A unique identifier for each account.
* balance: The current balance in the account.

The Account class provides the following methods:

* deposit(amount: double): Deposits the specified amount into the account balance.
* withdraw(amount: double): Withdraws the specified amount from the account balance.
* transfer(amount: double, destinationAccount: Account): Transfers the specified amount from the current account to the destination account.
* getBalance(): double: Retrieves the current balance of the account.
* getName(): String: Retrieves the name of the account holder.
* getAccountNumber(): String: Retrieves the account number of the account holder.
* pay(amount: double): deduct particular amount from user’s account if there is sufficient balance

## **Pizza Class**

The Pizza class represents a pizza order and contains information about the pizza's size and toppings. The class has the following attributes:

* size: The size of the pizza (e.g., small, medium, large).
* toppings: The list of toppings on the pizza.

The Pizza class provides the following methods:

* getSize(): String: Retrieves the size of the pizza.
* getToppings(): List<String>: Retrieves the list of toppings on the pizza.
* setToppings(topping: int): Adds a topping to the pizza, there are various methods including setHamToppings() and setPeproniToppings().
* calcCost(): returns double type containing the calculated cost of the pizza according to different toppings
* getDescription(): Returns the description of the pizza by showing different propertiers including size and toppings.

## **Coffee Class**

The **Coffee** class represents a coffee order and has the following properties:

* cSize (String): The size of the coffee.
* cType (String): The type of coffee.
* shots (int): The number of shots in the coffee.
* milk (String): The type of milk used in the coffee.
* topping (String): The topping on the coffee.

The **Coffee** class provides the following methods:

* setSize(String csize): Sets the size of the coffee.
* setType(String tp): Sets the type of coffee.
* setShots(int sht): Sets the number of shots in the coffee.
* setMilk(String mlk): Sets the type of milk used in the coffee.
* setToppings(String top): Sets the topping on the coffee.
* getSize(): String: Retrieves the size of the coffee.
* getCoffeeType(): String: Retrieves the type of coffee.
* getShots(): int: Retrieves the number of shots in the coffee.
* getMilk(): String: Retrieves the type of milk used in the coffee.
* getToppings(): String: Retrieves the topping on the coffee.
* calCost(): double: Calculates the cost of the coffee based on its properties.
* getDescription(): String: Retrieves a description of the coffee, including its size, type, shots, milk, topping, and cost.

## **LunchOrder Class**

The **LunchOrder** class contains the main entry point for the program and includes test code for simulating lunch orders. It has the following methods:

* main(String[] args): The main method that executes the lunch order simulation.
* createCoffeeOrders(): List<Coffee>: Creates a list of coffee orders for testing purposes.
* createPizzaOrders(): List<Pizza>: Creates a list of pizza orders for testing purposes.
* simulateLunchOrders(List<Coffee> coffeeOrders, List<Pizza> pizzaOrders): Simulates the lunch order process using the provided coffee and pizza orders.

The **LunchOrder** class is responsible for creating customer accounts, depositing money into the accounts, creating coffee and pizza orders, simulating lunch orders, calculating total costs, making payments using the customer's wallet, and transferring funds between accounts.