# Project: MoneyService/HQ

## Team Center

**StatisticData**

- data: Map<String, Map<String, Integer> >
- site: String
- date: LocalDate

+ mergeData(StatisticData)
+ putToData(String, Map<String, Integer>)
+ initializeDataFromStatistics(
  List<Statistic>, LocalDate,
  LocalDate): List<StatisticData>

Responsibilities
Storing calculated statistics for each day

**HQApp**

~ sites: Map<String, Site>
~ currencyMap: Map<String, Currency>
~ AMOUNT_LIMIT = 50

+ main()
+ readCurrencyConfigFile(String filename): map<String, Currency>
- parseInput(String): Currency
- HQmenu(): int
- CLIapplication():
- createNewSite():
- readSiteConfig():
- writeNewSiteToConfigFile(String):

Responsibilities
The initializing file for MoneyService HQ

**Currency**

- currencyCode: String
- exhangeRate: double

+ Currency(String, double)
+ getExhangeRate(): double
+ getCurrencyCode(): String
+ setExhangeRate(double):

Responsibilities

**Site**

- siteName: String
- transactions: List<Transaction>

+ Site(String siteName)
+ readTransactions(LocalDate, LocalDate)
+ getSiteName(): String
+ getCompletedTransactions(): List<Transaction>

Responsibilities

Contains a list of completed transaction related
to instance site.

ReadTransaction should read transactions from files according
to dates. The list should return to HQ App for statistics.

1..n

«creates»

«creates»

**CLIHelper**

- DISPLAY_COLUMN_WITH = 20

~ menuInput()
- readSites(): Set<Site>
- readStartDay(): Optional<LocalDate>
- readPeriod(): Optional<Period>
- readCurrencyCodes(): List<String>
- createEndDay(Optional<Period>, Optional<LocalDate>)
- currencyFilter(List<String>): Predicate<Transaction>
~ headDisplayer(List<String> titles): String
~ rowDisplayer(List<Map<String, Integer> >, String, List<String>): String
~ displayTable(Set<Site>, LocalDate, Period, List<String>, DisplayOption)
~ continueShowDisplayOptions(Optional<DisplayOption>): Boolean
~ displayStatData(List<String>): BiConsumer<String, List<StatisticData»

Responsibilities

Command-line interface for HQ application.
Handles user inputs and renders data.
Should present following
criteria: Site, Period and Currency.

**Statistic**

- currencyCodes: List<String>
- siteName: String
- transactions: List<Transaction>
- PROFIT_MARGIN_RATE = 0.005
- SELL_RATE = 1.005
- BUY_RATE = 0.995

+ Statistic(List<Transaction>, List<String>, String)
+ getTotalTransactions(): Map<String. int>
+ getTotalTransactionsBuy(): Map<String, int>
+ getTotalTransactionsSell(): Map<String, int>
+ getTotalAmount(String): Map<String. int>
+ getTotalAmountBuy(String): Map<String. int>
+ getTotalAmountSell(String): Map<String. int>
+ getDiffCurrency(): Map<String. int>
+ getProfit(String): Map<String. int>
+ getAverageAmountBuy(String): Map<String, int>
+ getAverageAmountSell(String): Map<String, int>
+ initializeFromSites(Set<Site>, LocalDate, LocalDate, List<String>): List<Statistic>

Responsibilities
An instance should calculate different values on
transaction list according to currencies in currencyCodes.

1..n

**«Value Type»**
**Transaction**

- serialVersionUID: Long
- id: int
- timeStamp: LocaleDateTime
- currencyCode: String
- amount: int
- mode: TransactionMode
- uniqueId: int

+ Transaction(String, int, TransactionMode)
+ getTimeStamp: LocalDateTime
+ getCurrencyCode: String
+ getAmount(): int
+ getMode: TransactionMode
+ getId(): int

Responsibilities
Represents a completed order. When order is
validated, a transaction is created.

**«Enum»**
**TransactionMode**

BUY, SELL

Responsibilities

Defines if the Order/Transaction is a
Sell or Buy type.