HQ_SouthGroup_V3.0

HQ_App
~input = new Scanner(System.in): Scanner
-logger: Logger

+main(String[] args): void

«use» →

«interface»
HQ

+filteredTran(Request request, String location): void
+printFilteredMap(): void
+profitStatistic(Request request): List<ProfitResult>

«use» ↓

LogHelper

+buildLog(): void

«use» ↓

HQ_MoneyService_Filter

+isLoggable(LogRecord record): boolean

«use»

«Create»

«instance»

AppGUI
~input = new Scanner(System.in): Scanner
-logger: Logger

~SiteChoice(): String
~PeriodChoice(): String
~StartDay_Period(): String
~currencyChoice(Map<String, Double>
            currencyMap): String
~filePath(): String
~StatisticType(): String
~printingChoiceForStats(String siteChoice,
            String periodChoice,
            String startDayPeriod,
            currencyCode): void

Responsibilities
Interact with the user

«Valuetype»
Request
-site = new ArrayList(): List<String>
-duration: String
-date: LocalDate
-currency: String
-logger: Logger

+Request(String site, String duration,
LocalDate date, String currency): ctor
+getSite(): List<String>
+getDuration(): String
+getDate(): LocalDate
+getCurrency(): String
+hashCode(): int
+equals(Object obj): boolean
+toString(): String

Responsibilities
Collect all the necessary info for
static

HQ_MoneyService
-result: Map<String, Map<LocalDate, List<Transaction>>
-currencyMap: Map<LocalDate, Map<String, Double>
-logger: Logger

+HQ_MoneyService(Map<LocalDate, Map<String, Double> currencyMap): ctor
+filteredTran(Request staticRequest, String location): void
-keyMapper(): Function<String, LocalDate>
-valueMapper(String currencyCode): Function<String, List<Transaction>>
+printFilteredMap(): void
+profitStatistic(Request staticRequest): List<ProfitResult>
+readObject(String filename): List<Transaction>
+dayCounter(String duration): int

«Valuetype»
Transaction
-id: int
-timeStamp: LocalDateTime
-currencyCode: String
-amount: int
-mode: TransactionMode
-uniqueId = 1: int

+Transaction(int id,
        LocalDateTime timeStamp,
        String currencyCode,
        int amount,
        TransactionMode mode): ctor
+getId(): int
+getTimeStamp(): LocalDateTime
+getCurrencyCode(): String
+getAmount(): int
+getMode(): TransactionMode
+hashCode(): int
+equals(Object obj): boolean
+toString(): String

Responsibilities
Hold all the info about each Transaction

n

n

«Valuetype»
ProfitResult
~buy: int
~sell: int
~timeStamp: LocalDate
~profit: int
~siteName: String
~currencyCode: String

+ProfitResult(int buy,
        int sell,
        LocalDate timeStamp,
        String siteName,
        String currencyCode): ctor
+getBuy(): int
+getSell(): int
+getTimeStamp(): LocalDate
+getProfit(): int
+getSiteName(): String
+getCurrencyCode(): String
+hashCode(): int
+equals(Object obj): boolean
+toString(): String

Responsibilities
Collecting All the necessary info for
presenting profit type of transactions
Statistic

«use»

ConExApp
+Equal_Separator = "=": String
+Semicolon_Separator = ";": String
+Currency_Code_Place = 2: int
+Currency_Rate_Place = 3: int
-logger: Logger

+readSiteNamesConfig(String fileName): Optional<List<String>>
+readCurrencyConfigFiles(String duration, LocalDate startDate, String filesLocations): Optional<Map<LocalDate, Map<String, Double>>>
+readCurrencyConfig(String fileName): Optional<Map<String, Double>>
+helpToParsing(String sepearator, int partNo): Function<String, String>
-keyMapper(String sepearator, int partNo): Function<String, String>
-valueMapper(String sepearator, int partNo): Function<String, Double>

Responsibilities
Configure the App with All the necessary info in order to be ready