**FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY UNIVERZITA KOMENSKÉHO**

# Špecifikácia komponentov

CupCalculator

Dávid Koszeghy
Jakub Bičian
Dávid Jámbor

# 1. Komponent Parser

Hlavna trieda, pre spracovanie XML súborov. Ako vstupný parameter dostáva adresu vstupného súboru. Konštruktor triedy vytvorí 3  premenné, ktoré obsahujú všetky potrebné informácie.

Vysvetlivky:
person - preposielam premennu, cez metódy
child - preposielam vrchol stromu, aby so mohol pokračovať novým for cyklom. V každej metóde pre prehladnosť kódu sa používa vhodný ekvivalent. Ale pre zjednodušenie dokumentu, dodržim jednotný názov a to child.

Podrobný popis metód triedy Parser.

## 1.1 def __init__(self, file):

Definuje premenné pre parsovanie vstupného súboru. Metóda plní 3 funkcie.
Načítanie .xml súboru do ElementTree parseru a definovanie koreňa xml stromu.
Inicializácia 3 premenných (class_count, event_data, runners_data).
Následne sa spustia metódy na naplnenie premenných.

## 1.2 def collect_event_data(self):

Nájde event tag v  xml súbore (pomocou get_event metódy) a zparsuje tagy: Name, EventClassificationId a StartDate. Tieto dáta sa uložia do event_data premenej.

### 1.2.1 def get_event(self):

For cyklom prejde koreň stromu, kde hľadá tag s názvom Event. Prideli ho premmenej, ktorú funkcia vracia.

### 1.2.2 def parse_startdate(self, child):

Potrebujem sa v noriť do stromu aby som načítal Date a Clock tag, ktoré obsahujú dátum a čas začatia behu. Pridá hodnoty do event_data pod klúčom data a clock.

## 1.3 def collect_runners_data(self):

Takisto ako collect_event_data prehľadáva xml strom, v ktorom hľadá ClassResult. Ak sa daný tag nájde, zavolá sa metóda handle_classresults. A zvýšim inkrement pre class_count (obsahuje počet kategórií).

### 1.3.1 def handle_classresults(self, child):

Metóda sa vnorí hlbšie do stromu a identifikuje tagy ClassShortName a PersonResult.

ClassShortName - obsahuje skrátený názov kategórie, prideľuje sa lokálnej premennej na úpravu

PersonResult - podrobný výpis výsledkov bežca, slovník s kategóriamy (ClassShortname)

### 1.3.2. def get_personresult(self, child):

Vstupuje do PersonResult tagu, ktorý obsahuje 3 ďalšie tagy (Person, Club, Result). Na každý z nich sa volá vlastná funkcie, ktorá spracuje vrchol xml stromu.

### 1.3.3. def parse_person_tag(self, person, child):

V Person tagu, vyhľadá PersonName tag, ktorý ma 2 tagy: Family a Given. Zavolá funkciu add_name_and_surname.

### 1.3.4 def add_name_and_surname(self, person, child):

Z Family a Given tagov získa meno a priezvisko bežca a prida do slovníka, ktorý obsahuje info o bežcovi.

### 1.3.5 def parse_club_tag(self, person, child):

Spracuje ShortName tag, ktorý ma meno klubu, v ktorý bežec reprezentuje.

### 1.3.6 def parse_result_tag(self, person, child):

Najpodstatnejšie informácie o bežcovi. Obsahuje CCard, StartTime, FinishTime, Time, ResultPosition, CompetitorStatus, CourseLength, SplitTime.

### 1.3.7 def add_ccard(self, person, child):

Pridá ccard hodnotu bežcovi.

### 1.3.8 def add_starttime(self, person, child):

Pridá starttime hodnotu bežcovi.

### 1.3.9 def add_finishtime(self, person, child):

Pridá finishtime hodnotu bežcovi.

### 1.3.10 def add_splittime(self, person, child):

SplitTime tag obsahuje hodnoty etáp behu. Vytvorí dvojicu informácií ohľadom ControlCode a Time, ktorý sa pridá bežcovi ako hodnota  splittime_seq_index.

# 2.0 Komponent databaza

## 2.1. class dbHandler():

Trieda pre narabanie s databazou, umoznuje vytvarad datovy model aplikacie a vkladat udaje do databazy. Trieda vytvori spojenie s built-in sqlite3 databazou kt. je defaultne v pythone. Vytvori subor v pracovnom adresary kt. bude pouzity ako ulozisko databazy. V pripade potreby je mozne tuto triedu adaptovat na vykonnejsi databazovy system( napr. PostgreSQL). Ak subor s databazou existuje, da sa overit ci databaza obsahuje potrebny navrhnuty datovy model. Ak nie je mozne ho vytvorit znova. Po uspesnom pripojeni na databazu je mozne vyuzivat metody na pridavanie udajov do databazy.

## 2.2. def __init__(self):

Inicalizuje celu triedu, vsetky potrebne premenne a vytvori spojenie s databazou. Momentalny pracovny nazov suboru je “calc.db”

## 2.3. def createTables(self):

Vytvori potrebny datovy model aplikacie pomocou SQL prikazov, kazda z jednotlivych tabuliek je popisana v Navrhu aplikacie.

## 2.4. def isRunnerExists(self):

Metoda vracia boolean vysledok. Ak sa v pripojenej databaze nachadza tabulka runners tak vrati True, inak False

## 2.5. def isRunsExists(self):

Metoda vracia boolean vysledok. Ak sa v pripojenej databaze nachadza tabulka runs tak vrati True, inak False

## 2.6. def isCategoriesExists(self):

Metoda vracia boolean vysledok. Ak sa v pripojenej databaze nachadza tabulka categories tak vrati True, inak False

## 2.7. def isClubsExists(self):

Metoda vracia boolean vysledok. Ak sa v pripojenej databaze nachadza tabulka clubs tak vrati True, inak False

## 2.8. def isResultsExists(self):

Metoda vracia boolean vysledok. Ak sa v pripojenej databaze nachadza tabulka rusults tak vrati True, inak False

## 2.9. def isSeasonsExists(self):

Metoda vracia boolean vysledok. Ak sa v pripojenej databaze nachadza tabulka seasons tak vrati True, inak False

## 2.10. def isPositionsExists(self):

Metoda vracia boolean vysledok. Ak sa v pripojenej databaze nachadza tabulka positions tak vrati True, inak False

## 2.11. def addRunner(self, cc_card, first_name, last_name, club_id, category_id):

Metoda umoznuje pridat bezca do databazy, potrebne udaje su:

- cc_card
- first_name
- last_name
- club_id
- category_id

## 2.12. def addCategory(self,name):

Metoda umoznuje pridat categoriu do databazy, potrebne udaje su:

- name

## 2.13. def addClub(self,name):

Metoda umoznuje pridat club do databazy, potrebne udaje su:

- name

## 2.14. def addRun(self, name, date_run, start_time, end_time, position, status, season_id, runner_id):

Metoda umoznuje pridat run do databazy, potrebne udaje su:

- name
- date_run

- start_time
- end_time
- position
- status
- season_id
- runner_id

## 2.15.def addSeason(self, name, start_date, end_date):

Metoda umoznuje pridat season do databazy, potrebne udaje su:

- name
- start_date
- end_date

## 2.16.def exitHandler(self):

Metoda korektne uzatvori spojenie s databazou a commitne zmeny kt. nastaly v spojeni s dataabzou, bez tohto prikazu sa zmeny v databaze NEprejavia a tak je potrebne tuto metodu pouzivat zakazdym ak sa nieco zmeni.

# 3. Komponent Hodnotenie

Tento komponent obsahuje štyri triedy : Evaluation1, Evaluation2, Evaluation3 a Evaluation4. Každá z týchto tried sa stará o jeden systém hodnotenia, vypísaný v špecifikácii aplikácie.

## 3.1 Evaluation1

Táto trieda má za úlohu ohodnotiť bežcov na základe pomeru času víťaza a súťažiaceho.

Ako vstupný parameter dostáva:
1. inputList - pole bežcov, kde bežec má meno (ID), zabehnutý čas, bodové ohodnotenie (na vstupe pravdepodobne 0)
2. evalType - typ hodnotenia na základe požiadavku užívateľa
3. resp 4. valueA, valueB - hodnoty užívateľa potrebné na ohodnotenie
5. percento bežcov použité na výpočet priemerného času preteku

Konštruktor vykonáva priradenie potrebných premenných a rozvetvuje program na tri vetvy na základe evalType, to znamená, spúšťa funkcie definované nižšie na základe evalType. V prípade výberu prvých dvoch spoôsobov hodnotenia zistí najrýchlejší čas a priradí ho do self.winner, v prípade treťom vypočíta priemerný čas preteku na základe zadaného percenta a priradí ho do premennej self.average. Z týchto funkcii sa výsledok priradí do premennej self.outputList, kde už sú bežci ohodnotení.

### 3.1.1 def first (self):

Vypočíta prvý spôsob hodnotenia, definovaný ako: (čas víťaza / čas bežca) * valueA
Priradí ohodnotenie každému bežcovi zo vstupného poľa a toto pole vráti.

### 3.1.2 def second (self):

Vypočíta druhý spôsob hodnotenia, definovaný ako: maximum(0, (2 - čas bežca / čas víťaza)) * valueA
Priradí ohodnotenie každému bežcovi zo vstupného poľa a toto pole vráti.

### 3.1.3 def third (self):

Vypočíta tretí spôsob hodnotenia, definovaný ako: maximum(0, valueA + valueB * (self.average - (čas bežca / self.average)))
Priradí ohodnotenie každému bežcovi zo vstupného poľa a toto pole vráti.

## 3.2 Evaluation2

Táto trieda má za úlohu ohodnotiť bežcov na základe zadaného intervalu medzi jednotlivými bodmi pre bežcov, a bodového ohodnotenia posledného bežca.

Ako vstupný parameter dostáva:
1. inputList - pole bežcov, kde bežec má meno (ID), zabehnutý čas, bodové ohodnotenie (na vstupe pravdepodobne 0)
2. interval - interval medzi jednotlivými bodmi pre bežcov
3. lastPoints - bodové ohodnotenie posledného bežca

Konštruktor vykonáva priradenie potrebných premenných. Potom vytvorí premennú self.evalTable, ktorá je výsledok funkcie listCreate. Slúži na jednoduché hodnotenie podľa vstupných parametrov. Následne spustí funkciu evall, ktorá už pracuje s premennou self.evalTable. Z tejto funkcie sa výsledok priradí do premennej self.outputList, kde už sú bežci ohodnotení.

### 3.2.1 def createList (self):

Má za úlohu vytvoriť pole zrozumiteľné pre funkciu evall. Vezme lastPoints a interval a na základe týchto dvoch údajov vygeneruje pole bodových ohodnotení tak, že prvý prvok poľa je bodové ohodnotenie víťaza, druhé druhého bežca, a tak postupne až po posledný prvok, ktorý je bodovým ohodnotením posledného bežca = lastPoints.

### 3.2.2 def evall (self):

Vypočíta hodnotenie bežcov na základe evalTable, definovaný tak, že víťaz dostane ohodnotenie prvé v poli evalTable, a potom každý ďalší bežec ďalšie ohodnotenie.
Ak je bežcov viac, ako hodnôt v poli, dostanú bežci navyše také ohodnotenie, aké je v poslednom prvku poľa.
Funkcia následne priradí ohodnotenie každému bežcovi zo vstupného poľa a toto pole vráti.

## 3.3 Evaluation3

Táto trieda má za úlohu ohodnotiť bežcov na základe bodovej tabuľky definovanej užívateľom. Ide o pole hodnôt, usporiadané podľa toho, aké body chceme priradiť bežcom, to znamená prvá hodnota poľa patrí víťazovi, druhá druhému bežcovi a tak ďalej.

Ako vstupný parameter dostáva:
1. inputList - pole bežcov, kde bežec má meno (ID), zabehnutý čas, bodové ohodnotenie (na vstupe pravdepodobne 0)
2. evalTable - tabulka hodnotenia - pole bodových ohodnotení bežcov od prvého bežca

Konštruktor vykonáva priradenie potrebných premenných. Následne spustí funkciu evall. Z tejto funkcie sa výsledok priradí do premennej self.outputList, kde už sú bežci ohodnotení.

### 3.3.1 def evall (self):

Vypočíta hodnotenie bežcov na základe evalTable, definovaný tak, že víťaz dostane ohodnotenie prvé v poli evalTable, a potom každý ďalší bežec ďalšie ohodnotenie.
Ak je bežcov viac, ako hodnôt v poli, dostanú bežci navyše také ohodnotenie, aké je v poslednom prvku poľa.
Funkcia následne priradí ohodnotenie každému bežcovi zo vstupného poľa a toto pole vráti.

## 3.4 Evaluation4

Táto trieda má za úlohu ohodnotiť bežcov na základe bodovej tabuľky definovanej užívateľom. Táto tabuľka sa skladá z dvojice percento-body, kde 100 percent je čas víťaza, a každé ďalšie percentá v tabuľke určujú bodové ohodnotenie bežca, ktorý toto percento dosiahol.

Ako vstupný parameter dostáva:
1. inputList - pole bežcov, kde bežec má meno (ID), zabehnutý čas, bodové ohodnotenie (na vstupe pravdepodobne 0)
2. evalTable - tabulka hodnotenia - pole dvojíc bodových ohodnotení a prislúchajúcich percent

Konštruktor vykonáva priradenie potrebných premenných. Taktiež zistí, ktorý bežec zabehol najlepší čas a priradí to do premennej self.winner. Následne spustí funkciu evall. Z tejto funkcie sa výsledok priradí do premennej self.outputList, kde už sú bežci ohodnotení.

### 3.4.1 def evall (self):

Bežci dostanú body podľa percenta a hodnôt v evalTable, kde percento znázorňuje, koľko času sú za víťazom. Ak je bežec pod posledným percentom v evalTable, dostane body na základe tohoto percenta.
Funkcia následne priradí ohodnotenie každému bežcovi zo vstupného poľa a toto pole vráti.