Žilinská Univerzita v Žiline

Fakulta Riadenia a Informatiky



Algoritmy a údajové štruktúry 2

Semestrálna práca č. 2

Tobiáš Hládek

5ZII11

# Triedy

Semestrálna prácu som implementoval v jazyku Swift, ktorý je primárne určený pre vývoj iOS a Mac OS aplikácií. Úlohou bolo vytvoriť systém na správu nehnuteľností, ktorý bude využívať minimálne miesto v operačnej pamäti.

Na vytvorenie používateľského rozhrania som sa rozhodol použiť návrhový vzor MVVM, kde každý “view” má svoj vlastný “view model”.

Storable

Interface, ktorý slúži na ukladanie jednotlivých štruktúr do súboru a ich následné načítanie do operačnej pamäti.

Hashable

Tento interface musí použiť objekt, ktorý chceme vloźiť do implementovanej štruktúry. Každý takýto objekt musí implementovať počítanú premennú typu BitSet

Blockable

Obdoba interface-u Storable pre bloky, ktoré sú ukladané do súboru, kde sa využíva blokovací faktor.

Block

Generická trieda, ktorá slúži na uchovanie jednotlivých záznamov v tzv. “bucketoch”. Tieto následne načítavame do operačnej pamäti. Trieda block obsahuje informácie o počte platných záznamov, hĺbke bloku, blokovacom faktore a samotné záznamy.

BlockInfo

Je trieda, z ktorej je tvorený adresár v hlavnej triede ExtendibleHashing. Táto trieda obsahuje minimálne množstvo informácií o bloku, ktoré potrebujeme na urýchlenie procesov. neobsahuje však samotné záznamy.

ExtendibleHashing

Je to hlavná trieda, v ktorej sa odohráva velková logika rozšíriteľného hešovania. Triedu je rovnako možné uložiť do súboru a následne načítať. v triede sa nachádzajú inštancie manažérov súborov pre hlavný, preplňovací a konfiguračný súbor. V triede sú držané oba adresáre - pre hlavný aj pre preplňovací súbor.

PDAState

Trieda, ktorá udržiava stav aplikácie. Stará sa o spojenie dátovej a grafickej vrstvy. Zabezpečuje uloženie štruktúry pri vypnutí aplikácie. je implementovaná návrhovým vzorom “Singleton”.

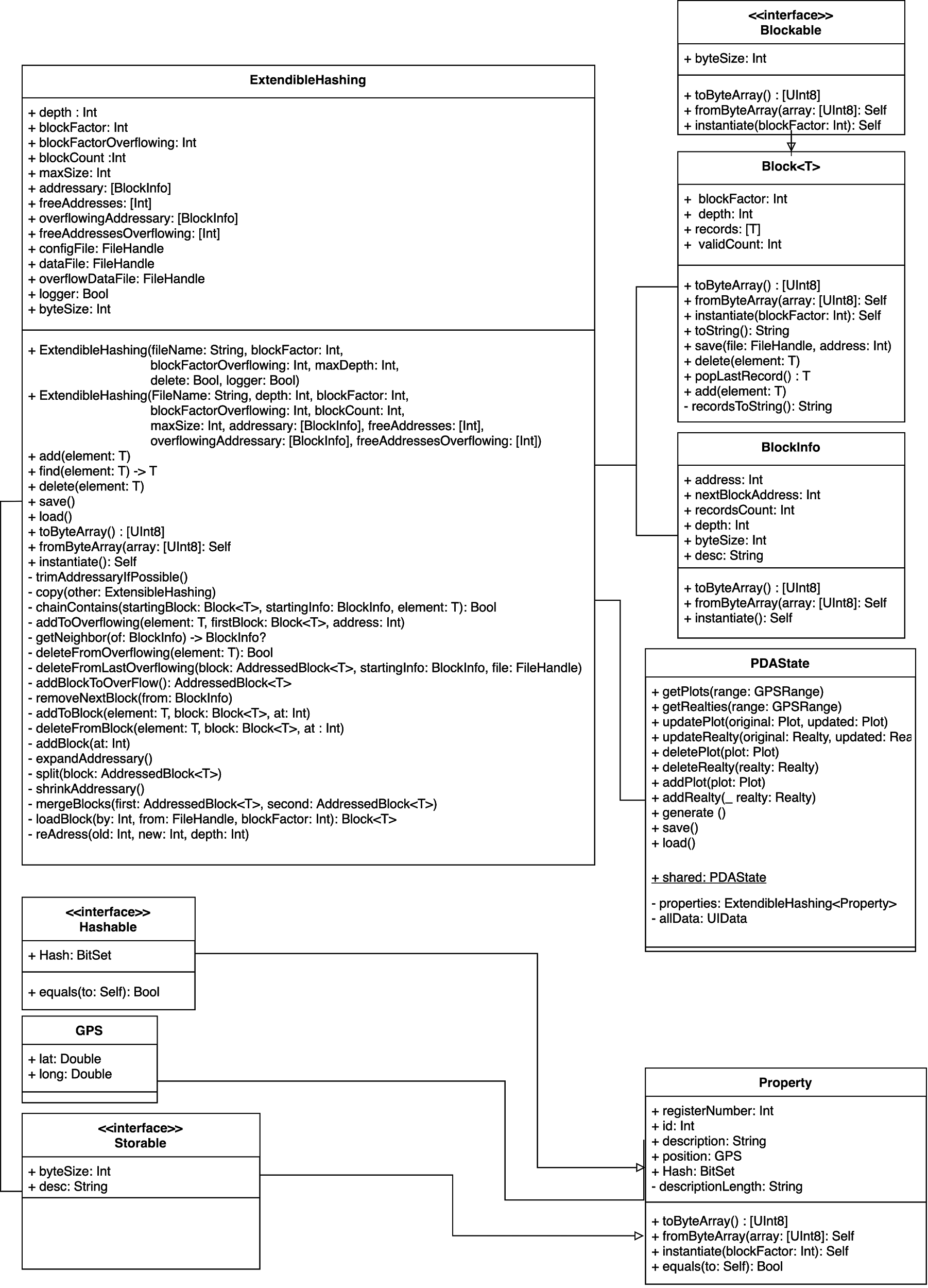
Property

Finálna implementácia objektu, ktorý je možné uložiť do implementovanej štruktúry. Obsahuje informácie o nehnuteľnosti.

UITests

Trieda zastrešujúca všetky testy: insert, update, delete a náhodné operácie

1. UML diagram



1. Zložitosti

**n** = počet zreťazených blokov

Insert

Blok, do ktorého vkladám vypočítam na základe hešovacej funkcie a adresári nájdem jeho adresu.

Blok do ktorého vkladám má voľné miesto:

Načítam blok, pridám záznam a znovu uložím (**2** prístupy)

Blok do ktorého pridávam nemá voľné miesto:

Nedosiahli sme maximálnu hĺbku:

Načítam blok, rozdeľujem, kým potrebujem, zapíšem do

súboru. Počet prístupov je teda: **1 + 2\*počet delení + 1**

Dosiahli sme maximálnu hĺbku:

Blok má nasledovníka:

Načítam blok, postupne prejdem bloky až k

poslednému zreťazenému blok,

doplním záznam a uložím.Zreťazené bloky musím prechádzať kvôli unikátnosti záznamov.

Počet prístupov: **1 + n + 1** **(+ 2 v prípade, že posledný blok je plný)**

Blok nemá nasledovníka:

Načítam blok, pridám blok, pridám záznam a uložím

blok. Počet prístupov: **1 + 1** **+ 1**

Find

Prvý nájdený blok nemá nasledovníka:

Načítam blok, vyberiem záznam. Počet prístupov: **1**

Prvý nájdený blok má nasledovníka:

Načítam blok, postupne prehľadávam záznamy v zreťazených

blokoch. Počet prístupov: **2 až 1 + n**

Prvý nájdený blok neobsahuje žiadny záznam:

Počet prístupov: **0**

Delete

Situácie po zmazaní záznamu.

Blok má suseda:

Blok nemá nasledovníka:

Počet záznamov v bloku + počet záznamov v

susednom bloku >= blokovací faktor:

Vymažem záznam z načítaného bloku

a následne blok uložím.

Počet prístupov: **1 + 1**

Počet záznamov v bloku + počet záznamov v

susednom bloku < blokovací faktor:

Vymažem záznam z bloku, spájam bloky, kým

mi to podmienky umožňujú. Spájanie požaduje

4 prístupy. 2 na načítanie blokov a 2 na uloženie.

Počet prístupov: **1 + 4 \* počet spojení**

Blok má nasledovníka:

Počet záznamov v bloku + počet záznamov v

susednom bloku >= blokovací faktor:

V operačnej pamäti sa dostanem na adresu

posledného zreťazeného bloku. Z toho vyberiem

záznam, ktorý následne uložím na miesto,

ktoré vzniklo vymazaním záznamu.

Oba bloky následne uložím.

Počet prístupov: **1 + 1 + 1 + 1**

Počet záznamov v bloku + počet záznamov v

susednom bloku < blokovací faktor:

Vymažem záznam z bloku, spájam bloky, kým

mi to podmienky umožňujú. Spájanie požaduje

4 prístupy. 2 na načítanie blokov a 2 na uloženie.

Tu treba dbať na to aby sa zachoval sused

spájaného bloku.

Počet prístupov: **1 + 4 \* počet spojení**

Blok nemá suseda:

Blok nemá nasledovníka:

Vymažem záznam z načítaného bloku

a následne blok uložím.

Počet prístupov: **1 + 1**

Blok má nasledovníka:

V operačnej pamäti sa dostanem na adresu posledného

zreťazeného bloku. Z toho vyberiem záznam, ktorý následne

uložím na miesto, ktoré vzniklo vymazaním záznamu.Oba

bloky následne uložím.

Počet prístupov: **1 + 1 + 1 + 1**