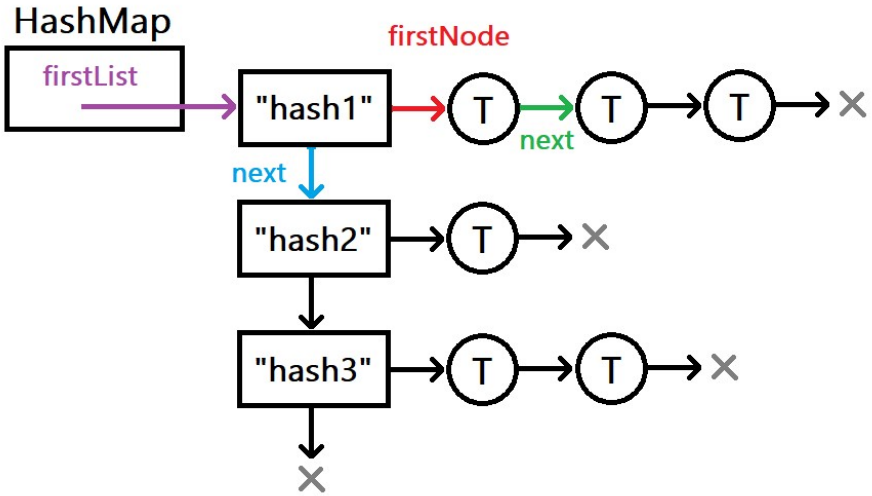
Dizajniranje Softvera

Popravni kolokvijum 2023

Vreme za izradu: **2h** | Broj poena: **20**

Potrebno je kreirati generičku kolekciju podataka **HashMap**. Ova kolekcija čuva podatke u odvojenim listama, tako da se podaci sa istim ***hash***-om nalaze u istoj listi. ***Hash*** je vrednost tipa ***string*** koja se računa na osnovu vrednosti nekog podatka, i omogućava nam da slične podatke, tj. one koji imaju isti hash, čuvamo jedne blizu drugih, tj. u istoj listi.



Napomene:

- **Nije dozvoljeno** korišćenje **kolekcija** podataka iz sistemskih biblioteka, kao što je npr. **List**
- **Iterator** je **dozvoljeno** realizovati i pomoću *yield*-a
- Sortiranje liste je dozvoljeno realizovati na bilo koj način
- U fajlovima postavke se nalazi kod za klasu **Student** koji treba doraditi prema postavci zadatka
- **Indekseri** vraćaju **null** ukoliko element sa datim indeksom ne postoji

Potrebno je kreirati **konzolnu aplikaciju** koja se sastoji iz sledećih delova:

Interfejs **IHash** koji sadrži:

- Metodu **GetHash** koja nema parametre a vraća **string**. Namena metode je da izračuna i vrati **hash** string objekta kako bi na osnovu tog hash-a mogao da bude svrstan u odgovarajuću listu u **HashMap**-i

Klasa **Node<T>** čuva podatak tipa **T** koji mora biti referencni tip i mora implementirati interfejs **IHash**, a sadrži:

- Promenljive:
  - **value** tipa **T** koja čuva referencu na podatak
  - **next** tipa **Node<T>** koja pokazuje na naredni **node** u listi

Klasa **HashList<T>** predstavlja listu izgrađenu od **node**-ova, a sadrži:

- Promenljive:
  - **hash** tipa **string** koji predstavlja vrednost hash-a koju imaju elementi koji se čuvaju u toj listi
  - **firstNode** tipa **Node<T>** koja pokazuje na prvi **node** u listi
  - **next** tipa **HashList<T>** koja pokazuje na narednu **listu** u **HashMap**-i
- Metoda **Add** sa parametrom tipa **T** i bez povratne vrednosti. Dodaje prosleđenu vrednost na kraj liste
- **Iterator** koji redom prolazi po **node**-ovima u listi, vraća **Node<T>** (podrška za *foreach* petlju)
- **Indekser** koji prima **int** i vraća vrednost **T** iz **node**-a na prosleđenoj poziciji u listi. Indeksiranje počinje od 0
- **Operator +** koji prima dva **HashList<T>** parametra i vraća nov objekat tipa **HashList<T>** koji sadrži sve elemente prve i druge prosleđene liste redom, leve pa desne. Ako liste imaju različit hash, operator vraća praznu listu.

Klasa **HashMap<T>** predstavlja kolekciju izgrađenu od hash lista, a sadrži:

- Promenljivu **firstList** tipa **HashList<T>** koja pokazuje na prvu hash listu, koja dalje pokazuje na naredu, itd.
- Svojstvo (*property*) **Length** tipa **int** koje vraća ukupan broj elemenata u hash mapi, tj. ukupan broja elemenata iz svih hash listi (samo ***get***)
- Metode:
  - **Add** sa parametrom tipa **T** i bez povratne vrednosti. Dodaje prosleđenu vrednost u hash listu koja čuva vrednosti sa onim hash-om koji ima prosleđena vrednost. Ukoliko takva lista ne postoji, potrebno je kreirati novu hash listu za taj hash, dodati element u nju, i okinuti event **NewHash**
  - **Sort** bez parametara i povratne vrednosti. Sortira hash liste u okviru hash mape, tako da budu poređane leksikografski po hash-u. Primer upoređivanja stringova je dat na slici ispod postavke
- Event **NewHash** koji se okida kada se u hash mapu dodaje element za čiji hash ne postoji hash lista. Funkcije koje se prijavljuju na event vracaju **void** a primaju **string** koji predstavlja taj hash
- **Iterator** koji prolazi po hash listama u hash mapi, vraća **HashList<T>** (podrška za *foreach* petlju)
- **Indekser** koji za prosleđeni **string hash** vraća listu **HashList<T>** koja odgovara tom hash-u
- **Operator +** koji prima dva **HashMap<T>** parametra i vraća nov objekat tipa **HashMap<T>** koji sadrži sve elemente prosleđenih hash mapa
- **Eksplicitna konverzija** tipa **HashMap<T>** u tip **T[ ]**. Novonastali niz treba da sadži sve elemente iz hash mape, iz svih njenih listi

Klasa **Student** treba da implementira interfejs **IHash** kako bi objekti te klase mogli da se čuvaju u **HashMap**-i. Metoda **GetHash** treba da vrati **godinu** upisa kao **string**, i po tome će se studenti razvrstavati u hash mapi.

U **Main** metodi isprobati funkcionalnosti **HashMap**-e. Obavezno je kreirati jednu **anonimnu funkciju** (pomoću bilo koje sintakse) i jednu **statičku metodu** i prijaviti ih na event **NewHash**. Te funkcije treba da ispišu hash koji im se prosledi.

Primer korišćenja metode **CompareTo**. Vraća **int** vrednost po sledećem šablonu:

```
string s = "B";
s.CompareTo("A"); //  1 - s ide nakon parametra
s.CompareTo("B"); //  0 - s je po redosledu isti kao i parametar
s.CompareTo("C"); // -1 - s ide pre parametra
```