Generics

Implement generic `DefaultDictionary<K, V>` that wraps normal `Dictinary<K, V>`. It should implement `IDictionary<K, V>` interface.

Unlike the normal dictionary this one should add a new key when trying to access a non existing key. This will be performed by the `defaultFactory` delegate that will be called to create a new value and this value will be inserted under the key and returned.

`defaultFactory` should be generic, readonly and be injected via constructor.

```
var d = new DefaultDictionary<string, int>(key => 0);
d["thing"]; // returns 0
```

Implement the application that counts word frequency from standard input.

OOP

Model fictional banking system that has multiple Account types. All account types have current amount on them, and also an account holder id and account id. You can put or take money from them. Different accounts have different behaviour.

- `NormalAccount` the most basic type, you can fill the money with it and you can draw money from it. When drawing and going overdraw should return `false` and not draw anything.
- `OverdrawableAccount` like the Normal Account but the current amount can go negative.
- `LinkedAccount` represents a collection of different accounts. It's current amount is the sum of the linked accounts. When drawing should prioritize drawing from the linked accounts with the biggest current amount. When adding amount should prioritize putting it into the linked account with the smallest current amount.

Use inheritance and abstract classes.

OOP - Optinal

Write some cli interface to interact with your system.