```
interface NetbankingService {
  double getBalance(String
accountNo);
}
```

```
class NetbankingServiceImpl
implements NetbankingService {
  public double getBalance(String
accountNo) {

    // fetch the balance of the
account by going to database
    return balance;
  }
}
```

```
class CachedNetbankingService implements NetbankingService {
  NetbankingService netbankingService;
  Cache cache;

  public CachedNetbankingService(NetbankingService
netbankingService) {
    this.netbankingService = netbankingService;
  }

  public double getBalance(String accountNo) {
    double balance = 0.0;

    cache = Cache.getInstance();
    if(cache.containsKey("getBalance("+accountNo+")") {
      balance = cache.get("getBalance("+accountNo+")");
      return balance;
    }
    balance = netbankingService.getBalance(accountNo);
    cache.put("getBalance("+accountNo+")", balance);
    return balance;
  }
}
```

How to apply additional functionality to our class?
use proxy design pattern

How to add additional functionality to a subset or group of classes within our application?

#1 solution
Per each class we wanted to add additional functionality create an proxy class
drawback:
with this approach we endup in creating several proxy classes, wherein all of them has the same functionality duplicated across

is there a way we can apply the additional functionality to group of classes without duplicating?
Yes, that is where Runtime proxies are introduced.

```
interface StatementService {
  List<Transaction>
getLast10Transactions(String accountNo);
}
```

```
class OnScreenStatementServiceImpl
implements StatementService {
  List<Transaction>
getLast10Transactions(String accountNo) {
    // fetch the last #10 transactions
    return transactions;
  }
}
```

```
class CachedStatementService implements StatementService {
  Cache cache;
  StatementService statementService;
  public CachedStatementService(StatementService statementService) {
    this.statementService = statementService;
  }
  public List<Transaction> getLast10Transactions(String accountNo) {
    List<Transaction> transactions = null;

    cache = Cache.getInstance();
    if(cache.containsKey("getLast10Transactions("+accountNo+")") {
      transactions = cache.get("getLast10Transactions("+accountNo+")");
      return transactions;
    }
    transactions = netbankingService.getBalance(accountNo);
    cache.put("getLast10Transactions("+accountNo+")", transactions);
    return transactions;

  }
}
```