Description	Points
Static review - multithreading	
Each request is handled by a thread (new or thread pool)	3
Connections and buffers are closed where needed	2
Can explain design decisions (i.e. why thread pool, why the size of the threadpool)	3
Transfer functionality	
A variable to hold all accounts is created - the type of the variable is appropriate	2
A method for searching an account is created	2
Accounts are read from file	2
Transfer considers all possible errors: account not existent, balance not enough, etc	2
Static review - race conditions and deadlock	
Can describe when a race condition might occur and the solution implemented	3
Uses an appropriate mechanism to avoid race condition (atomic variable, lock)	5
Can describe when a deadlock might occur and the solution implemented	3
Uses an appropriate mechanism to avoid deadlock(lock ordering, detecting deadlock	5
Static review code quality	
Methods, parameters and variables have meaningful names	2
Code is well-documented, and comments help understanding	2
Code is broken into classes and methods where appropriate. No code duplication	2
Follows consistent coding standards (i.e. camelCase)	2
Errors are correctly handled, not just print	2
Execution	
Code compiles and executes without errors	5
Can handle 1000 clients concurrently without race conditions (final balance is the expected one). Run at least 3 times	15
When executing clients transsactions in the same accounts, no deadlock occurs (i.e transfer from 123 to 321 and from 321 to 123 in different clients)	15
User interface is updated succesfully showing results of transfer	2
Report	
Explains the code structure and strategies for multithreading	5
Explains the code structure and strategies for race conditions / deadlock	5
Explains the code structure and strategies for transfer functinality	5
UML diagram preferred	3
Testing plan includes various conditions: succesful, error	3
	100