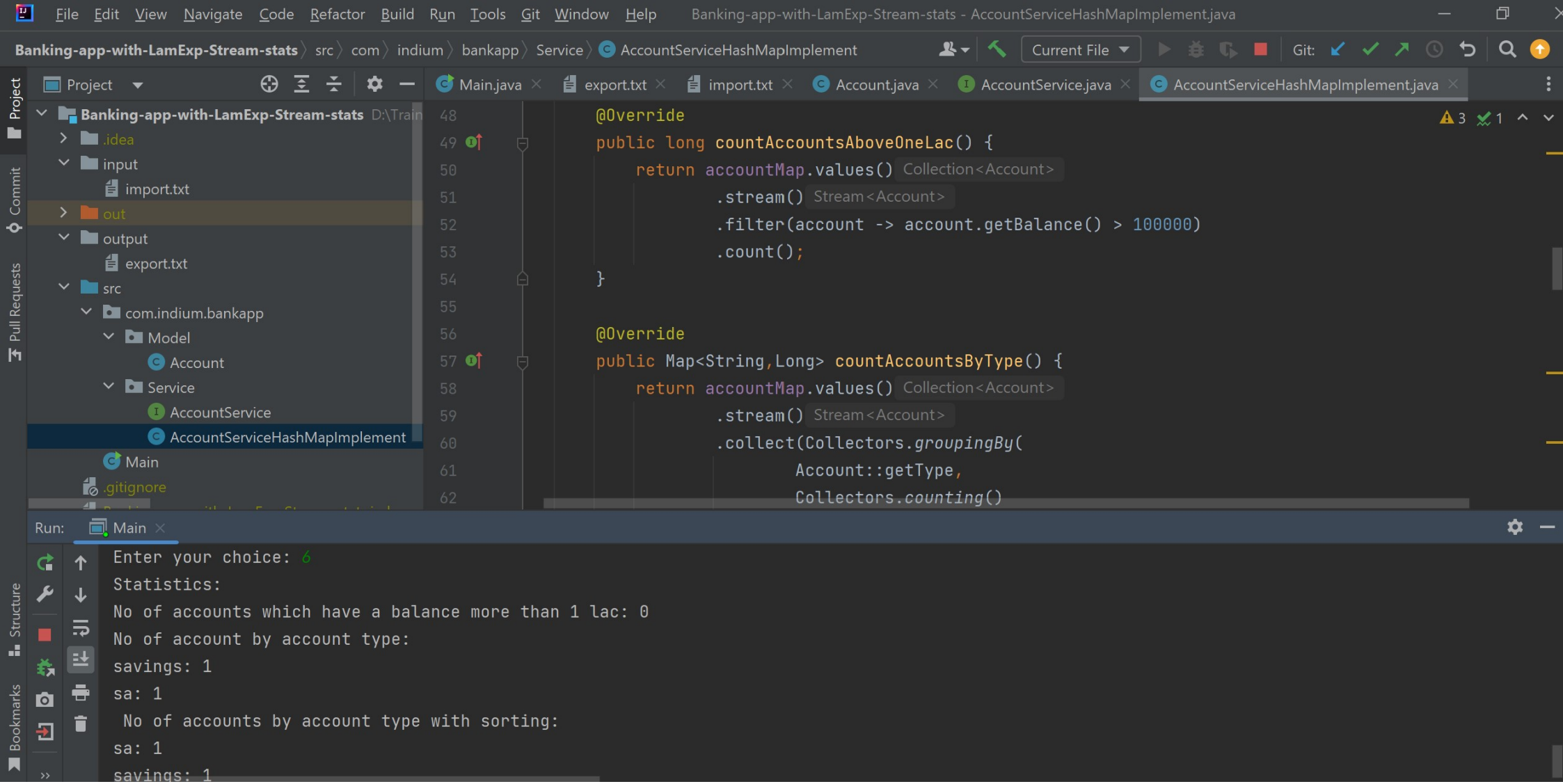
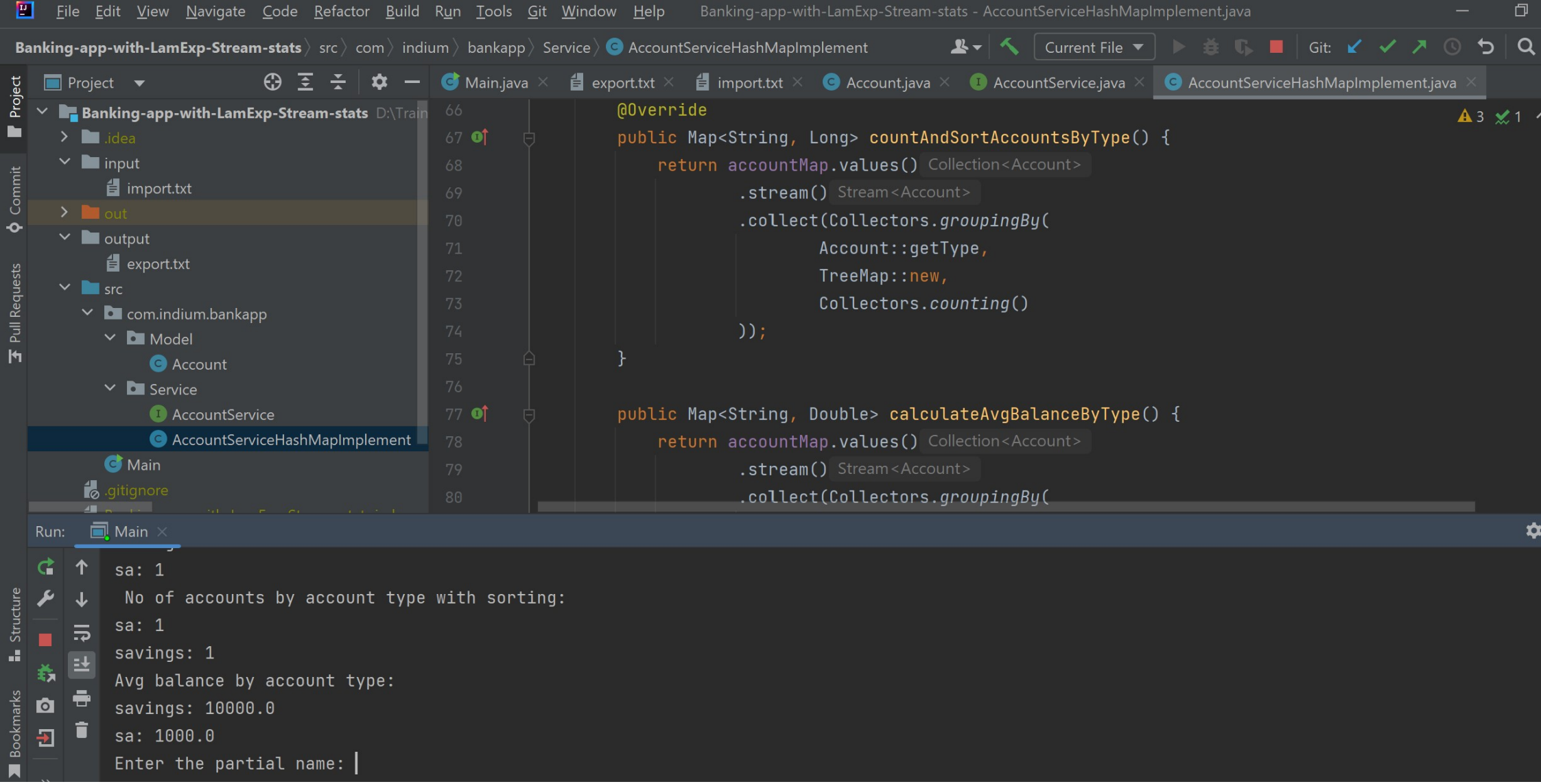

File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Banking-app-with-LamExp-Stream-stats - AccountServiceHashMapImplement.java
Banking-app-with-LamExp-Stream-stats 〉 src 〉 com 〉 indium 〉 bankapp 〉 Service 〉 AccountServiceHashMapImplement
Current File
Git:
Project
Banking-app-with-LamExp-Stream-stats D:\Train
.idea
input
import.txt
out
output
export.txt
src
com.indium.bankapp
Model
Account
Service
AccountService
AccountServiceHashMapImplement
Main
.gitignore
Main.java
export.txt
import.txt
Account.java
AccountService.java
AccountServiceHashMapImplement.java
48 @Override
49 public long countAccountsAboveOneLac() {
50 return accountMap.values() Collection<Account>
51 .stream() Stream<Account>
52 .filter(account -> account.getBalance() > 100000)
53 .count();
54 }
55
56 @Override
57 public Map<String,Long> countAccountsByType() {
58 return accountMap.values() Collection<Account>
59 .stream() Stream<Account>
60 .collect(Collectors.groupingBy(
61 Account::getType,
62 Collectors.counting()
Run: Main
Enter your choice: 6
Statistics:
No of accounts which have a balance more than 1 lac: 0
No of account by account type:
savings: 1
sa: 1
No of accounts by account type with sorting:
sa: 1
savings: 1


File Edit View Navigate Code Refactor Build Run Tools Git Window Help
Banking-app-with-LamExp-Stream-stats - AccountServiceHashMapImplement.java
Banking-app-with-LamExp-Stream-stats 〉 src 〉 com 〉 indium 〉 bankapp 〉 Service 〉 AccountServiceHashMapImplement
Current File
Git:
Project
Banking-app-with-LamExp-Stream-stats D:\Train
.idea
input
import.txt
out
output
export.txt
src
com.indium.bankapp
Model
Account
Service
AccountService
AccountServiceHashMapImplement
Main
.gitignore
Main.java
export.txt
import.txt
Account.java
AccountService.java
AccountServiceHashMapImplement.java
@Override
public Map<String, Long> countAndSortAccountsByType() {
return accountMap.values() Collection<Account>
.stream() Stream<Account>
.collect(Collectors.groupingBy(
Account::getType,
TreeMap::new,
Collectors.counting()
));
}
public Map<String, Double> calculateAvgBalanceByType() {
return accountMap.values() Collection<Account>
.stream() Stream<Account>
.collect(Collectors.groupingBy(
Run:
Main
sa: 1
No of accounts by account type with sorting:
sa: 1
savings: 1
Avg balance by account type:
savings: 10000.0
sa: 1000.0
Enter the partial name: