**=============== Banking System Project ==============**

* Banking System is a project that uses Thread concepts to do WIthdrawal and deposit operations.
* Bank Account contains id, account number, name and balance.
* We use HashMap to maintain the cache and the database to store the accounts.
* We have used executorService to run threads.
* We have used synchronization for withdrawal and deposit functions, because multiple threads can be run at a time.
* We have generated a random number to choose the operations, if the number is even we did deposit operation, while it is odd we did withdraw operation.
* We have a total of 5 accounts in the database, so we have given the size of the cache to 3.
* We have also maintained the hit and miss counts, so it will be easy to calculate statistics.
* As the statistics are displaying immediately after the for loop without the threads being completely executed, we have stopped the Main Thread by using sleep() method for 5000 milli seconds.
* During the for loop, the operations performed are printed and after the sleep time, since all the threads have already been executed we get correct cache statistics.
* In the end, the correct cache statistics are being printed.

**CODE:**

package com.mycompany.app.BankingSystem;

import java.io.IOException;

import java.util.\*;

import java.util.concurrent.\*;

import java.util.concurrent.atomic.AtomicInteger;

public class BankingSystem {

private final Map<Integer, BankAccount> cache=new ConcurrentHashMap<>();

private final Map<Integer,BankAccount> dataBase=new HashMap<>();

private final int cache\_Size=3;

private final AtomicInteger cache\_hits=new AtomicInteger(0);

private final AtomicInteger cache\_Misses=new AtomicInteger(0);

private static class BankAccount{

private int id;

private int acc\_no;

private String name;

private double balance;

BankAccount(int id,int acc\_no, String name, double balance){

this.id = id;

this.acc\_no=acc\_no;

this.name=name;

this.balance=balance;

}

public String getName(int id){

return name;

}

public double getBalance(int id){

return balance;

}

public synchronized void withDraw(int amt){

if(balance>=amt){

System.out.println("With Drawn ammount"+amt);

balance-=amt;

System.out.println("Remaining balance:"+balance+"\n");

}

else{

System.out.println("Insufficient Funds"+"\n");

}

}

public synchronized void Deposit(int amt){

System.out.println("Deposited amount is:"+amt);

balance+=amt;

System.out.println("After deposit the balance is:"+balance+"\n");

}

@Override

public String toString() {

return "Id: "+id+", Account Number: "+acc\_no+", Account Holder: "+name+", Balance: "+balance+"\n";

}

}

public void getCacheStatistics(){

System.out.println("Cache Hits: "+cache\_hits.get());

System.out.println("Cache Misses: "+cache\_Misses.get());

System.out.println("Cache Size: "+cache.size());

System.out.println("Cache Capacity: "+cache\_Size);

System.out.println("Cache Efficiency: "+((double)cache\_hits.get()/(cache\_hits.get()+cache\_Misses.get())\*100)+"%");

System.out.println("Cache Hit Ratio: "+((double)cache\_hits.get()/(cache\_hits.get()+cache\_Misses.get())\*100)+"%");

System.out.println("Cache Miss Ratio: "+((double)cache\_Misses.get()/(cache\_hits.get()+cache\_Misses.get())\*100)+"%");

}

private void addToCache(int id, BankAccount account) {

while(cache.size()>=cache\_Size)

removeCacheElement();

cache.put(id,account);

}

private void removeCacheElement() {

int id = cache.keySet().iterator().next();

cache.remove(id);

}

public BankAccount getAccount(int id) {

BankAccount account = cache.get(id);

if(account!=null) {

cache\_hits.incrementAndGet();

System.out.println("Cache Hit Book Found: "+id);

return account;

}

else {

cache\_Misses.incrementAndGet();

System.out.println("Cache Miss Book Not Found: "+id);

BankAccount accdb = dataBase.get(id);

if(accdb!=null)

addToCache(id,accdb);

return accdb;

}

}

public BankingSystem(){

dataBase.put(1,new BankAccount(1,1,"Vishnu",24000));

dataBase.put(2,new BankAccount(2,2,"Udaya",23450));

dataBase.put(3,new BankAccount(3,3,"Lalitha",23540));

dataBase.put(4,new BankAccount(4,4,"Madhuri",42350));

dataBase.put(5,new BankAccount(5,5,"Anushka", 54320));

}

public static void main(String[] args){

BankingSystem bank=new BankingSystem();

ExecutorService executor = Executors.newFixedThreadPool(5);

System.out.println("Starting Bank Operations: ");

int[] arr = {1,2,3,4,5};

for(int i=0;i<=10;i++) {

System.out.println("=============== loop"+i+" ===============\n");

int id = arr[new Random().nextInt(arr.length)];

int random = new Random().nextInt(10);

executor.execute(()->{

try {

if(random%2==0) {

BankAccount acc = bank.getAccount(id);

if(acc!=null) {

acc.Deposit(1000);

System.out.println(acc);

}

else {

System.out.println("Account not found\n");

}

}

else {

BankAccount acc = bank.getAccount(id);

if(acc!=null) {

acc.withDraw(300);

System.out.println(acc);

}

else {

System.out.println("Account not found\n");

}

}

}

catch(Exception e){

e.printStackTrace();

}

finally {

try {

executor.awaitTermination(1,TimeUnit.SECONDS);

}

catch(InterruptedException e) {

System.out.println("There is an interruption");

e.printStackTrace();

}

}

});

}

try {

Thread.sleep(5000);

}

catch(Exception e) {

System.out.println("There is an exception");

}

bank.getCacheStatistics();

System.out.println();

}

}