Race condition using multithreading

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | package com.crunchify.tutorial;    /\*\*  \* @author Crunchify.com  \*  \*/    public class CrunchifyRaceCondition {    public static void main(String[] args) {  CrunchifyBankAccount crunchifyAccount = new CrunchifyBankAccount("CrunchifyAccountNumber");    // Total Expected Deposit: 10000 (100 x 100)  for (int i = 0; i < 100; i++) {  CrunchifyTransaction t = new CrunchifyTransaction(crunchifyAccount, CrunchifyTransaction.TransactionType.DEPOSIT\_MONEY, 100);  t.start();  }    // Total Expected Withdrawal: 5000 (100 x 50)  for (int i = 0; i < 100; i++) {  CrunchifyTransaction t = new CrunchifyTransaction(crunchifyAccount, CrunchifyTransaction.TransactionType.WITHDRAW\_MONEY, 50);  t.start();    }    // Let's just wait for a second to make sure all thread execution completes.  try {  Thread.sleep(1000);  } catch (InterruptedException e) {  System.out.println(e);  }    // Expected account balance is 5000  System.out.println("Final Account Balance: " + crunchifyAccount.getAccountBalance());  }  } |

CrunchifyTransaction.java

Java

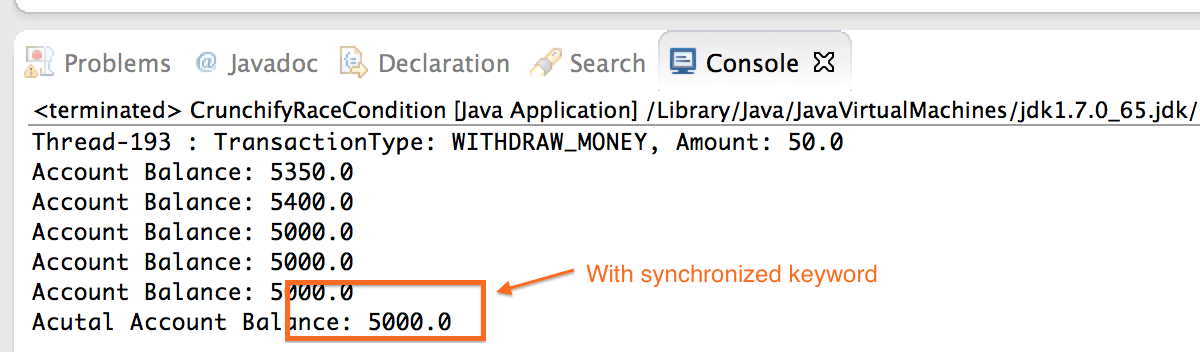
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56 | package com.crunchify.tutorial;    /\*\*  \* @author Crunchify.com  \*/    class CrunchifyTransaction extends Thread {    public static enum TransactionType {  DEPOSIT\_MONEY(1), WITHDRAW\_MONEY(2);    private TransactionType(int value) {  }  };    private TransactionType transactionType;  private CrunchifyBankAccount crunchifyAccount;  private double crunchifyAmount;    /\*  \* If transactionType == 1, depositAmount() else if transactionType == 2 withdrawAmount()  \*/  public CrunchifyTransaction(CrunchifyBankAccount crunchifyAccount, TransactionType transactionType, double crunchifyAmount) {  this.transactionType = transactionType;  this.crunchifyAccount = crunchifyAccount;  this.crunchifyAmount = crunchifyAmount;  }    public void run() {  switch (this.transactionType) {  case DEPOSIT\_MONEY:  depositAmount();  printBalance();  break;  case WITHDRAW\_MONEY:  withdrawAmount();  printBalance();  break;  default:  System.out.println("NOT A VALID TRANSACTION");  }  }    public void depositAmount() {  this.crunchifyAccount.depositAmount(this.crunchifyAmount);  }    public void withdrawAmount() {  this.crunchifyAccount.withdrawAmount(crunchifyAmount);  }    public void printBalance() {  System.out.println(Thread.currentThread().getName() + " : TransactionType: " + this.transactionType + ", Amount: " + this.crunchifyAmount);  System.out.println("New Account Balance: " + this.crunchifyAccount.getAccountBalance());  }  } |

CrunchifyBankAccount.java

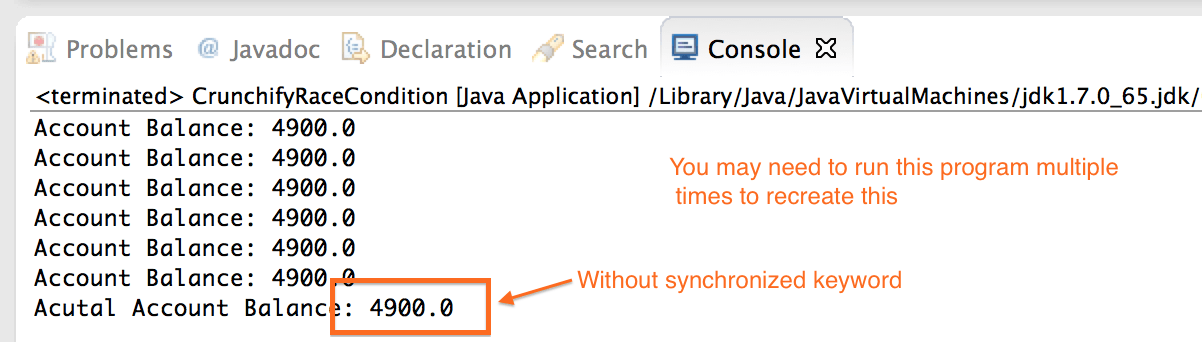
Java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42 | package com.crunchify.tutorial;    /\*\*  \* @author Crunchify.com  \*/    class CrunchifyBankAccount {  private String crunchifyAccountNumber;  private double crunchifyAccountBalance;    public String getAccountNumber() {  return crunchifyAccountNumber;  }    public double getAccountBalance() {  return crunchifyAccountBalance;  }    public CrunchifyBankAccount(String crunchifyAccountNumber) {  this.crunchifyAccountNumber = crunchifyAccountNumber;  }    // Make a note of this line -- synchronized keyword added  public synchronized boolean depositAmount(double amount) {  if (amount < 0) {  return false;  } else {  crunchifyAccountBalance = crunchifyAccountBalance + amount;  return true;  }  }    // Make a note of this line -- synchronized keyword added  public synchronized boolean withdrawAmount(double amount) {  if (amount > crunchifyAccountBalance) {  return false;  } else {  crunchifyAccountBalance = crunchifyAccountBalance - amount;  return true;  }  }  } |

Please check line 24 and 34 above. Keep that Synchronized keyword and run your program. You should see correct result as you see it in below image.

[](http://cdn.crunchify.com/wp-content/uploads/2014/09/Java-Synchronized-Block-Keyword-Crunchify.png)

Now **remove synchronized keyword from line 24 and 34 and run the same program**. You may need to run this program multiple times to see an issue. In java there is no guarantee you will see Race condition all the times.

[](http://cdn2.crunchify.com/wp-content/uploads/2014/09/Java-without-Synchronized-Block-Keyword-Crunchify.png)

If you have enterprise level application and you are talking of millions of transaction per seconds then race condition may cause disaster for your company